Genetic Search for Optimally-Constrained Multiple-line Fitting of Discrete Data Points

Xuefei Shi¹ Jie Shen^{2,3} David Yoon³

¹School of Automation and Electrical Engineering, Beijing University of Science & Technology, Beijing 100083, P.R. China, sxf1245@ies.ustb.edu.cn

²College of Automotive Engineering, Shanghai University of Engineering Science, Shanghai, China

³Department of Computer and Information Science, University of Michigan-Dearborn, Dearborn, MI 48128, USA, shen@umich.edu

Abstract:

Advanced sensing technologies have produced a significant amount of discrete point data in the past decade. Measurement uncertainty frequently occurs at the geometric discontinuity of mechanical parts. In this paper, a genetic search algorithm is developed for optimally-constrained multiple-line fitting of discrete data points. It contains two important technical components: a) constrained least-squares fitting of multiple lines, and b) genetic search for optimal corner/edge points. The algorithm is designed for both two-dimensional and three-dimensional cases. Numerical experiments demonstrate the effectiveness of the proposed approach, compared to the conventional least-squares fitting method as well as exhaustive search method. A comparative study with a particle swarm method indicates that both the genetic search and particle swarm search produce similar results in terms of minimum fitting errors. It can be used for the effective determination of sharp edges or corners based on discrete data points measured for high-precision industrial inspection and manufacturing.

Keywords: geometric discontinuity, genetic algorithm, particle swarm, constrained fitting of multiple lines, discrete data point, optimization.

Nomenclature:

The symbols used in this paper are listed in Table 1 below.

Table 1: Definition of mathematical symbols.

Symbol	Definition
a, b, c	Direction numbers of a line in a three-dimensional space
b_{c1}, b_{c2}, b_{c3}	Binary numbers for a chromosome
c_1, c_2	Cognitive and social parameters
d_{c1}, d_{c2}, d_{c3}	Decimal value corresponding to binary bits
d_i	The shortest distance of the i -th data point and the fitting line

$f(x, y)$	Fitness function in two-dimensional cases
$f(x, y, z)$	Fitness function in three-dimensional cases
$f_i, f_m, f_{\max}$	The fitness value of the i -th candidate corner, mean and maximum fitness
pc	The probability of crossover
pm	The probability of mutation
p_g	The position of the globally-best individual
p_i	The best previous position of the i -th particle
r_1, r_2	Random numbers that are uniformly distributed within a range [0,0.2]
$q_i^k = (x_i, y_i)$	The k -th position of the i -th particle in 2D
$q_i^k = (x_i, y_i, z_i)$	The k -th position of the i -th particle in 3D
x_i, y_i, z_i	The coordination of the i -th data point in a three-dimension space
x_{c0}, y_{c0}, z_{c0}	The coordination of start point for genetic search
x_i^p, y_i^p, z_i^p	Orthogonal projection of data points to fitted lines
$\Delta x, \Delta y, \Delta z$	The real increment in a given genetic search
x_c, y_c, z_c	The coordination of candidate constrained corner
x_s, y_s, z_s	The size of genetic search in three coordinate directions
x_0, y_0, z_0	The coordinates of the point passed by a line
x, y, z	The coordinates of any point in a three-dimension space
v_i^k	The k -th velocity of the i -th particle
w	Inertial weight
A, B, C, D	The coefficients of a line equation in two and three dimensions
$E(A, B),$	The sum of squared coordinate difference in x coordinate between data points and their corresponding points on the fitted line
$E(C, D),$	The sum of squared coordinate difference in y coordinate between data points and their corresponding points on the fitted line
$\nabla E(A, B)$	The gradient of $E(A, B)$
$\nabla E(C, D)$	The gradient of $E(C, D)$
σ	Standard deviation of data points

1. Introduction

Transformation of physical objects into digital models of three-dimensional (3D) point clouds represents a revolution in the 21st century together with 3D printing technology in manufacturing. Large-scale 3D discrete data points are generated by a variety of emerging sensor technologies, including laser scanning, LIDAR (light + radar), and digital range finder. Unfortunately, these loosely-connected point clouds lack structural information about physical objects, and the uncertainty of surface normal at sharp features of objects leads to unpredictable measurement results of the aforementioned sensor technologies. Moreover, a limited number of laser rays likely miss the exact location of corner or edge points. This leads to a significant amount of data errors around the sharp features, as in Fig. 1.

