Simple algorithms for partial point set pattern matching under rigid motion☆

Arijit Bishnu[a], Sandip Das[b], Subhas C. Nandy[b,*], Bhargab B. Bhattacharya[b]

[a] *Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur 721 302, India*
[b] *Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata 700 108, India*

Abstract

This paper presents simple and deterministic algorithms for *partial point set pattern matching* in 2D. Given a set P of n points, called sample set, and a query set Q of k points ($n \geqslant k$), the problem is to find a matching of Q with a subset of P under rigid motion. The match may be of two types: *exact* and *approximate*. If an exact matching exists, then each point in Q coincides with the corresponding point in P under some translation and/or rotation. For an approximate match, some translation and/or rotation may be allowed such that each point in Q lies in a predefined ε-neighborhood region around some point in P. The proposed algorithm for the exact matching needs $O(n^2)$ space and $O(n^2 \log n)$ preprocessing time. The existence of a match for a given query set Q can be checked in $O(k\alpha \log n)$ time in the worst-case, where α is the maximum number of equidistant pairs of point in P. For a set of n points, α may be $O(n^{4/3})$ in the worst-case. Some applications of the *partial point set pattern matching* are then illustrated. Experimental results on random point sets and some fingerprint databases show that, in practice, the computation time is much smaller than the worst-case requirement. The algorithm is then extended for checking the exact match of a set of k line segments in the query set with a k-subset of n line segments in the sample set under rigid motion in $O(kn \log n)$ time. Next, a simple version of the approximate matching problem is studied where one point of Q exactly matches with a point of P, and each of the other points of Q lie in the ε-neighborhood of some point of P. The worst-case time and space complexities of the proposed algorithm are $O(n^2k^2 \log n)$ and $O(n)$, respectively. The proposed algorithms will find many applications to fingerprint matching, image registration, and object recognition.

Keywords: Point set pattern matching; Approximate matching; Fingerprint matching; Computational geometry; Algorithms; Complexity

1. Introduction

In computer vision and related applications, point sets represent some spatial features like spots, corners, lines, curves in various images pertaining to fingerprints, natural scenery, air traffic, astronomical maps, etc. Pose estimation involves assessment of object position and orientation relative to a model reference frame. In many problems of pattern recognition, such as registration and identification of an object, a suitably chosen set of points may efficiently preserve the desired attributes of the object. In all such cases, the problem can be transformed to matching point sets with templates.

The objective of the *point set pattern matching* (PSPM) problem is to determine the resemblance of two point sets P and Q (representative of two different objects), under different transformations. The most simple kind of transformation is translation. If rotation is allowed along with translation, it is called rigid motion or congruence. Other transformations include reflection and scaling. The latter refers to magnifying (or reducing) the object by a certain factor π. Combination of rotation, translation and scaling is called similarity. Under these transformations, the problem can be classified into three groups [1]:

Exact PSPM: Let P and Q be two sets of points in $\mathbb{R}^d$ with $|P| = |Q| = n$ (say). The objective is to report the

☆ Part of this work has appeared in the Proceedings of the Italian Conference on Algorithms and Complexity, CIAC 2003, Lecture Notes in Computer Science, vol. 2653, pp. 36–45, 2003. This work was funded in part by a grant from the Intel Corporation, USA.

* Corresponding author. Tel.: +91 33 2575 3006; fax: +91 33 2577 3035.
E-mail address: nandysc@isical.ac.in (S.C. Nandy).

existence of a congruence between the two sets under some transformation. In other words, whether each point in Q can be positioned on a point in P under the said transformation of the entire set Q.

Partial PSPM: Let $|P| = n$, $|Q| = k$, and $k \leqslant n$, the problem is to ascertain whether a subset of P matches with Q under some transformation. Here P is referred to as the *sample set*, and Q is the *query set*. The subset or partial matching is a difficult problem than the equal cardinality matching as there can be $\binom{n}{k}$ possibilities and also no condition can be fixed about the centroid of the points.

Approximate PSPM: Here, given two finite sets of points P and Q ($|P| = n$, $|Q| = k$ and $k \leqslant n$), the problem is to find an approximate matching under some transformation, i.e., for each point $q_i \in Q$, find its match $p_i \in P$ such that q_i lies in the ε-neighborhood of p_i (a predefined ε-region centered at p_i), where ε is specified as an input. The most common ε-region is a circle or an axis-parallel square.

Our study on PSPM has been motivated by an important application, namely fingerprint minutiae matching, used as a tool for biometric authentication. It requires very fast congruence checking. There may be more than one sample set. Given a query point set, the objective is to report the sample sets whose at least one k-subset exactly matches with Q in an efficient manner. Another important application of PSPM is image registration, where the objective is to match the feature points extracted from a sensed image to their counterparts in the reference image. An efficient robust heuristic for the partial PSPM is proposed by Mount et al. [2]. It is based on a nice blend of branch-and-bound and Monte Carlo algorithms, and the experimental results (on remote sensed data) show favorable performance. In computational molecular biology, PSPM is extensively used for molecular docking, where the objective is to decide whether the ligands (small molecules) can sterically fit into its designated cavities (the active sites of some protein) [3]. The approximate version of PSPM finds many useful applications in pattern recognition and computer vision [4–9]. A detailed survey on PSPM along with many important applications was reported by Alt and Guibas [1].

1.1. Prior works

The study on the exact point set pattern matching problem was initiated by Atallah [10]. Here the two sets P and $Q \subset \mathbb{R}^2$, $|P| = |Q| = n$, and the objective was to check the existence of a one-to-one matching in the sense that if each point in Q is joined by its matched point in P by straight line segments, then the n line segments become non-intersecting. The proposed algorithm runs in $O(n \log^2 n)$ time. Atkinson [11] showed that exact point pattern matching under Euclidean congruence (i.e., with appropriate rotation followed by translation and then using reflection, if needed) in 3D can easily be reduced to string matching [12], and can be solved in $O(n \log n)$ time. In $\mathbb{R}^d$, the problem is first studied in Ref. [13]. The time complexity of the proposed algorithm is $O(n^{d-2} \log n)$. Later, the time complexity of the general problem was improved to $O(n^{\lfloor d/2 \rfloor} \log n)$ [14]. The best known algorithm for this problem runs in $O(n^{\lfloor d/3 \rfloor} \log n)$ time [15].

