Particle swarm optimization for point pattern matching

Peng-Yeng Yin *

Department of Information Management, National Chi Nan University, 303 University Road, Puli, Nantou 545, Taiwan, ROC

Received 24 May 2004; accepted 25 February 2005
Available online 30 March 2005

Abstract

The technique for point pattern matching (PPM) is essential to many image analysis and computer vision tasks. Given two point patterns, the PPM technique finds an optimal transformation for one point pattern such that a distance measure from the transformed point pattern to the other is minimized. This paper presents a new PPM algorithm based on particle swarm optimization (PSO). The set of transformation parameters is encoded as a real-valued vector called particle. A swarm of particles are initiated at random and fly through the transformation space for targeting the optimal transformation. The proposed algorithm is validated through both synthetic datasets and real fingerprint images. The experimental results manifest that the PSO-based method is robust against practical scenarios such as positional perturbations, contaminations, and drop-outs from the point sets. The PSO algorithm is also shown to be superior to a genetic algorithm and a simulated annealing algorithm on both effectiveness and efficiency.

© 2005 Elsevier Inc. All rights reserved.

Keywords: Point pattern matching; Particle swarm optimization; Genetic algorithm; Simulated annealing; Local optimal solution; Global optimal solution

* Fax: +886 49 2915205.
E-mail address: pyyin@ncnu.edu.tw.
1. Introduction

Point pattern matching (PPM) has found a variety of applications, such as image registration, stereo vision, and object recognition. When two images are matched, people prefer to deal with sets of local features (for example, dominant points) instead of pixel arrays to increase the accuracy and save the computation time. This relies on a PPM algorithm that matches two point patterns by finding an optimal transformation such that a distance measure for the alignment of the two point patterns under this transformation is minimized. According to the computational complexity, PPM can be characterized in two dimensions as shown in Fig. 1. The first dimension indicates whether the point matching is complete or incomplete. Complete matching requires that the two point patterns have the same number of points and there exists a one-to-one correspondence mapping between the two point sets. While in incomplete matching, no such one-to-one correspondence mapping is constructed due to the missing and spurious points. Instead, incomplete matching seeks for a mapping between subsets of the point patterns. The second dimension classifies the problems into labeled and unlabeled matching. Labeled matching is conducted using the a priori information (e.g., point order, intensity, gradient, etc.) as well as the point coordinates. While unlabeled matching determines the point correspondences based on the coordinates of the data points only. Clearly, incomplete matching is harder than complete matching and unlabeled matching is more complex than labeled matching. As such, the PPM problems fall into four categories (from Type I to Type IV) as shown in Fig. 1 where Type I problems are the simplest, Type II and Type III problems are harder, and Type IV problems are the most difficult.

In this paper we confine our attention to Type IV PPM problems with the affine transformation consisting of rotation, scaling, and translation. Let \( A = \{a_i \mid i = 1, 2, \ldots, n\} \) and \( B = \{b_i \mid i = 1, 2, \ldots, m\} \) be two point patterns in \( \mathbb{R}^2 \) and they are affinely dependent under a transformation \( T = (\theta, s, t_x, t_y) \) where \( \theta \) denotes the rotation angle, \( s \) is the scale factor, and \( t_x \) and \( t_y \) are the translation offsets along the \( x \)- and

![Fig. 1. Complexity of point pattern matching problems.](image-url)
$y$-axis directions. Also let $(a_i, b_j)$ be one of the point-correspondence pairs under $T$, and denote by $a_i = [x_{a_i}, y_{a_i}]^T$ and $b_j = [x_{b_j}, y_{b_j}]^T$ the corresponding coordinates, the following affine relation holds.

$$T(a_i) = s \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_{a_i} \\ y_{a_i} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_{b_j} \\ y_{b_j} \end{bmatrix} = b_j,$$  \tag{1}$$

When dealing with incomplete unlabeled PPM problems, some difficulties arise in realistic situations. (1) Since there is no the a priori information for unlabeled matching, the point-correspondence relation should be determined, in general, in an exhaustive way. (2) Since point patterns $A$ and $B$ appear with missing and spurious points and a one-to-one correspondence relation no longer exists ($n \neq m$), an appropriate distance measure for incomplete matching is needed. (3) The point pattern is distorted due to the data acquisition process such as two fingerprints of the same finger but taken at different time, resulting in a local perturbation on point coordinates. Hence, when matching two point patterns, a distance measure should be devised to accommodate these phenomena.