It is therefore a meaningful research how to handle such kind of data abnormality at geometric discontinuity. Since the geometric discontinuity can be complicated by the intersection of complex geometric shapes, this paper decomposes an entire problem into a number of relatively simple line intersection problems.

The problem of traditional piecewise regression of measurement data at G^1 discontinuity is the generation of gaps (i.e., G^0 discontinuity) at the location of G^1 discontinuity, as illustrated in Fig. 2. Here, G^1 discontinuity means the discontinuous change in direction of the tangent vectors of lines or curves, while G^0 discontinuity refers to the discontinuity of data points of lines, curves, and surfaces. The gap in Fig. 2 is an example of G^0 discontinuity, and is enlarged for the purpose of clarity. In the case of high-precision digital manufacturing, an opening of several hundreds of micrometers would be problematic. On the other hand, it is not important in entertainment industry for the sole purpose of modeling or rendering.

Figure 2: Opening of two regression line segments at the location of G^0 discontinuity.

Figure 3: Instances of G^1 discontinuity of a mechanical part.

G^1 discontinuity is the most prevailing geometric discontinuity among manufactured parts in various industries, as shown in Fig. 3 (sharp edge, corner, and crease). The direction of surface normal becomes undefined or ambiguous at this type of geometric discontinuity. In order to address the high-fidelity fitting at G^1 discontinuity for high-precision digital manufacturing and industrial inspection, the gap in Fig. 2 should be avoided at edge points, where two planes meet to form an edge, or corner points, where three or more planes converge to a common point.

Although piecewise fitting can give a very low fitting error of each fitted segment, it may cause the gap in Fig. 2. To avoid G^0 discontinuity (i.e., a gap), a constrained fitting at geometric discontinuity is needed. Unfortunately, this type of constrained fitting demands the information on the exact location of geometric discontinuity, which is not known a priori, given only a set of discrete data points. Thus, it is crucial to determine the location of corner/edge points before the execution of the constrained fitting.

2. Literature Review

2.1 Sharp features

Sharp corner and edge are the typical features of mechanical parts. A number of studies have been conducted to preserve sharp features during a mesh smoothing process. One technique was anisotropic diffusion that was used in height fields [1], triangle meshes [2-6], level set surfaces [7], surface and function [8], and surface reconstruction [9-11]. Another method was bilateral filtering [12, 13], which was extended to a trilateral filter [14]. Recently, several new methods were proposed for the denoising at geometric discontinuity [15-22]. The aforementioned methods were designed primarily for entertainment industry where geometric tolerance and small gap between piecewise fittings are not a concern. In essence, smoothing is an iterative process in which the sharp features are inevitably altered to a certain extent, causing unacceptable numerical errors in the field of high-precision manufacturing and industrial inspection. Yet, some other approaches [16] did not change data points at the geometric discontinuity, meaning no data rectification at all. Very little

work has been conducted with a focus on determining the location of geometric discontinuity for high-precision manufacturing and industrial inspection.

2.2 Line fitting

Past research on line fitting spanned over several different disciplines. In image processing and computer vision, line fitting is a process in which a fitted line is determined on the basis of the pixels of feature lines. Hough transform (HT) [23] is a typical line fitting method in which a special Radon transform is applied on all points in the image and then a search on the maxima is conducted in the line parameter plane. Subspace-based Line DEtection (SLIDE) is another approach [24], which uses a planar wavefront propagation on a sensor array to perform line detection. Iterative adaptive approach (IAA) [25] is a nonparametric spectral estimation method for determining the line parameters. However, the geometric discontinuity at the intersection of lines was not the focus.