The one-dimensional version of the exact partial PSPM problem can be solved by sorting the points in both P and Q with respect to their coordinates on the respective real lines, and then performing a merge like pass among the two sets by considering the distances of consecutive members in the respective sets. Using the same idea, Rezende and Lee [16] showed that given a query set Q of k points, the existence of its match in a sample set P of n ($\geqslant k$) points in $\mathbb{R}^d$ can be reported in $O(kn^d)$ time under translation and rotation. They also extend the idea to include scaling. The implementation of their algorithm needs the circular ordering of points in $P \setminus \{p_i\}$ around each point $p_i \in P$; this needs $O(n^d)$ time for all the points in P [17]. The algorithm proposed in Ref. [16] solves the 2D version of the partial PSPM problem in $O(kn^2)$ time using $O(n^2)$ space. For testing the congruence in 2D (i.e., if only translation and rotation are considered), the time complexity of the best known algorithm is $O(kn^{4/3} \log n + \mathscr{A})$ [18], where $\mathscr{A}$ is the time complexity of locating rth smallest distance among a set of n points in the plane. In Ref. [19], it is shown that $\mathscr{A}$ is $O(n^{4/3} \log^{8/3} n)$ in the average case, and $O(n^{4/3+\varepsilon})$ in the worst-case. Determination of $\mathscr{A}$ needs parametric searching technique [20]. Though efficient implementation of some specific problems using parametric search technique is proposed [21], it is hard to implement in general [21–23]. The Las Vegas expected time of the algorithm proposed by Akutsu et al. [18] is $O(n^{4/3} \log^{8/3} n + min\{k^{0.77} n^{1.43} \log n, n^{4/3} k \log n\})$, which also uses parametric searching. Thus, the algorithms proposed in Ref. [18] are nice theoretical results on the time complexity of the problem, but are not suitable for practical applications.

The approximate PSPM is more realistic in many actual applications. Baird [4] did some pioneering work on this area, but the problem of designing a polynomial time algorithm was left open. Alt et al. [13] solved the one-to-one version of this problem (i.e., where $k = n$) in its most general setting using intersection of curves of high degree and bipartite graph matching. Better algorithms for some restricted cases and with $k = n$ were given in [24]. An effort of the approximate matching of partial point set under rigid motion and using Hausdroff distances appeared in Ref. [7]. It determines the smallest ε required for a match of Q ($|Q| = k$) with a subset of size k from P. Later, the time complexity of the same problem is improved in [25] using parametric searching technique [20]. The proposed algorithm runs in $O(n^2 k^3 \log^2 kn)$ time. It may be noted that the techniques used in Refs. [13,24,25] involve either computing the intersection of curves of high degree making it numerically unstable or use parametric searching. Thus, the algorithms are conceptually complex, difficult to implement and have usually high running time [7].

Apart from point patterns, research has been done on detecting resemblance of more complex patterns, for example line segments or polygons [45,46]. The approximate pattern matching of polygonal shapes is studied in Ref. [26]. It assumes that the number of vertices in the two polygons P and Q are same, and it minimizes the Hausdorff distance of P and $I(Q)$, where $I(Q)$ is the transformation of Q under rigid motion. If P and Q are two sets of polygons with n and k line segments respectively, then the minimum Hausdorff distance between P and Q is ε if the vertices of Q lies in the ε-neighborhood of a k-subset of P. An algorithm for computing the minimum Hausdorff distance between two sets of polygons is given in Ref. [25]. This also uses parametric searching, and runs in $O(k^3n^3\log^2 kn)$ time. In Ref. [27], an $O(n)$ time algorithm is given for computing the Hausdorff distance of a pair of convex polygons. The Hausdorff distance of two arbitrary sets of line segments can be computed in $O(n \log n)$ time [26]. In addition, several polynomial time algorithms for checking approximate resemblance between two polygons with possibly different number of vertices are given in Ref. [26]. Several applications of line segment matching can be found in map matching [28], cartography [29], aerial scene matching [30], edge linking problems of image processing [31], to name a few. Polygon matching problem finds its use in shape retrieval from image databases [32].

1.2. Main results

We present a simple and easy to implement deterministic algorithm for the partial PSPM under translation and rotation. Our proposed algorithm creates efficient data structure for each sample point set in the preprocessing phase, and it leads to an efficient query algorithm which does not need the complicated parametric searching technique. The time and space complexities of the preprocessing are $O(n^2 \log n)$ and $O(n^2)$, respectively, and the query time complexity is $O(k\alpha \log n)$ in the worst-case, where α is the maximum number of equidistant pairs of points in the set P. In Ref. [33], it is shown that α may be $O(n^{4/3})$ in the worst-case for a point set of size n. Thus for the applications where repeated queries need to be answered, our algorithm is an improvement over the best known existing algorithm where the query time complexity is $O(n^{4/3}\log^{8/3}n + k\alpha \log n)$ [18], and which uses parametric searching technique. Our empirical evidences show that α is much less than $O(n^{4/3})$. The algorithm presented by Rezende and Lee [16] appears to be the most efficiently implementable algorithm. Thus, we have compared our proposed algorithm against Ref. [16]. We have run experiments on random point sets and also for some actual applications, like fingerprint matching. In all the cases, our algorithm performs much better than that of Rezende and Lee [16].

We then extend the exact matching algorithm to match a set of query line segments with a k-subset of a sample set under rigid motion. This algorithm runs in $O(kn \log n)$ time in the worst-case using $O(n)$ space.

In actual applications, a more useful problem is the approximate matching of partial point set. By approximate matching of a pair of points a and b, we mean that each one (say a) lies in the circle of radius ε centered at the other point b, where ε is specified as an input. Here a match exists if each point in the query set Q approximately matches with a point in the sample set P under rigid motion of Q. The existing algorithms for approximate matching of point set patterns in 2D are computationally unstable since they use computation of intersections of high degree curves. Moreover, earlier algorithms often use parametric searching, which is difficult to implement. We have studied a simplified variation of this problem, where an ε-approximate match is searched under rigid motion with the constraint that one member of Q exactly matches with a member in P. This type of approximate matching is useful in image registration problems by control points [34]. Under this restricted scheme, if a match is not found with an approximation factor 2ε, we can safely say that there does not exist any k-subset in P that has an ε-approximate match with Q in the conventional sense. But the converse is not true, i.e., the existence of a 2ε-matching using our scheme does not guarantee a conventional ε-approximate matching.