Two distance measures are usually applied for these purposes.

- **Agrawal’s heuristic dissimilarity (AHD) measure.** Let $\Omega$ be the set of point correspondences from both patterns with $|\Omega| \leq \min(n, m)$. The alignment error of $\Omega$ with respect to transformation $T$ can be evaluated using the integral squared error defined as

$$e^2 = \sum_{(a_i, b_j) \in \Omega} \|T(a_i) - b_j\|^2,$$  \tag{2}$$

where $\|\|_2$ is the vector length defined in the Euclidean space. To define a measure of the overall match between both patterns, Agrawal et al. [1] proposed a heuristic dissimilarity measure as

$$\text{AHD}(A, B) = \left\{ \begin{array}{ll} \frac{e^2}{|\Omega|} \left( 1 + \left( \frac{m-2}{|\Omega|-2} \right) \log_2 \left( \frac{m-2}{|\Omega|-2} \right) \right) & |\Omega| \geq 3, \\
\infty & |\Omega| = 0, 1, 2. \end{array} \right.$$  \tag{3}$$

This measure is normalized with the scale factor $s$ and penalizes for an incomplete match.

- **Partial Hausdorff distance (PHD).** Instead of taking the sum of squared errors as used in Eq. (2), Huttenlocher et al. [2] used the directed partial Hausdorff distance from $A$ to $B$ as

$$\text{DPHD}_d(A, B) = K^\text{th}_{a_i \in A} \min_{b_j \in B} \|T(a_i) - b_j\|,$$  \tag{4}$$

where $K^\text{th}$ returns the $k$th smallest value of $\min_{b_j \in B} \|T(a_i) - b_j\|$ for all $a_i \in A$. The directed partial Hausdorff distance from $B$ to $A$ can be analogously defined as

$$\text{DPHD}_d(B, A) = K^\text{th}_{b_j \in B} \min_{a_i \in A} \|b_j - T(a_i)\|.$$  \tag{5}$$
Finally, the partial Hausdorff distance from both patterns is given by

$$\text{PHD}_k(A, B) = \max(\text{DPHD}_k(A, B), \text{DPHD}_k(B, A)).$$

(6)

Many approaches have been proposed for various PPM problems. We summarize them in Table 1 and give brief descriptions as follows:

- **Clustering methods.** These techniques calculate the transformation parameters $\theta$, $s$, $t_x$, and $t_y$ for all combinations of point pairs from both patterns and increase the merit of the corresponding cell ($\theta, s, t_x, t_y$) in an accumulator matrix. The clusters in the accumulator matrix are then detected, the cluster with the maximum merit corresponds to the optimal parameters [3–7]. The clustering methods are computationally intensive due to the large number of combinations of point pairs and the dimensionality of the parameter space.

- **Parameter decomposition methods.** These algorithms divide the parameter estimation process into multiple phases. At the first phase, a selected parameter is estimated based on the domain knowledge such as the geometric invariant constraints. Then, at each of the following phases, one or more of the remaining parameters are estimated by referring to those parameters values previously determined, hence, the number of possible combinations between values of separate parameters is greatly reduced [2,8,9]. However, the inaccuracy level of parameter estimation could be increased by propagation through various phases.

- **Relaxation methods.** These techniques iteratively update merit score of a point mapping $(a_i, b_j)$ from both patterns given the merit scores of the other interacting point mappings. The interacting point mappings are those that are mutually constrained for matching. The algorithm converges when those merit values become consistent (or hardly changed) and the point mappings with the maximum merits are considered as the true transformation point correspondence [10–12].

- **Bounded alignment.** Mount et al. [13] proposed a geometric branch-and-bound search of the transformation space and used the point alignment information to bound the search. They specify an approximation factor to guarantee the accuracy of the final match and use point alignments when a significant number of point correspondences can be inferred to accelerate the search. The robustness of the algorithm has been demonstrated on registration of real satellite images.