In the field of statistics, research on the data fitting of a single line has been extensively studied in the past. Deming [26] proposed the minimization of the following sum:

$$\chi^2 = \sum_{i=1}^n \left(\frac{(x_i - x_i^p)^2}{u_{x,i}^2} + \frac{(y_i - y_i^p)^2}{u_{y,i}^2} \right), \quad (1)$$

where (x_i, y_i) and (x_i^p, y_i^p) refer to data points and their orthogonal projection on the fitted line, respectively. $(u_{x,i}, u_{y,i})$ is related to the uncertainties of data points. York [27, 28] later developed an exact analytic solution for minimizing χ^2 . When $u_{x,i} = u_{y,i} = \sigma$ (standard deviation of data points), Equation (1) is reduced to a total least-squares problem, which was applied in many studies [29-31]. Furthermore, the fitting of a straight line to data with errors in all the coordinate directions was also investigated extensively by using effective variance methods [32-34] or schemes dealing with the variances of the fitting parameters [35-37]. To this day, very little work has been done in investigating the data fitting problem at geometric discontinuity where two lines meet, and it is an important problem in high-precision industrial inspection of mechanical parts with sharp edges or corners.

2.3 Optimization

Various types of global numerical optimization methods have been developed in the past. Six major types are listed below.

(1) Simulated Annealing

Simulated annealing is a probabilistic metaheuristic search scheme for a good approximation of global optimization in a large solution space [38, 39]. It is analog to annealing in metallurgy where heating and cooling affect the crystal structure of materials. Slow cooling in the simulated annealing algorithm means a slow decrease in probability of accepting worse solutions for more extensive search in a solution

space. Monte Carlo method [40] is a key component to generate sample states for this scheme. Neighbors of a state, transition probabilities, acceptance probabilities, barrier avoidance, and cooling schedule are important issues that need to be considered.

(2) Genetic Algorithm

Evolutionary algorithm (EA) is a population-based optimization methodology that mimics the Darwinian concept of evolution [41]. An iterative process is used to evolve a working set of individuals (i.e., population) toward the optimization of the problem based on a fitness function (i.e., objective function) [42]. The key idea is to simulate the evolution of individual structures via processes of selection, mutation, and reproduction. Genetic algorithm (GA) [43] is a typical type of evolutionary algorithm.

(3) Evolution Strategies

Evolution strategies (ES) [44, 45] is also an evolution at the “individual” level with recombination (crossover) involved. Mutation and selection are its main search operators. The environmental selection is deterministic based purely on the fitness rankings (not fitness values). This leads to an algorithm that is invariant with the monotonic transformations of the objective function. In each iteration (i.e., generation), multiple parents and their mutants with a recombination operator can be used to determine the parent of next generation.

(4) Evolutionary Programming

Evolutionary programming (EP) [46, 47] is an evolution at the “species” level without recombination between different species. The structure of the program is fixed, while its numerical parameters are changeable during the evolution. Contemporary algorithms do not impose the fixed structure, leading to close similarity with evolution strategies.

(5) Swarm Intelligence

Swarm intelligence was developed for mimicking the swarm characteristics of biological species. Particle swarm optimization (PSO) is a typical metaheuristic method in which candidate solutions are represented by a swarm of particles that have certain locations and speeds in the solution space [48, 49]. In PSO, each individual (i.e., particle) utilizes two kinds of information in its evolutionary process. The first type is the particles’ own experience; the particles have memory and always keep track of their previous best position. The other is the intelligence of the whole swarm: each particle adjusts itself towards the global best position [50]. Note that the quality of initial particles can affect the speed of optimization. Normally, the size of initial population and its corresponding positions are randomly generated. There are different ways to initialize particles and velocities [51].