The rest of the paper is organized as follows. The problem of exact matching for partial point set under rigid motion is studied in Section 2. The exact matching problem for line segments is studied in Section 3. Finally, a variation of the approximate matching problem for point set is studied in Section 4. Concluding remarks appear in Section 5.

2. Point set pattern matching

Let $P = \{p_1, p_2, \ldots, p_n\}$ be the sample set and $Q = \{q_1, q_2, \ldots, q_k\}$ be a query set, $k \leqslant n$. The objective is to find a subset of size k in P that exactly matches Q under translation and/or rotation. The most trivial method is inspecting all possible $\binom{n}{k}$ subsets of P for a match with Q. The time complexity is improved in Ref. [16] by extending the concept of *circular sorting* [17] to higher dimensions, which facilitates an orderly traversal of the point sets. It may be noted that the points in $\mathbb{R}^d$, $d \geqslant 2$, lack a total order. A characterization of canonical ordering of a point set in $\mathbb{R}^d$ is first reported in Ref. [35]. In Ref. [16], the authors pointed out that the method given in Ref. [35], though elegant, may not be of much use in subset matching. But, the idea of canonical ordering of Ref. [35] can be extended to the concept of circular sorting [17], which basically imposes a partial order on the points. The algorithm in Ref. [16] stores the circular order of the points in $P\backslash\{p_i\}$ with each point $p_i \in P$. Then it selects a point, say q_1, from the query set Q, and computes the circular order of the points in $Q\backslash\{q_1\}$ around q_1. It anchors q_1 with each point in P, and rotates Q to identify a k-subset match in P. This scheme can detect a match (if any) under translation, rotation and scaling. In 2D, the space and query time complexities of this algorithm are $O(n^2)$ and $O(n^2+k \log k+kn(n-1))=O(kn^2)$, respectively.

We follow a different scheme for detecting a match under congruence (translation and rotation only) by selectively choosing a few points in P for anchoring with q_1. We adopt a preprocessing method on the sample set and use the following facts to design an efficient query algorithm with reduced number of anchorings.

Fact 1. *As the distances amongst the points in a set are preserved under translational or rotational transformations, a sufficient condition for a match of Q with a k-subset of P under translation and/or rotation is that all the $\binom{k}{2}$ distances in the point set Q should occur among the $\binom{n}{2}$ distances in the point set P.*

Fact 2 (Szekely [33]). *For a sample set P of n points, the number of equidistant pairs of points is $O(n^{4/3})$ in the worst-case.*

2.1. Preprocessing

Let $P=\{p_1, p_2, \ldots, p_n\}$ be a sample set, where each point in P is assigned a unique label. In the entire text, we shall denote a line joining a pair of points a and b by (a, b), the corresponding line segment by $\overline{ab}$, and the length of the line segment $\overline{ab}$ by $\lambda(\overline{ab})$.

The distances of all $\binom{n}{2}$ pairs of points are computed, and a height balanced tree $\mathscr{T}$ is created with the set of distinct distances. Each node $\delta \in \mathscr{T}$ is attached with a pointer to an array χ_δ, whose each element is the pair of points separated by a distance δ. Each element of the array χ_δ contains a triple (p_i, p_j, ψ_{ij}), where p_i and p_j denote the labels of the pair of points contributing distance δ, and ψ_{ij} denotes the angle of the line (p_i, p_j) with the x-axis.

An array $\mathscr{S}$ of size n is also used during the query. Its ith element (corresponding to the point $p_i \in P$) holds the address of the root of a height balanced tree $\mathscr{S}_i$, containing exactly $n-1$ elements. Each element is a tuple (θ_{ij}, r_{ij}), where $r_{ij}=\lambda(\overline{p_i p_j})$ and θ_{ij} is the angle made by the line (p_i, p_j) with the x-axis.

The total space used by $\mathscr{T}$ is $O(n^2)$. The structure $\mathscr{S}$ also needs $O(n^2)$ space since it has n elements, and each $\mathscr{S}_i$ takes $O(n)$ space. The time complexity for the preprocessing is dominated by sorting the $\binom{n}{2}$ distances, which is $O(n^2 \log n)$ in the worst-case.

2.2. Query answering

Given a set of points $Q=\{q_1, q_2, \ldots, q_k\}$, the query is to ascertain whether a k-subset of P matches with Q under translation and/or rotation. The technique for matching used is based on Fact 1. We select any two points, say $q_1, q_2 \in Q$, and search whether the distance $\lambda(\overline{q_1 q_2})$ is in $\mathscr{T}$. If not, then no k-subset of P matches with Q (by Fact 1). If such an entry is found, the following steps are needed:

Let $\lambda(\overline{q_1 q_2})=\delta$. We consider each member in χ_δ separately. Assume that $\lambda(\overline{p_i p_j}) \in \chi_\delta$ is under processing. We anchor the line segment $\overline{q_1 q_2}$ with the line segment $\overline{p_i p_j}$ by positioning q_1 on p_i (if it fails we would position q_2 on p_i) and search in $\mathscr{S}_i$ (data structure corresponding to p_i in $\mathscr{S}$) to identify the presence of a match.

For each point $q_\alpha \in Q$, the tuple $(\angle q_\alpha q_1 q_2, \lambda(\overline{q_1 q_\alpha}))$ is searched in $\mathscr{S}_i$. If such an element $(\angle p_\ell p_i p_j, \lambda(\overline{p_i p_\ell}))$ is found (i.e., the triangle $\Delta p_\ell p_i p_j$ in P is congruent with the triangle $\Delta q_\alpha q_1 q_2$ in Q), p_ℓ matches with q_α. If all the points in Q are matched with k points of P, then Q matches with P (see Fig. 1 for an illustration).

2.2.1. Query time complexity

Searching for the distance $\lambda(\overline{q_1 q_2})$ in $\mathscr{T}$ needs $O(\log n)$ time. If the search is successful, i.e., $\lambda(\overline{q_1 q_2})=\delta \in \mathscr{T}$, then one needs to consider each member $\overline{p_i p_j} \in \chi_\delta$. The line segment $\overline{q_1 q_2}$ is anchored with the line segment $\overline{p_i p_j}$, and for each point $q_\alpha \in Q \backslash \{q_1, q_2\}$, the tuple $(\angle q_\alpha q_1 q_2, \lambda(\overline{q_1 q_\alpha}))$ is searched in $\mathscr{S}_i$, which needs $O(\log n)$ time. For $k-2$ points in Q, the total time required is $O(k \log n)$. If the match fails, the same steps are repeated by anchoring $\overline{q_1 q_2}$ with $\overline{p_j p_i}$, i.e., searching in S_j.