- **Spectral graph analysis.** Carcassoni and Hancock [14] applied the spectral graph theory to compute the point pattern correspondence. The global structural properties of the point pattern are ascribed by the eigenvalues and eigenvectors of the proximity weighting matrix. The influence of the contamination and drop-out in the point pattern is discounted via the EM algorithm so the accuracy of the matching is increased.

- **Genetic algorithms.** The search space of point mappings between two patterns can be explored by genetic algorithms. The chromosomes encoding instances of point correspondences improve their fitness during evolution by use of three genetic
Table 1
Categories of existing approaches for the PPM problems

<table>
<thead>
<tr>
<th>Name of the Author(s) and Year</th>
<th>Clustering</th>
<th>Parameter decomposition</th>
<th>Relaxation</th>
<th>Bounded alignment</th>
<th>Spectral graph analysis</th>
<th>Genetic algorithms</th>
<th>Simulated annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goshtasby and Stockman, 1985</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Umeyama, 1991</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yuen, 1993</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wang and Chen, 1997</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chang et al., 1997</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Griffin and Alexopoulos, 1991</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Huttenlocher et al., 1993</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Olson and Huttenlocher, 1997</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ranade, 1980</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ogawa, 1984</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>Ton and Jain, 1989</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mount et al., 1998</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carcassoni and Hancock, 2003</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>Ansari et al., 1990</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Agrawal et al., 1994</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Zhang et al., 2003</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>Ansari et al., 1993</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Starink and Backer, 1995</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
</tr>
</tbody>
</table>
operators: selection, crossover, and mutation. The fitness of the chromosome is defined as the inverse of the AHD function \[1,15\]. Zhang et al. [16] use the reference triplet points as the chromosome representation and thus significantly reduce the search space.

- **Simulated annealing.** In [17,18] simulated annealing technique is applied to the point pattern matching problems. The identification of point correspondences between two point sets is mathematically formulized as energy minimization. The matching error corresponding to the current configuration of point correspondences is treated as the energy of that configuration. The configuration is iteratively rearranged to reach thermal equilibrium at various temperature levels and finally converges to an optimum as the system is frozen.

In this paper, we propose a particle swarm optimization (PSO) algorithm for the PPM problem. PSO is a new meta-heuristic proposed in 1995 [19] and has exhibited success in many applications including evolving weights and structure for artificial neural networks [20], manufacture end milling [21], reactive power and voltage control [22], state estimation for electric power distribution systems [23], and curve segmentation [24]. The convergence and parameterization aspects of the PSO have also been discussed [25–27].

The remainder of this paper is organized as follows. Section 2 reviews the general principles of PSO and renders the details of the proposed method. In Section 3, we present the experimental results. Finally, a summary is given in Section 4.

2. The proposed method

In this section, we first give a brief review of PSO and outline its general principles. Then the specific features of the proposed method for solving the PPM problem are presented.

2.1. Brief review of PSO

The first PSO algorithm was proposed by Kennedy and Eberhart [19] in 1995. It is a biologically inspired algorithm which models the social dynamics of bird flocking. A large number of birds flock synchronously, change direction suddenly, scatter and regroup iteratively, and finally perch on a target. This form of social intelligence not only increases the success rate for food foraging but also expedites the process. The PSO algorithm facilitates simple rules simulating bird flocking and serves as an optimizer for continuous nonlinear functions. The attractiveness of the PSO algorithm is due to the features: natural metaphor, stochastic move, adaptivity, and positive feedback.
The general principles of the PSO algorithm are outlined as follows:

- **Particle representation.** The particle in the PSO is a candidate solution to the underlying problem and move iteratively about the solution space. The particle is represented as a real-valued vector containing an instance of all parameters that characterize the optimization problem. We denote the \( i \)th particle by \( P_i = (P_{i1}, P_{i2}, \ldots, P_{id})^T \in \mathbb{R}^d \) where \( d \) is the number of parameters.