Other swarm intelligence algorithms include differential evolution [52, 53], bat algorithm [54], firefly algorithm [55], and cuckoo search [56]. These methods are promisingly efficient in solving multimodal, multi-objective, global optimization problems.

(6) Hybrid Approaches

Hybrid approaches were designed for handling with complexity, noisy environment, imprecision, uncertainty, and vagueness via the combination of evolutionary method and other optimization algorithms, machine learning techniques, heuristics, etc. One possible benefit of such a scheme is a more efficient convergence to high-quality solutions over conventional schemes. In a large number of previous studies, evolutionary algorithm was hybridized with neural network [57], fuzzy logic [58-60], particle swarm [61], colony optimization [62], bacterial foraging [63], tabu search [64], simulated annealing [65], hill climbing [66, 67], dynamic programming [68], greedy search [69], etc.

In this paper, we explore two typical evolutionary approaches (GA and PSO) to solve the multiple-line fitting problem at geometric discontinuity. The two methods work well on mixed (continuous and discrete) combinatorial problems. One major benefit of using the evolutionary approaches is no need for the gradient information of the objective function, which is not known in the problem of Section 1. Since these two methods generate sufficiently good solutions to the problem, hybrid methods are not further investigated.

2.4 Goal and Organization

In this paper, we propose an effective data rectification method for discrete points at geometric discontinuity. Since many mechanical parts consist of planar surfaces, our effort is limited to the cases of straight lines. The key ideas of this method include a) constrained least-squares fitting of multiple lines (especially, two lines), b) a simple and fast propagation method to determine the search space of geometric discontinuity, and c) genetic search for optimal corner points. The result of our approach is compared with the traditional least-square fitting method, an exhaustive search, and a particle swarm optimization. The rest of the paper is organized as follows. In Section 3, the concept of constrained fitting of multiple lines is introduced, and a specially-designed genetic search is given. Next, in Section 4 the experimental settings of our numerical analyses are described. Section 5 includes numerical results and discussions. Some conclusions on this study are drawn in Section 6.

3. Constrained Least-squares Fitting of Multiple Lines

As explained in Section 1, one main problem with algorithms in entertainment industry is that a gap may be introduced in the line fitting of two-dimensional or three-dimensional cases. This is detrimental to industrial inspection for high-fidelity manufacturing, where few hundred

micrometers are important to the quality of the inspection. For most mechanical parts, G^0 discontinuity (i.e., gap) is not existent if no cracks or material defects are considered. To avoid artificial G^0 discontinuity caused by piecewise numerical fitting, constrained fitting has to be conducted. Unfortunately, the location of corner points or edge points is not known before the fitting. Thus, the first important step for a constrained fitting is to determine the location of G^1 discontinuity, as illustrated in Fig. 4.

In general, the location of corner or edge points should be located in a high-curvature region. The curvature can be computed by a numerical estimate [16] based on principal component analysis. The drawback of principal component analysis is its high computational cost. In the cases of two-dimensional and three-dimensional lines, a simple line propagation scheme can be designed as follows. The basic idea is to start from one end point (point A in Fig. 4) and to propagate along one direction of a line. The propagation continues until the tangent direction is significantly changed (point C in Fig. 4). Similarly, a new round of propagation begins with another end point (point B in Fig. 4) and terminates at point D. Points C and D define an approximate region for the corner point where two lines are supposed to meet.

Figure 4: An approximate search space for corner location determined by line propagation.

Since discrete data may be quite noisy due to the nature of laser scanning, our numerical tests indicate that connection between two neighboring data points is not a reliable index for guiding the propagation process, because the computed tangent direction could vary dramatically with local noise. To be more resilient on the local variation, a finite interval method is proposed: the direction of line is estimated by connecting two data points at a finite interval. For instance, points 1 and 5 form the first reference line. If points 3 and 7 are connected as the i -th line, and if points 5 and 9 form the $(i+1)$ -th line, point 7 is considered as a candidate for corner point only when the directions of the $(i+1)$ -th line becomes inconsistent with the direction of the reference line. The inconsistency is judged by an angular threshold between the two directions. Our test results indicate that the comparison of the directions between the i -th and $(i+1)$ -th lines does not generate reliable results for locating the corner point because it is easily influenced by the local noise.