In the worst-case, one may need to check for all the elements of χ_δ. As the number of elements in χ_δ can be at most $O(n^{4/3})$ (see Fact 2), the worst-case query time complexity of this algorithm is $O(kn^{4/3} \log n)$.

2.3. Experimental results and applications

2.3.1. Experiments on randomly generated point sets

We have implemented the proposed algorithm as well as the one reported in Ref. [16] on a Sun Ultra-5_10, Sparc, 233 MHz; the OS is SunOS Release 5.7 Generic. In most of the pattern matching problems in general, and fingerprint matching in particular, there are two different problems: (i) verification and (ii) identification [36]. A verification system authenticates a person's identity by conducting a one-to-one comparison to determine whether the identity claimed by the individual is true. Verification is basically a query of type "*Am I whom I claim I am?*". On the other hand, identification systems recognize an individual (without self-identification) by searching the entire sample database by performing a one-to-many comparison. Identification is basically a query of type "*Who am I?*". In keeping with the above, we do two types of experiments. The first experiment, whose results are summarized in Table 1, is designed for verification purposes. We randomly generate sample point sets having cardinality n as shown in the first column of Table 1. From that set, we again randomly pick up points to generate the query set whose cardinality k is varied as shown in the second column of Table 1. It can be seen from Table 1 that the proposed algorithm is fast and also depends on the number

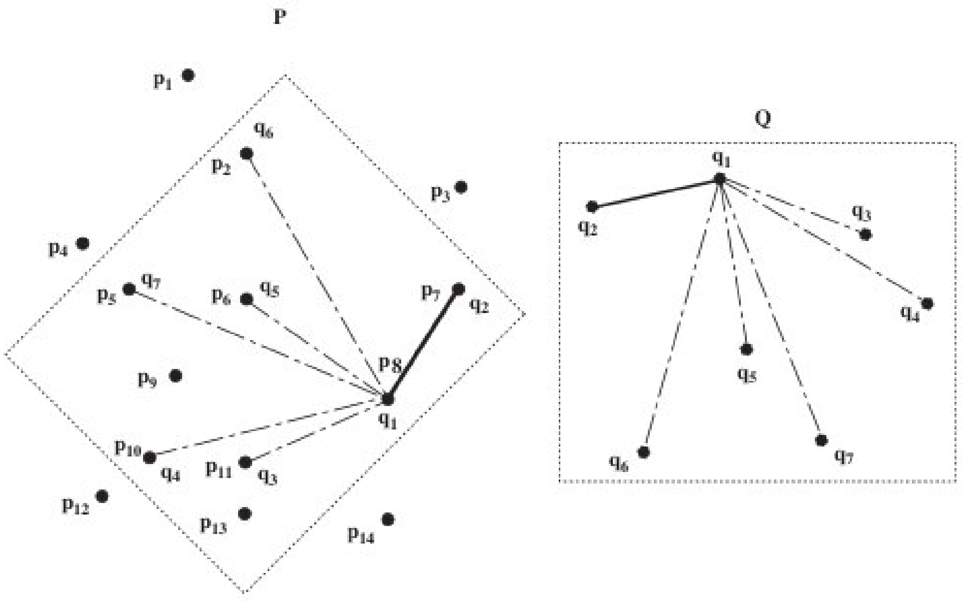

Fig. 1. Illustration of the match with the bold lines showing anchors.

Table 1
Comparative results for verification with respect to CPU time

Number of points in sample, n	Number of points in query, k	Number of anchorings, α	CPU time in µs: Rezende and Lee's method [16]	CPU time in µs: Proposed method	(%) Savings
200	50	1	730.0	22.00	97.00
	100	1	807.40	23.40	97.0
	150	2	857.60	31.20	96.0
500	100	2	1905.0	32.2	98.0
	200	1	1937.20	23.0	99.0
	400	2	1952.80	32.0	98.0
1000	300	12	4037.60	129.0	97.0
	500	7	4054.0	76.0	98.0
	800	11	4098.0	11.0	97.0

Table 2
Comparative results for identification with respect to CPU Time

Number of sample sets, K	CPU time in s: Rezende and Lee's method [16]	CPU time in s: Proposed method	(%) Savings
10	39.70	1.28	96.70
20	85.02	3.46	95.90
50	228.24	11.78	94.83
100	548.03	19.75	96.39
200	1075.98	39.22	96.35

Table 3
Maximum number of equidistant pairs in a point set

Number of points (n)	100	200	500	1000
$\lceil n^{4/3} \rceil$	465	1170	3969	10001
Maximum number of equidistant pairs observed	4	6	20	53

Experiment is performed for randomly generated point sets for different values of n.

of anchorings, α. Table 2 illustrates the experimental performance of the two algorithms for identification purposes. We considered K sample sets $P_i, i = 1, 2, \ldots, K$, each of size ranging from 50 to 80. Thereafter, a particular sample set P_i is selected at random; a subset from that set is then chosen which under random translation and rotation forms the query set Q. Thus, a match of Q with at least one P_i is ensured. Next, both the algorithms for partial PSPM are executed with each of the K sample sets. The time reported is

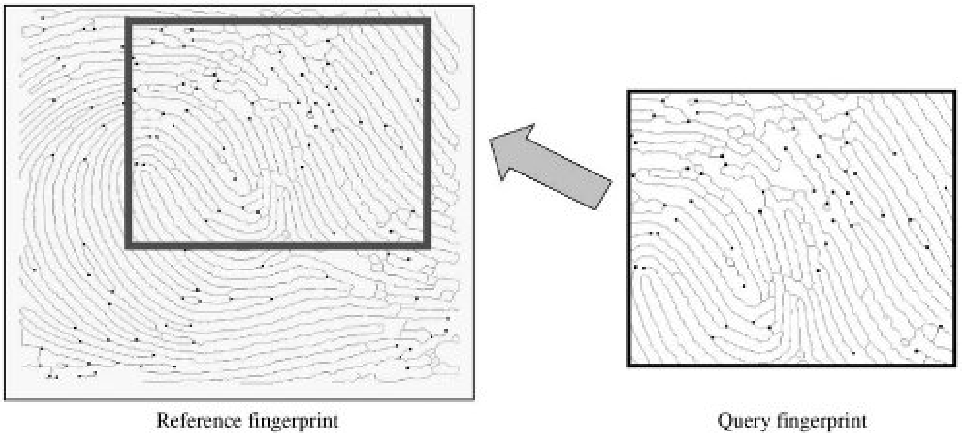

Fig. 2. An example of a subset matching in a fingerprint.

averaged over a number of experiments for different values of K. Also, no false match was reported. It may be noted that, as exact matching is not possible in the real coordinate system, a small tolerance on the distances and angles is allowed. In other words, around each point of the query set, we consider a circle of radius ε; if a point of the query set falls inside the circle corresponding to the desired point, it is considered to be a match. Such a matching is always useful in matching minutiae in fingerprint images. The drastic improvement in the execution time, as reported in the last columns of both Tables 1 and 2, is due to the fact that, in general, the maximum number of equidistant pairs of points in a point-set is very small compared to its theoretical upper bound (see Table 3).