- **Swarm.** The PSO explores the solution space by flying a number of particles, called swarm. The initial swarm is generated at random and the size of swarm is usually kept constant through iterations. At each iteration, the swarm of particles search for target optimal solution by referring to previous experiences.

- **Personal best experience and swarm’s best experience.** The PSO enriches the swarm intelligence by storing the best positions visited so far by every particle. In particular, particle \( i \) remembers the best position among those it has visited, referred to as \( p_{best_i} \), and the best position by its neighbors. There are two versions for keeping the neighbors’ best position, namely \( l_{best} \) and \( g_{best} \). In the local version, each particle keeps track of the best position \( l_{best} \) attained by its local neighboring particles. For the global version, the best position \( g_{best} \) is determined by any particles in the entire swarm. Hence, the \( g_{best} \) model is a special case of the \( l_{best} \) model. It has been shown that the local version is often better, particularly the one using random topology neighborhood where each particle generates \( L \) links at random after each iteration if there has been no improvement, i.e., if the best solution seen so far by the swarm is still the same. In our implementation, we set \( L = 10 \).

- **Particle movement.** The PSO is an iterative algorithm according to which a swarm of particles fly about the solution space until the stopping criterion is satisfied. At each iteration, particle \( i \) adjusts its velocity \( v_{ij} \) and position \( p_{ij} \) through each dimension \( j \) by referring to, with random multipliers, the personal best position \( (p_{best_{ij}}) \) and the swarm’s best position \( (l_{best_{ij}}, \text{if the local version is adopted}) \) using Eqs. (7) and (8) as follows:

  \[
  v_{ij} = v_{ij} + c_1 r_1 (p_{best_{ij}} - p_{ij}) + c_2 r_2 (l_{best_{ij}} - p_{ij}) \tag{7}
  \]

  and

  \[
  p_{ij} = p_{ij} + v_{ij}, \tag{8}
  \]

  where \( c_1 \) and \( c_2 \) are the cognitive coefficients and \( r_1 \) and \( r_2 \) are random real numbers drawn from \( U(0, 1) \). Thus, the particle flies toward \( p_{best} \) and \( l_{best} \) in a navigated way while still exploring new areas by the stochastic mechanism to escape from local optima. Clerc and Kennedy [26] have pointed out that the use of a constriction factor is needed to insure convergence of the algorithm by replacing Eq. (7) with the following:

  \[
  v_{ij} = K[v_{ij} + c_1 r_1 (p_{best_{ij}} - p_{ij}) + c_2 r_2 (l_{best_{ij}} - p_{ij})] \tag{9}
  \]

  and

  \[
  K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \tag{10}
  \]
where $\varphi = c_1 + c_2$, $\varphi > 4$. Typically, $\varphi$ is set to 4.1 and $K$ is thus 0.729.

- **Stopping criterion.** The PSO algorithm is terminated with a maximal number of iterations or the best particle position of the entire swarm cannot be improved further after a sufficiently large number of iterations.

### 2.2. PSO for PPM

To apply PSO for solving the PPM problem, we add the following features.

**Particle coding scheme.** We propose to code the affine transformation parameters, namely the rotation angle $\theta$, the scale factor $s$, and the translation offsets $t_x$ and $t_y$ in the particle representation, viz., the particle vector would look like

$$ P = (\theta, s, t_x, t_y)^T, \quad (11) $$

where each parameter value is a random real number and is restricted by an appropriate range. In particular, $0^\circ \leq \theta \leq 360^\circ$, $0 < s \leq 10$, and $-200 \leq t_x, t_y \leq 200$ are appropriate for a large number of applications. As such, each particle encoded in this way corresponds to one set of affine transformation parameters to align the point patterns. During evolution, the particles are constrained to move in the same ranges as they are initialized. When the particles reach the boundary constraints, they are set to boundary values.

**Fitness evaluation and bounding criterion.** In PSO, the solution quality, or fitness, delivered by each particle is evaluated. The two alternative matching distance measures, AHD and PHD as introduced in Section 1, can be used for this purpose. Since these measures are error functions, a particle delivering smaller AHD or PHD value is considered to be superior to the other particles with larger AHD or PHD values. As such, the $p_{best}$ and $l_{best}$ can be determined according to the fitness values of all particles.