In this paper, the exact location of the corner point is searched by a specially-designed genetic algorithm. Its details are given below for both two and three dimensional cases. As a reference of fair comparison, one approach (Scheme 1: Traditional Least-Squares Fitting) is designed to select the best data point as the corner for constrained fitting. In this scheme, approximate corner space is first determined by propagation (Fig. 4). For every potential constrained corner that is one of discrete data points, its corresponding fitting error is calculated on the basis of least-squares fitting. The point that has the minimum fitting error is selected as the best constrained corner. The flowchart of this scheme is demonstrated in Fig. 5. Since there are a limited number of data points in a space near the intersection of two/multiple lines, this scheme can achieve only limited optimization at the geometric discontinuity.

Figure 5: Flowchart of Scheme 1 (Traditional least-squares fitting).

3.1 Two-dimensional Genetic Search

Binary coding is applied to our first step. The (x, y) coordinates of each two-dimensional data point is represented by an n -bit binary number, i.e., a chromosome. For example, in this paper we set $n=20$. The first ten digits of the binary number, b_{c_1} , refer to x coordinate, while the second ten digits, b_{c_2} , are used to represent the y coordinate of the point. The initial population of genetic search is a set of binary numbers that are produced by a random number generator. The first generation of corner candidates is constructed by the following equation:

$$x_c = x_{c0} + \Delta x, y_c = y_{c0} + \Delta y, \quad (2)$$

where (x_{c0}, y_{c0}) is a given point as a starting point for our genetic search. Points C and D approximately define a region for the corner search. The exact boundary might be assigned by point (x_{c0}, y_{c0}) and search size (x_s, y_s) , as illustrated in Fig. 6. Δx and Δy are random decimal increments with respect to the start point (x_{c0}, y_{c0}) . To maximize the coverage of binary bits, binary numbers are used to represent Δx and Δy instead of x and y . Furthermore, a scaling operation based on the search size (x_s, y_s) is devised as

$$\Delta x = \frac{d_{c_1} \times x_s}{2^{n/2} - 1}, \Delta y = \frac{d_{c_2} \times y_s}{2^{n/2} - 1}, \quad (3)$$

where x_s and y_s are user-defined decimal constants. d_{c_1} and d_{c_2} are decimal numbers that correspond to binary numbers b_{c_1} and b_{c_2} , which form an n -bit binary number for a chromosome. The corner candidate is considered to be in the range $x \in [x_0, x_0 + x_s], y \in [y_0, y_0 + y_s]$. The points defined by $(x_0 + \Delta x, y_0 + \Delta y)$ are called grid points on which the genetic algorithm searches for an optimal solution.

Figure 6: Genetic search space.

The initial population for the genetic search is specified by users. In this paper, 20 candidate points are used for this population. These points are computed by Eqs. (2) and (3) as well as the random number generator.

There are many different ways to assign the start point, (x_0, y_0) . One way (Scheme 2) is that (x_0, y_0) is defined by the lower-leftmost data point that has the minimum coordinate in each coordinate direction among all the data points within the search region. (x_s, y_s) should be large enough to contain all the data points within the search region (Fig. 7). The rationale for this scheme is that all possible optimally-constrained corners are expected to fall over the area starting from this end point. Another way (Scheme 3) is that a fixed point is set as the center of genetic search region. This point is determined by the best data point in the discrete data set. Since this best data point has the minimum least-squares fitting error of multiple lines among the discrete data points inside the search region, it is reasonably expected that the optimal solution is near this point, leading to a fast search. These two schemes are implemented in this paper and experimental results are compared in Section 5.