2.3.2. *Experiment with real fingerprint minutiae*

The proposed algorithm is also tested using real fingerprint minutiae, extracted from fingerprints in the NIST 14 sdb [37,38]. The one pixel thick ridge lines are extracted from a grayscale fingerprint image using the algorithm reported in Ref. [39]. Thereafter, minutiae (the terminations and bifurcations on the ridge lines) are extracted using the method in Ref. [40] (see Fig. 2). We analyzed 100 images of the NIST 14 sdb [38] for extracting minutiae. A fingerprint image is then chosen at random; its minutiae are extracted to form the query set (see Fig. 2). The query set is then searched in the database using our proposed algorithm. It always reported a match, and no false match is reported. The preprocessing stage for the database containing 100 images requires 2.3 s on an average, and the search for a particular minutiae point set in the database takes on an average 0.7 s.

3. Line segment matching

We now tailor our earlier algorithm for the partial matching of a query set containing k line segments with a k-subset of the sample set P containing n line segments. Here the preprocessing phase consists of creating two data structures: (i) a Voronoi diagram $VOR(P)$ with the end-points of the line segments in P, and (ii) a height-balanced binary search tree $\mathscr{T}$ containing the length of the line segments $\ell = \overline{p_i p_m} \in P$. The members in $\mathscr{T}$ are all distinct. If more than one line segment are of the same length δ, then they are stored in the structure χ_δ attached to the node δ. Each entry of χ_δ stores a line segment $\ell = \overline{p_i p_m} \in P$. Each entry of χ_δ contains a 4-tuple $\{p_i, p_m, ptr_1, ptr_2\}$, where ptr_1 and ptr_2 point to the faces of $VOR(P)$ containing p_i and p_m, respectively. Construction of $VOR(P)$ and its storing need $O(n \log n)$ time and $O(n)$ space.

During a query operation, a line segment, say $\ell_1 = \overline{q_1 q_2} \in Q$, is chosen from the query set. Let δ = length of ℓ_1. If $\delta \in \mathscr{T}$, then for each line segment in χ_δ, we need to check for a match. We anchor $\overline{q_1 q_2}$ with each member $\overline{p_i p_j} \in \chi_\delta$ (if the match fails, we need to anchor $\overline{q_1 q_2}$ with $\overline{p_j p_i}$), and check the presence of the line segments in $Q \backslash \{\ell_1\}$ among the members in P. This anchoring gives us the parameters of the rigid motion, and we transform the members in Q accordingly. For each line segment $\ell = \overline{qq'} \in Q\backslash\{\ell_1\}$ (in the transformed plane), we search for the faces of $VOR(P)$ containing q and q'. Let the points in P representing these two faces be p and p', respectively. If $\overline{pp'} \in P$ then we have a match. The total time required for searching in $VOR(P)$ for all the members in Q is $O(k \log n)$. Thus, the worst-case query time complexity is $O(k|\chi_\delta| \log n)$, where $|\chi_\delta|$ (the number of elements in χ_δ) may be $O(n)$ in the worst-case.

4. Approximate matching under rigid motions

As mentioned earlier, the exact subset matching problem considered in the previous sections is not realistic in practice, and there is no theoretical guarantee for any exact algorithm to give correct results when bounded precision real coordinates are used in actual implementation. This motivates our study in this section for developing algorithm with theoretical guarantee for a more realistic version of the matching problem with some tolerance. Given the sample set P ($|P|=n$) and a query set Q ($|Q|=k$) of points, $k \leqslant n$, we consider the following problem: decide if there exists a rigid motion that maps each point in Q to a closed ε-neighborhood of some point in P in a one-to-one fashion (see Fig. 3).

Fact 3 (*Alt et al. [13]*). *If there exists a transformation $T(Q)$, which gives an ε-approximate partial matching of the points in Q with a k-subset of P, then there exists another transformation $T'(Q)$ where at least two points $q_i, q_j \in Q$ lie on the boundary of the ε-circle centered at points $p_a, p_b \in P$ and can produce the same matching* (see Fig. 3 *for an illustration*).

The existing algorithms for the approximate matching of point sets in the one-to-one case involve computing the intersection of curves of high degree making it numerically unstable or use parametric searching. Thus, the algorithms are conceptually complex, difficult to implement and have usually high running times [7].

Note that our algorithm for the exact case in 2D is fast because it chooses a correct anchoring point of Q with an appropriate point in P very fast. Though Fact 3 is a necessary and sufficient condition for the existence of an approximate matching, it does not say any rule of anchoring for the searching of a match. This motivates us to study the special case of the problem. We present a simple algorithm for the following special case of approximate matching problem.

One point of Q exactly matches with a point in P, and each of the other points of Q lies inside the ε-circle centered at some point in P. This problem is important in the case of image registration [34] where there is a reference point.