Here, we propose a bounding criterion to speedup the computation for determining $p_{best}$ and $l_{best}$. The formulae of particle movement, as shown in Eq. (9), refer to representations of $p_{best}$, $l_{best}$, and the particle itself, not directly to their fitness values [28]. We propose to use this property for saving computation time. Since the fitness value of a particle is only used for updating of $p_{best}$ and $l_{best}$, we can use the fitness value of the incumbent $p_{best}$ as an error upper bound to terminate the fitness computation of particle $i$. More precisely, the computation of AHD involves an error summation over point matching (see Eqs. (2) and (3)) and the computation of PHD is also resulted from the maximum of two sub-error measures (see Eq. (6)), both of which are a value-increasing computation. Hence, we can terminate the error computation for particle $i$ once the intermediate error value exceeds the fitness value of the incumbent $p_{best}$, and go directly to the fitness evaluation of the next particle. Also, only those $p_{best}$ that have been updated at the current iteration need to be compared to associated $l_{best}$ for its possible updating. The use of bounding criterion can save the computational time significantly.
The proposed algorithm is summarized in Fig. 2. Initially, a swarm of particles are created at random and each of which is a vector corresponding to an instance of transformation parameters to the underlying problem. Then, the particle movement is repeated until a maximal number of iterations have been passed. During each iteration, the particle individual best and swarm's best positions are determined using the bounding criterion. The particle adjusts its position based on the individual experience ($p_{best_i}$) and the swarm intelligence ($l_{best_i}$). When the algorithm is terminated, the best experience among all $p_{best}$ and the corresponding fitness value are output and considered as the optimal transformation parameters and the alignment error.

Fig. 2. PSO algorithm for the PPM problem.

3. Experimental results

In this section, we present the experimental results and analyze the computational performance. The performance of the proposed algorithm is analyzed on both synthetic point patterns and real images. The superiority of the method is also demonstrated by comparing to a genetic algorithm (GA) and a simulated annealing (SA) algorithm on several testing scenarios. The platform of the experiments is a PC with a 1.8 GHz CPU and 192 MB RAM. All programs are coded in C++ language.

3.1. Synthetic point patterns

We first validate the correctness of the proposed algorithm by synthetic point patterns, and then extensive experiments are conducted to analyze the comparative performances of PSO, GA, and SA. Fig. 3A shows a point set, referred to as pattern $A$, consisting of 250 points generated at random. We set up four scenarios to generate testing patterns to be matched with pattern $A$. 
Scenario RST: A testing pattern is generated by applying to pattern \( A \) with an affine transformation consisting of rotation, scaling, and translation. In particular, the transformation parameters \((h, s, tx, ty)\) are set to \((30/176, 1.5, 19.0, 42.0)\). The resulting testing pattern, referred to as pattern \( B \), is shown in Fig. 3B. It can be formulated by

- Scenario RST: A testing pattern is generated by applying to pattern \( A \) with an affine transformation consisting of rotation, scaling, and translation. In particular, the transformation parameters \((\theta, s, t_x, t_y)\) are set to \((30^\circ, 1.5, 19.0, 42.0)\). The resulting testing pattern, referred to as pattern \( B \), is shown in Fig. 3B. It can be formulated by
Pattern $B = T[\text{Pattern } A]$. 

- **Scenario RSTP**: In addition to applying Scenario RST, a random perturbation quantity is added to the coordinates of each data point. The random perturbation quantity is generated uniformly within 1% of the maximum point span along every coordinate axis dimension and the resulting testing pattern is shown in Fig. 3C. The formulation is as follows:
  
  $\text{Pattern } B = T[\text{Pattern } A] + \text{perturbation}$.

- **Scenario RSTPA**: Besides applying Scenario RSTP, we augment the size of the resulting point set by 20% by adding 50 spurious random points and thus yielding a 300-point pattern (see Fig. 3D), viz.,

  $\text{Pattern } B = T[\text{Pattern } A] + \text{perturbation} + 20\% \text{ points}$.