Figure 7: Discrete data points within a search space.

The flowchart of Schemes 2 and 3 is given in Fig. 8, and it is applicable to both two-dimensional and three-dimensional spaces.

For every candidate corner, its fitness can be evaluated by least-squares fitting errors. In our algorithm, the fitting error is determined by the shortest distance between every discrete data point and the fitted line. If the fitting-line equation is $Ax + By + C = 0$, the error is given by

$$d_i = \frac{|Ax_i + By_i + C|}{\sqrt{A^2 + B^2}}, \quad (4)$$

where (x_i, y_i) is the i -th discrete data point.

Fitness function, $f(x, y)$, is defined by a reciprocal of the sum of those distances:

$$f(x, y) = 1 / \sum_{i=1}^n d_i . \quad (5)$$

In our case, the corner point has to be constrained on the two intersecting lines that meet at this point, as shown in Fig. 9. Note that these two intersecting lines are not the same as the lines fitted piecewise by the least squares. The former represents globally-fitted multiple lines, while the latter refers to two locally-fitted lines. The piecewise (i.e., local) fitting means that one line is fitted by the first part of data point on the left, and the other line is produced from the second part of data points on the right, as in Fig. 9. In this paper, the slopes of globally-fitted multiple lines are chosen to be the same as the slopes of the locally-fitted lines, respectively. In Fig. 9, the slopes of L_1 and L_3 are respectively the same as those of L_2 and L_4 .

Figure 9: Globally-fitted lines and piece-wise fitted lines.

After the fitness of every corner candidate is obtained, all the corner candidates are ranked on the basis of the value of their fitness function. The points with high ranks should be selected into a pool for crossover and mutation operations. In details, the first quarter of candidate points are duplicated into the pool. The second and third quarters of points are kept, and the last quarter of points with the lowest fitness values are removed [70]. For example, assume eight candidates in the population. (x_1, y_1) and (x_6, y_6) are the first quarter, while (x_3, y_3) , (x_5, y_5) , (x_7, y_7) and (x_8, y_8) belong to the second and third quarters, respectively. (x_2, y_2) and (x_4, y_4) are the last quarter. When the selection step based on the aforementioned scheme is completed, the population is changed to: (x_1, y_1) , (x_6, y_6) , (x_3, y_3) , (x_5, y_5) , (x_7, y_7) , (x_8, y_8) , (x_6, y_6) , (x_1, y_1) .

Individuals in the pool are crossed over to generate a next-generation population. This step is conducted according to the following crossover probability calculation [23]:

$$pc = \begin{cases} 0.9 & f_i < f_m \\ (0.9 - (0.9 - 0.6) \times (f_{\max} - f_i)) / (f_{\max} - f_m) & f_i \geq f_m \end{cases} \quad (6)$$

where pc is the probability of crossover, f_i is the fitness of i -th corner candidate, f_m is the mean value of the whole population, and $f_{\max}$ is the maximum fitness of the current generation.

Different individuals have different crossover chances. This is called adaptive crossover. A

high-quality point has a lower probability, while a low-quality point has a higher probability. Single point crossover is applied in our algorithm. With one group of binary codes each of which includes n bits for x_i and y_i , their crossovers are implemented at two randomly-selected gene positions.

Mutation operator is controlled by a probability pm , which is 0.1 in this paper. Because n bits refer to x_i and y_i , single point mutation means that binary values at two random gene positions are complemented. For instance, 0011100010 1100001100 would be mutated to 0010100010 1100101100, if the values at the fourth position of x_i and the fifth position of y_i are changed.

The best individual of the current generation is computed on the basis of their fitness values after the population at the initial or a previous step is applied to selection, crossover and mutation operators. The above procedures iterate until a maximum number of iterations is reached or a termination condition is satisfied. In this paper, the maximum number of iterations is 20, and the termination threshold can be set to be 0.1% increase, which is a ratio of fitness difference in two or three consecutive searches. (See the details in Section 5.4.)