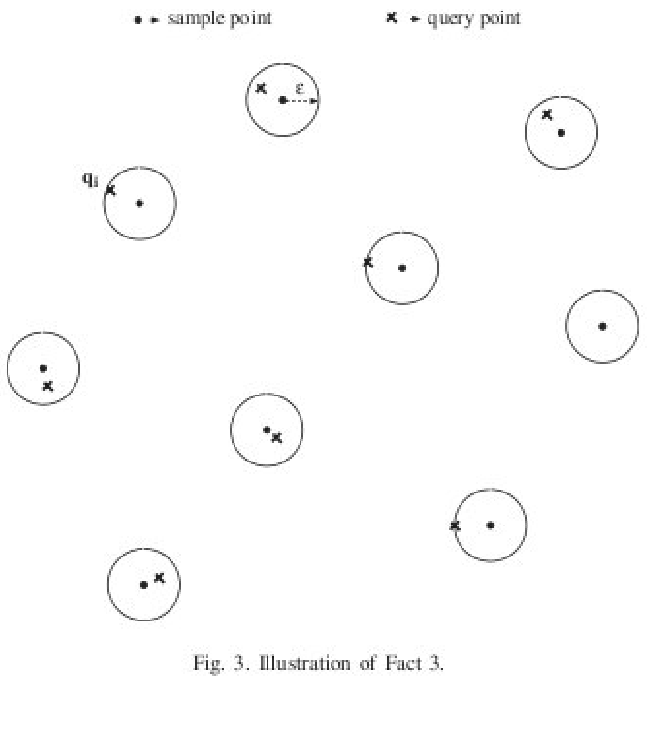

Fig. 3. Illustration of Fact 3.

From now onwards, this will be referred to as *restricted approximate matching problem*.

4.1. Algorithm

In this section, we first assume that the point set P is sparse enough in the sense that the ε-circles do not overlap each other. The lemma, stated below, suggests an anchoring scheme for the restricted approximate matching problem.

Lemma 1. *If there exists a transformation $T(Q)$, which gives an ε-approximate partial matching of the points in Q with a k-subset of P, and one point $q_i \in Q$ coincides with a point $p' \in P$, then there exists another transformation $T'(Q)$, which can produce the same matching, where $T'(q)$ is obtained by coinciding q_i with p' and placing at least one other point $q_j \in Q$, on the boundary of the ε-circle centered at some other point $p'' \in P$, $p'' \neq p'$.*

Proof. If there exists a transformation T as said, then let $q_i \in Q$ match exactly with $p' \in P$ and all other points $q_j \in Q \backslash \{q_i\}$ lie within an ε-circle centered around some point in P. Now, fixing q_i at p', start rotating all points $q_j \in Q \backslash \{q_i\}$ in anticlockwise direction, until a point q_j hits an ε-circle. Note that, at this moment, every other point $q_k \in Q \backslash \{q_i, q_j\}$ lies inside the corresponding ε-circle. □

Lemma 1 indicates the following algorithm for the restricted approximate matching problem. Let the point $q_i \in Q$ be anchored with a point $p' \in P$. Consider each element $q_j \in Q$ and draw a circle C_{ij} centered at q_i and with radius $\overline{q_i q_j}$. As each of these circles may intersect $O(n)$ ε-circles, we may have $O(kn)$ such intersections in the worst-case. For each such intersection (event-point), we need to do the following:

At an event-point, if C_{ij} intersects the ε-circle of $p'' \in P$ at a point π, then transform Q by placing q_i on point p' and q_j at point π, and search for a match with the remaining $k-2$ points in Q. This needs $O(k \log n)$ time.

Thus the total time required for checking the existence of an instance of a match with q_i anchored with p' is $O(nk^2 \log n)$ in the worst-case. As each point in Q is to be anchored with each member in P, the worst-case time complexity of this algorithm is $O(n^2k^3 \log n)$.

The time complexity of this problem can be improved as follows. Let $q_j \in Q$ be the farthest point from q_i, and the existence of a match implies the existence of at least one ε-circle centered at some point in P that will intersect the circle C_{ij}. We identify such an ε-circle. Let it be centered at a point $p'' \in P$, and it intersects C_{ij} at two points π_1 and π_2 (say). We shall denote the angle $\angle \pi_1 q_i \pi_2$ by θ. Now, we align $\overline{q_i q_j}$ with $p'\pi_1$, and determine the co-ordinates of points in Q exactly among the points in P with the necessary translation and rotation. Now, rotate the entire point set Q by an angle θ fixing q_i on point p' (i.e., until q_iq_j is aligned with $p'\pi_2$),

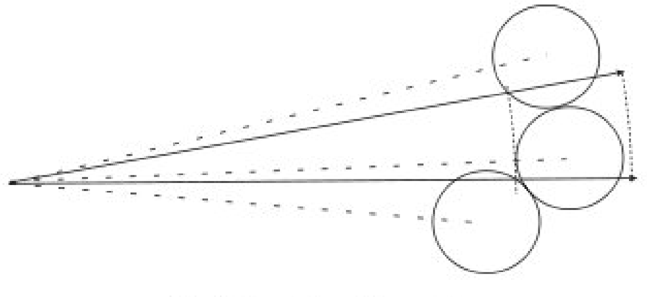
Fig. 4. Illustration of Lemma 2.

and consider all the arcs C_ℓ obtained by the movement of q_ℓ, $\ell \neq i, j$. Thus, each point in $Q \setminus \{q_i\}$ generates an arc satisfying the property stated in the following lemma.

Lemma 2. *The number of* ε*-circles intersected by each of these arcs is* $O(1)$.

Proof. Note that q_j is the farthest point from q_i, and length of the line segment $\overline{\pi_1\pi_2}$ is less than 2ε. Consider another point q_l which is moving around q_i by an angle θ. Let its locus be an arc $[\alpha_1, \alpha_2]$ of a circle of radius $\overline{q_i q_l}$ and centered at q_i (see Fig. 4). We need to prove that the number of ε-circles intersected by the arc $[\alpha_1, \alpha_2]$ is bounded by a constant number.

There may exist two ε-circles touching the points α_1 and α_2, respectively. Next, consider all the ε-circles whose centers are in one side of the arc $[\alpha_1, \alpha_2]$, and are cut by the interior of the arc $[\alpha_1, \alpha_2]$. Let us join their centers with q_i as shown in Fig. 4, and mark the points of intersection of these lines with the arc $[\alpha_1, \alpha_2]$ by $\gamma_1, \gamma_2, \ldots$. Note that the length of the line segment $\overline{\gamma_j \gamma_{j+1}}$ is at least ε. This proves the fact that the number of such ε-circles is at most 3.

Similarly, the number of such ε-circles having center to the other side of the arc $[\alpha_1, \alpha_2]$ and intersecting the arc $[\alpha_1, \alpha_2]$ can also be at most 3. Thus, the number of ε-circles intersecting the arc $[\alpha_1, \alpha_2]$ cannot exceed 8. □

We create a set $\mathscr{A}$, each element of which is an arc corresponding to the locus of a point in $Q \setminus \{q_i\}$ inside the ε-circle of some point in P. Surely, $[0, \theta] \in \mathscr{A}$ and it corresponds to the point q_j. The members in $\mathscr{A}$ can define an interval graph [41]. As the ε-circles corresponding to the points in P are assumed to be non-intersecting, if more than one member in $\mathscr{A}$ correspond to the same member of Q, then they must be non-overlapping. We compute all the cliques of this graph. If there exists a clique of size $k-1$, there exists a matching.