Fig. 4. Matching results obtained using the proposed method for all testing scenarios. (A) Scenario RST with AHD measure, (B) Scenario RST with PHD measure, (C) Scenario RSTP with AHD measure, (D) Scenario RSTP with PHD measure, (E) Scenario RSTPA with AHD measure, (F) Scenario RSTPA with PHD measure, (G) Scenario RSTDA with AHD measure, and (H) Scenario RSTDA with PHD measure.
Scenario RSTDA: First, Scenario RST is applied to pattern A. Then, randomly select 20% of the points and remove them. Generate the same number of spurious points and add them to the point set (see Fig. 3E). The formulation is as follows:

$$\text{Pattern } B = T[\text{Pattern A}] - 20\% \text{ points} + 20\% \text{ points}.$$ 

Fig. 4 shows the matching results obtained using the proposed method from pattern A and pattern B of all testing scenarios. For each testing scenario, the proposed method is performed with AHD and PHD distance measures, respectively. For the Scenario RST (see Figs. 4A and B) and the Scenario RSTP (see Figs. 4C and D), the PSO can find the complete matching from both patterns and the one-to-one correspondence relationship is correctly identified. For the Scenario RSTPA (see Figs. 4E and F) and the Scenario RSTDA (see Figs. 4G and H), the PSO derives the incomplete matching. The point patterns are appropriately aligned for both scenarios and the groundtruth point matching correspondences are found. Note that in all testing scenarios both AHD and PHD work well with the proposed method.
To illustrate the advantages of the proposed method, we also implement algorithms using GA and SA for performance comparison. Analogous to the PSO algorithm, GA is also a population-based search algorithm which explores the search space using a number of individual agents, called chromosomes. We implement the GA with the same coding scheme, initialization ranges, and fitness evaluation as used by our algorithm. In addition to the broadly used genetic operations, namely the selection, crossover, and mutation, we further employ fitness scaling and elitist strategy [29] to enhance the performance of the GA. On the other hand, SA is a search meta-heuristic based on perturbation of a single candidate solution simulating the thermal annealing process. The implemented SA also uses the same coding scheme, initialization ranges, and fitness function as used by the other algorithms.

Following the four testing scenarios previously described, we generate extensive testing datasets with different numbers of points, in particular, point patterns having 50, 250, and 500 points are used. For a fair comparison, all competing algorithms are terminated when they have consumed 4000 times of fitness evaluations. In all experiments, PSO is conducted with 20 particles and is allowed to experience 200 iterations, GA is conducted with 20 chromosomes and is performed with 200 generations. While SA is a single agent search algorithm, we let SA execute with 4000 iterations. These algorithms will be named by PSO/20/200, GA/20/200, and SA/1/4000 representing the experimental setting algorithm/number_of_agents/number_of_iterations. The numerical results for each dataset are the mean value and the standard deviation (σ) from 30 independent runs and they are summarized in Tables 2 and 3 (the CPU times are evaluated in seconds) for AHD and PHD measures, respectively. We have the following observations. (1) PSO/20/200 has the best performance (for both of distance measures and computation time), GA/20/200 is the worst, and SA/1/4000 produces the intermediate results. (2) The results obtained using PSO/20/200 are more stable (with smaller values of σ) than those obtained using GA/20/200 and SA/1/4000. (3) For all RST testing datasets, PSO/20/200 does not yield any matching errors for both AHD and PHD measures because RST incurs a complete matching with no perturbation, while both GA/20/200 and SA/1/4000 fail to find the optimal transformation. (4) For all RSTDA testing datasets, PSO/20/200 produces no matching errors for the PHD measure, which means the PSO/20/200 algorithm is able to find the groundtruth transformation for incomplete matching without perturbation (PSO/20/200 does not generate zero error with AHD measure because AHD penalizes an incomplete matching, however, the point correspondences are still correctly identified). (5) On the average, PSO/20/200 consumes about 80% of the CPU time required by GA/20/200 and about 91% of the CPU time required by SA/1/4000. This is because the bounding criterion (see Section 2.2) we adopted can speed up the optimization process of PSO.