3.2 Three-dimensional Genetic Search

In three-dimensional cases, the search space of corner points (i.e., intersection points of two 3-D lines) is also estimated by line propagation from two end points A and B, as shown in Fig. 10. The propagation proceeds on the basis of a dot product of line-segment vectors. In this paper, line-segment vector is constructed by every five data points. For instance, points 1 (i.e., point A) and 5 form the first line segment, points 3 and 7 form the second, and so on. The dot product is computed between the first and the i -th line-segment vectors. If the dot product is positive, the i -th line-segment is considered to be in a consistent direction with the first line-segment vector. Otherwise, the i -th line segment is inconsistent with the original direction, and its ending point (i.e., point C) is defined as a boundary point of a genetic search space. In the same way, the other line propagation starts from point B and ends at point D. These two points C and D determine an approximate genetic search space.

Figure 10: Line propagation in a 3-D case.

$$\begin{array}{ccc}
n \text{ bit binary number} & & \\
\hline
0011100010 & 1100001100 & 1010101111 \\
\hline
\Delta x & \Delta y & \Delta z
\end{array}$$

Figure 11: A chromosome for a 3-D case.

For each 3-D data point, the coordinate increments $(\Delta x, \Delta y, \Delta z)$ are concatenated to form an n -bit binary number, i.e., a chromosome, as depicted in Fig. 11. Those binary bits are also generated at random. The first generation of corner candidates is constructed by the following equation:

$$x_c = x_{c0} + \Delta x, y_c = y_{c0} + \Delta y, z_c = z_{c0} + \Delta z, \quad (7)$$

where (x_{c0}, y_{c0}, z_{c0}) is a point determined by a spatial translation from the data point with the minimum fitting error among existing data points in the search space, as shown in Fig. 12. The coordinate increments can be easily converted into a decimal value based on the following equation:

$$\Delta x = \frac{d_{c1} \times x_s}{2^{n/3} - 1}, \Delta y = \frac{d_{c2} \times y_s}{2^{n/3} - 1}, \Delta z = \frac{d_{c3} \times z_s}{2^{n/3} - 1}, \quad (8)$$

where x_s, y_s, z_s denote the size of the genetic search space, and should be kept constant in each genetic search. In a way similar to the two-dimensional cases, d_{c1}, d_{c2}, d_{c3} are produced by a random binary-number generator and utilized to determine the coordinate increments in the above equation.

Figure 12: Genetic search space in a 3-D case.

A symmetric form of line equation in three dimensions is given as:

$$\frac{x - x_0}{a} = \frac{y - y_0}{b} = \frac{z - z_0}{c}, \quad (9)$$

where x_0, y_0, z_0 represent the coordinates of the point passed by the line. a, b, c are direction numbers of the line. Assume that the z -component of the data is functionally dependent upon the x - and y -components. Then, equation (9) can be written as

$$\begin{cases} x = \frac{a}{c}(z - z_0) + x_0 = Az + B \\ y = \frac{b}{c}(z - z_0) + y_0 = Cz + D \end{cases}, \quad (10)$$

where A through D are coefficients. Given a set of data points $\{(x_i, y_i, z_i)\}_{i=0}^m$, the above first two coefficients are determined in a sense of least squares for letting the sum of squared errors between x_i and $Az_i + B$ be the minimum. A similar treatment can be applied between y_i and $Cz_i + D$.