4.2. Complexity analysis

Theorem 1. *The time complexity of our proposed algorithm for the restricted approximate matching problem is* $O(n^2 k^2 \log n)$.

Table 4
Results on CPU times for approximate matching

Number of points in sample, n	Number of points in query, k	CPU time in seconds
50	12	0.102621
	25	0.414141
	37	0.341578
100	25	1.0838
	50	5.37816
	75	7.99264
200	50	20.4028
	100	88.0108
	150	139.606

Proof. The anchoring of each $q_i \in Q$ is to be made with all the members of P. While anchoring q_i with a member $p' \in P$, if q_j is farthest from q_i, then the circle of radius $\overline{q_i q_j}$ and centered at q_i may intersect at most $O(n)$ ε-circles. For each such intersection, we need to find the cliques of an interval graph with a set of intervals $\mathscr{A}$ as mentioned above. By Lemma 2, $|\mathscr{A}| = O(k)$, and each member in $\mathscr{A}$ can be recognized in $O(\log n)$ time by using the Voronoi diagram of the point set P. Thus, we have the proof of the result. □

4.3. Experimental results

In order to demonstrate the efficacy of the above algorithm, some experiments are performed with randomly generated point sets. Let P be such a point set. To ensure disjointness of the ε-neighborhoods of P, we do the following: (i) compute the distance δ between the closest pair of points in P, and (ii) set ε as strictly less than $\delta/2$. Next, to construct Q, we choose a subset of the points from P randomly; it is then translated, rotated and perturbed with a tolerance of ε to generate Q. Thus, with such a data set, the algorithm should report a match. The implementation of the algorithm requires several geometric computations, and we used LEDA [42,43] for this purpose. The entire experiment is performed on the same setup as was done for the exact PSPM problem, and the results are shown in Table 4. In our experiment, even if a match is found during the execution, we have continued searching for multiple matches. Thus, the CPU time estimates also indicate the worst-case scenario, considering the fact that a match may not be found in an actual data set.

4.4. Applications

The direct application of the restricted version of approximate matching problem can be found in the image registration problem [34]. Another motivation of studying the problem arises from its relationship with the conventional problem of approximate PSPM, where a match of a query

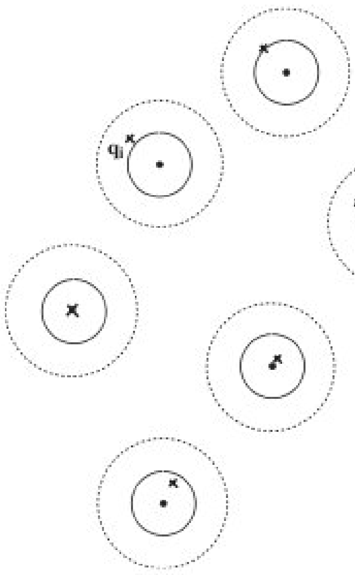

Fig. 5. Illustration of Fact 4.

set Q with a k-subset of P is said to occur if there exists a transformation $T(Q)$, which brings each $q_i \in Q$ within an ε-circle centered at some point $p_\alpha \in P$ in a one-to-one sense. The following facts are relevant in this context.

Fact 4. *If there exists a transformation $T(Q)$ for an approximate matching under the conventional sense, then there exists another transformation $T'(Q)$, where (i) one point, say $q_i \in Q$, lies at the center of the ε-circle of a point in P, (ii) each of the other members in Q lies within the 2ε-circle of some point in P (see Fig. 5).*

Fact 5. *As the distances amongst the points in a set are preserved under rigid motions, a sufficient condition for a match of Q with a k-subset of P under rigid motion is that for a distance Δ_q among any two points in Q there should exist a distance Δ_p among any two points in P satisfying $\Delta_q - 2\varepsilon \leqslant \Delta_p \leqslant \Delta_q + 2\varepsilon$.*

These facts lead to the following result:

Result 1. *If a 2ε approximate matching of Q with a k-subset of P does not exist in the restricted sense, then there does not exist an ε-approximate matching of Q with a k-subset of P in the conventional sense. But the converse is not true.*

This result has a significance in identification problems (as mentioned in Section 2.3) in the case of approximate matching that looks for a match by performing a one-to-many comparison. It can also be used for faster rejection of point sets. If our algorithm reports a match, then further accurate confirmation can be achieved by running other high-complexity approximate matching algorithms [13,24].

5. Conclusion

In this paper, new algorithms for the exact and approximate PSPM problems in 2D have been proposed. The algorithms are simple and easy to implement. They can handle multiple queries on the same sample set, and do not need complicated tools used for parametric searching and computation of intersections of high degree curves. Experimental results on exact matching of partial point set reveal that although the worst-case time complexity of the algorithm is $O(kn^{4/3} \log n)$, it terminates very fast in most of the practical cases. Similar results are also observed for approximate matching. An interesting open problem is to design a robust and implementable algorithm for the general problem of approximate matching of partial point set (i.e., a matching implies when each query point appears in the ε-circle of some point in the sample set).

Acknowledgment

We thank Mr. Arindam Biswas of Bengal Engineering and Science University for his help in the experiments.