To demonstrate the computation saving time at each iteration, we compute the time reduction ratio between the saving time with the bounding criterion and the computation time without using this property. Fig. 5 shows the variation of the time reduction ratio due to the bounding criterion for a particular run. We observe that the time reduction ratio initially ranges from 30 to 50% which indicates a significant saving. After the 60th iteration, the time reduction ratio starts decreasing and finally
### Table 2
Comparative performances of the PSO, GA, and SA algorithms with respect to the AHD measure and the used computation time (in seconds)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>R</th>
<th>S</th>
<th>T</th>
<th>P</th>
<th>D</th>
<th>A</th>
<th>PSO/20/200</th>
<th>GA/20/200</th>
<th>SA/1/4000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AHD (σ)</td>
<td>Time (σ)</td>
<td>AHD (σ)</td>
</tr>
<tr>
<td>PPM50</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>0.0 (0.0)</td>
<td>8.8 (0.1)</td>
<td>9.2 (0.5)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>11.2 (0.0)</td>
<td>8.6 (0.1)</td>
<td>14.3 (1.1)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td>14.1 (0.0)</td>
<td>10.3 (0.1)</td>
<td>14.9 (1.2)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>204.2 (0.0)</td>
<td>8.7 (0.1)</td>
<td>211.4 (6.2)</td>
</tr>
<tr>
<td>PPM250</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>0.0 (0.0)</td>
<td>206.4 (3.0)</td>
<td>9.5 (0.7)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>10.8 (0.0)</td>
<td>206.4 (1.6)</td>
<td>40.9 (2.4)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>13.0 (0.0)</td>
<td>245.4 (0.2)</td>
<td>29.5 (1.7)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>112.3 (0.0)</td>
<td>204.9 (0.2)</td>
<td>123.9 (3.6)</td>
</tr>
<tr>
<td>PPM500</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>0.0 (0.0)</td>
<td>821.2 (2.1)</td>
<td>15.1 (1.3)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>11.4 (0.0)</td>
<td>817.7 (1.8)</td>
<td>15.9 (1.1)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>55.3 (0.0)</td>
<td>979.3 (2.4)</td>
<td>72.6 (3.4)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>53.9 (0.0)</td>
<td>816.6 (2.0)</td>
<td>62.8 (3.0)</td>
</tr>
</tbody>
</table>

### Table 3
Comparative performances of the PSO, GA, and SA algorithms with respect to the PHD measure and the used computation time (in seconds)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>R</th>
<th>S</th>
<th>T</th>
<th>P</th>
<th>D</th>
<th>A</th>
<th>PSO/20/200</th>
<th>GA/20/200</th>
<th>SA/1/4000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PHD (σ)</td>
<td>Time (σ)</td>
<td>PHD (σ)</td>
</tr>
<tr>
<td>PPM50</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>0.0 (0.0)</td>
<td>12.1 (0.2)</td>
<td>20.4 (1.4)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td>16.6 (0.1)</td>
<td>11.8 (0.1)</td>
<td>24.5 (1.6)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>22.4 (0.6)</td>
<td>14.1 (0.1)</td>
<td>36.4 (2.2)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>0.0 (0.0)</td>
<td>11.8 (0.1)</td>
<td>29.5 (1.9)</td>
</tr>
<tr>
<td>PPM250</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>0.0 (0.0)</td>
<td>288.3 (1.8)</td>
<td>149.7 (4.8)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>19.8 (0.1)</td>
<td>289.3 (1.7)</td>
<td>37.2 (2.1)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>24.1 (1.5)</td>
<td>341.6 (1.9)</td>
<td>168.9 (6.1)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>0.0 (0.0)</td>
<td>283.8 (0.2)</td>
<td>116.8 (3.9)</td>
</tr>
<tr>
<td>PPM500</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>0.0 (0.0)</td>
<td>1148.8 (2.2)</td>
<td>9.7 (0.7)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>18.4 (0.1)</td>
<td>1147.3 (2.7)</td>
<td>24.5 (1.2)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>53.7 (2.1)</td>
<td>1385.7 (8.0)</td>
<td>70.4 (4.5)</td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>0.0 (0.0)</td>
<td>1130.9 (2.7)</td>
<td>76.2 (3.8)</td>
</tr>
</tbody>
</table>

156
stays around 10%. This is because the particle quality is far from good at the initial stage, a substantial amount of the computation for particle fitness is likely to be saved due to the bounding criterion. As the number of iterations increases, the

Fig. 5. Time reduction ratio vs. number of iterations.