References

[1] H. Alt, L.J. Guibas, Discrete geometric shapes: matching, interpolation, and approximation—a survey, Technical Report B 96-11, Freie Universität Berlin, 1996.
[2] D.M. Mount, N.S. Netanyahu, J.L. Moigne, Efficient algorithms for robust feature matching, Pattern Recognition 32 (1999) 17–38.
[3] R. Norel, D. Fischer, H. Wolfson, R. Nussinov, Molecular surface recognition by a computervision based technique, Protein Eng. 7 (1994) 39–46.
[4] H.S. Baird, Model based image matching using location, Distinguished Dissertation Series, MIT Press, Cambridge, MA, 1984.
[5] W. Eric, L. Grimson, Object Recognition by Computer: The Role of Geometric Constraints, MIT Press, Cambridge, MA, 1990.
[6] D. Forsyth, J.L. Mundy, A. Zisserman, C. Coelho, A. Heller, C. Rothwell, Invariant descriptors for 3-D object recognition and pose, IEEE Trans. PAMI 13 (1991) 971–991.
[7] M.T. Goodrich, J.S.B. Mitchell, M.W. Orletsky, Approximate geometric pattern matching under rigid motions, IEEE Trans. PAMI 21 (4) (1999) 371–379.
[8] R.M. Haralick, C.N. Lee, X. Zhuang, V.G. Vaidya, M.B. Kim, Pose estimation from corresponding point data, IEEE Trans. SMC 19 (1989) 1426–1446.
[9] D. Huttenlocher, S. Ullman, Recognizing solid objects by alignment with an image, Int. J. Comput. Vision 5 (1990) 195–212.
[10] M.J. Atallah, A matching problem in the plane, J. Comput. Syst. Sci. 31 (1985) 63–70.
[11] M.D. Atkinson, An optimal algorithm for geometrical congruence, J. Algorithms 8 (1987) 159–172.
[12] D.E. Knuth Jr., J.H. Morris, V.R. Pratt, Fast pattern matching in strings, SIAM J. Comput 6 (2) (1977) 240–267.
[13] H. Alt, K. Mehlhorn, H. Wagener, E. Welzl, Congruence similarity and symmetries of geometric objects, Discrete Comput. Geometry 3 (1988) 237–256.
[14] T. Akutsu, On determining the congruence of point sets in d dimensions, Comput. Geometry Theory Appl. 9 (1998) 247–256.
[15] P. Brass, C. Knauer, Testing the congruence of d-dimensional point sets, Int. J. Comput. Geometry Appl. 12 (2002) 115–124.

[16] P.J. Rezende, D.T. Lee, Point set pattern matching in *d*-dimensions, Algorithmica 13 (1995) 387–404.

[17] D.T. Lee, Y.T. Ching, The power of geometric duality revisited, Inform. Process. Lett. 21 (1985) 117–122.

[18] T. Akutsu, H. Tamaki, T. Tokuyama, Distribution of distances and triangles in a plane set and algorithms for computing the largest common point sets, Discrete Comput. Geometry 20 (1998) 307–331.

[19] P.K. Agarwal, B. Aronov, M. Sharir, S. Suri, Selecting distances in the plane, Algorithmica 9 (1993) 495–514.

[20] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, J. Assoc. Comput. Mach. 30 (1983) 852–865.

[21] R. van Oostrum, R.C. Veltkamp, Parametric search made practical, in: Proceedings of the 8th Annual Symposium on Computational Geometry, 2002, pp. 1–9.

[22] P.K. Agarwal, M. Sharir, Efficient algorithm for geometric optimization, ACM Comput. Surv. 30 (4) (1998) 412–458.

[23] J. Matousek, On enclosing *k* points by a circle, Inform. Process. Lett. 53 (1995) 217–221.

[24] E.M. Arkin, K. Kedem, J.S.B. Mitchell, J. Sprinzak, M. Werman, Matching points into pairwise-disjoint noise regions: combinatorial bounds and algorithms, ORSA J. Comput. 4 (1992) 375–386.

[25] L.P. Chew, M.T. Goodrich, D.P. Huttenlocher, K. Kedem, J.M. Kleinberg, D. Kravets, Geometric pattern matching under Euclidean motion, Comput. Geometry Theory Appl. 7 (1–2) (1997) 113–124.

[26] H. Alt, B. Behrends, J. Blömer, Approximate matching of polygonal shapes, in: Proceedings of the ACM Symposium on Computational Geometry, 1991, pp. 186–193.

[27] M.J. Atallah, A linear time algorithm for Hausdorff distance between convex polygons, Inform. Process. Lett. 17 (1983) 207–209.

[28] W.M. Wells, Statistical approaches to feature based object recognition, Int. J. Comput. Vision 21 (1997) 63–98.

[29] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography, Commun. Assoc. Comput. Mach 24 (1981) 381–395.

[30] J.R. Beveridge, C.R. Graves, J. Steinborn, Comparing random starts local search with key feature matching, in: Proceedings of the 15th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1997, pp. 1476–1481.

[31] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison-Wesley, Reading, MA, 1993.

[32] L.-H. Tung, I. King, P.-F. Fung, W.-S. Lee, Two-stage polygon representation for efficient shape retrieval in image databases, Proceedings of the First International Workshop on Image Databases and Multi-Media Search (IAPR '96), 1996, pp. 146–153.

[33] L. Szekely, Crossing numbers and Hard Erdös problems in discrete geometry, Combinatorics, Probab. Comput. 6 (1997) 353–358.

[34] L.G. Brown, A survey of image registration techniques, ACM Comput. Surv. 24 (4) (1992) 325–376.

[35] J.E. Goodman, R. Pollack, Multidimensional sorting, SIAM J. Comput. 12 (1983) 484–507.

[36] D. Maltoni, D. Maio, A.K. Jain, S. Prabhakar, Handbook of Fingerprint Recognition, Springer, New York, 2003.

[37] G.T. Candela, P.J. Grother, C.I. Watson, R.A. Wilkinson, C.L. Wilson, PCASYS—a pattern-level classification automation system for fingerprints, NISTIR 5647, National Institute of Standards and Technology, August 1995.

[38] C.I. Watson, Mated fingerprint card Pairs 2, Technical Report Special Database 14, MFCP2, National Institute of Standards and Technology, September 1993.

[39] A. Bishnu, P. Bhowmick, J. Dey, B. B. Bhattacharya, M.K. Kundu, C.A. Murthy, T. Acharya, Combinatorial classification of pixels for ridge extraction in a gray-scale fingerprint image, in: Proceedings of the third Indian Conference on Computer Vision, Graphics and Image Processing, Ahmedabad, India, December 2002, pp. 451–456.

[40] A. Farina, Zs.M. Kovacs-Vajna, A. Leone, Fingerprint minutiae extraction from skeletonized binary images, Pattern Recognition 32 (1999) 877–889.

[41] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, NY, 1980.

[42] K. Mehlhorn, S. Näher, LEDA: A Platform for Combinatorial and Geometric Computing, Cambridge University Press, Cambridge, UK, 1999.

[43] ⟨http://www.algorithmic-solutions.com/⟩.

[45] M. Maes, Polygonal shape recognition using string-matching techniques, Pattern Recognition 24 (1991) 433–440.

[46] G. Manacher, An application of pattern matching to a problem in geometrical complexity, Inform. Process. Lett. 53 (1995) 217–221.