Fig. 6. Matching error vs. number of iterations. (A) Average AHD value obtained from all pbest of the swarm, (B) AHD value obtained by the swarm’s best experience, (C) average PHD value obtained from all pbest of the swarm, and (D) PHD value obtained by the swarm’s best experience.
quality of particles improves, the amount of computation that can be saved becomes limited.

3.2. Experience convergence

The quality solution delivered by PSO is established on the fundamentals of managing the knowledge derived from the personal best experience (pbest) and the swarm’s best experience. The pbest of each particle is a form of distributed awareness which exploits a local region and reports the historic fitness, while the swarm’s best experience is the collective intelligence that is generated through communication between individual particles and is used as a feedback to avoid pbest getting trap in a local optimum.

Fig. 7. Two different fingerprint images of the same finger. (A) Fingerprint image A and (B) fingerprint image B.
Fig. 6 illustrates the convergence behavior of the personal best and the swarm’s best experiences using the AHD and PHD error measures. The average AHD value obtained by all $p_{best}$ decreases as the number of iterations increases (see Fig. 6A), these distributed awareness carried by all particles are collected to establish the

Fig. 8. Distributions of the minutiae points. (A) Minutiae point pattern $A$ and (B) minutiae point pattern $B$. 
swarm’s best experience (see Fig. 6B) which represents the best solution ever seen so far by the entire swarm. The swarm’s best experience is used to navigate the move of every particle such that the individual particle will not get trapped in a local optimum and keep exploring new regions. This property is observed from Figs. 6A and B that the average AHD value obtained by pbest does not get stuck before

Fig. 9. Alignment of the two minutiae point patterns. (A) Result obtained by PSO/20/200 and (B) result obtained by GA/20/200.
swarm’s best experience converges. Figs. 6C and D correspond to the convergence curves using the PHD measure. They behave similarly as in the AHD case.

3.3. Fingerprint images

Fingerprint matching is an important process in forensic applications due to its uniqueness across individuals. Most automatic fingerprint identification systems match fingerprint images based on the ridge minutiae points [30]. Minutiae points consist of ridge bifurcations and ridge endings. The proposed algorithm can be used to determine fingerprint identification by aligning two fingerprint minutiae point patterns. Figs. 7A and B show two fingerprint images taken from the same finger at different time. The distributions of the minutiae points are shown in Figs. 8A and B and they are referred to as minutiae point patterns A and B, respectively. PSO/20/200 is applied to search the optimal transformation from both patterns and determines \((h, s, t_x, t_y) = (-2.165^\circ, 0.98, -68.47, -22.14)\), while GA/20/200 gives the parameter values as \((h, s, t_x, t_y) = (-9.017^\circ, 0.93, -67.73, -5.48)\). Figs. 9A and B show the alignment from both minutiae point patterns using PSO/20/200 and GA/20/200. It is seen that PSO/20/200 outperforms GA/20/200 in producing better alignment.

4. Conclusions

In this paper, we have proposed a PSO-based algorithm for incomplete unlabeled PPM problems. The advantage of the proposed algorithm is due to the natural metaphor, stochastic move, adaptivity, and positive feedback. The proposed method has been shown to be robust against real-world testing scenarios. It is also more effective and efficient than the algorithms based on GA and SA in minimizing the matching error from two given point sets. Both synthetic datasets and real fingerprint images have been tested to illustrate the superiority of the proposed method.

References

[21] V. Tandon, Closing the gap between CAD/CAM and optimized CNC end milling, Master thesis, Purdue School of Engineering and Technology, Indiana University, Purdue University, Indianapolis, 2000.