Define $E(A, B) = \sum_{i=1}^m [(Az_i + B) - x_i]^2$ and $E(C, D) = \sum_{i=1}^m [(Cz_i + D) - y_i]^2$. These two functions are non-negative hyper-paraboloids that reach their minimum values when the gradients satisfy $\nabla E(A, B) = (0, 0)$ and $\nabla E(C, D) = (0, 0)$. This leads to a system of four linear equations in A, B, C and D , which can be easily solved. Precisely,

$$\begin{aligned} (0, 0) &= \nabla E(A, B) = 2 \sum_{i=1}^m [(Az_i + B) - x_i](z_i, 1), \\ (0, 0) &= \nabla E(C, D) = 2 \sum_{i=1}^m [(Cz_i + D) - y_i](z_i, 1), \end{aligned} \quad (11)$$

and thus

$$\begin{aligned} \begin{bmatrix} \sum_{i=1}^m z_i^2 & \sum_{i=1}^m z_i \\ \sum_{i=1}^m z_i & \sum_{i=1}^m 1 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} &= \begin{bmatrix} \sum_{i=1}^m x_i z_i \\ \sum_{i=1}^m x_i \end{bmatrix}, \\ \begin{bmatrix} \sum_{i=1}^m z_i^2 & \sum_{i=1}^m z_i \\ \sum_{i=1}^m z_i & \sum_{i=1}^m 1 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} &= \begin{bmatrix} \sum_{i=1}^m y_i z_i \\ \sum_{i=1}^m y_i \end{bmatrix}. \end{aligned} \quad (12)$$

A 3D line can then be represented by the following parametric equation:

$$\begin{cases} x = At + B \\ y = Ct + D \\ z = t \end{cases}, \quad (13a)$$

where t is the parameter that has the same range as z . If one line is perpendicular to the z axis, the z -component in Eqs. (10) and (13a) should be switched with x -component or y -component:

$$\begin{cases} x = At + B \\ z = Ct + D \\ y = t \end{cases} \quad \text{or} \quad \begin{cases} y = At + B \\ z = Ct + D \\ x = t \end{cases}. \quad (13b)$$

In a way similar to the two-dimensional cases, the sum of the shortest distance between every discrete data point and the fitted line is used to evaluate the fitness of each corner point candidate. As in Fig. 13, the distance between a data point and a spatial line is:

$$d_i = \frac{\left| \vec{v} \times \overrightarrow{\mathbf{M}_0 \mathbf{M}_i} \right|}{\left| \vec{v} \right|}, \quad (14)$$

where $\vec{\mathbf{v}}$ is a direction vector of the fitted line. M_i is the i -th discrete data point, and M_0 is a corner candidate point that must be on the fitted line.

Figure 13: The distance between a data point (M_i) and a spatial line ($\vec{\mathbf{v}}$).

In this paper, a fitness function, $f(x, y, z)$, is defined as

$$f(x, y, z) = 1 / \sum_{i=1}^m d_i . \quad (15)$$

Our algorithm for 3D lines consists of three main steps:

- (1) Determine the initial orientation of a line based on Eq. (13),
- (2) Conduct a position search of the optimal corner point based on the initial orientation and genetic algorithm,
- (3) Carry out an incremental orientation search of each line with the optimal corner point fixed.

For the above step (2), the selection, crossover and mutation rules are similar to those in two-dimensional cases. As to step (3), the orientation search is performed on one of the following ranges:

$$(A, C, 1 + \Delta z), \quad (A, 1 + \Delta y, C), \quad \text{or} \quad (1 + \Delta x, A, C), \quad (16)$$

where Δx , Δy and Δz represent a small amount of perturbation (less than 1) in directions x , y , z , respectively. The above three orientation ranges (represented by a vector) correspond to the three equations in Eq. (13).

4. Experimental Settings

In this study, experimental data are computed on a DELL Precision workstation with 8 CPU cores and 4 GB RAM, and some figures are presented by using MatLab software. The proposed algorithm was validated in various test cases with different parameters. Noisy discrete data points were generated by using a random number generator on the basis of perturbation from the specified line segments. The search space of genetic algorithm is defined as a rectangular prism with regular grids in each direction, as shown in Fig. 6(b). The size of the search space can be changed, and the total number of grid points (as in Fig. 6(b)) within the search space is 1024×1024 and $1024 \times 1024 \times 1024$ for two dimensional and three dimensional cases, respectively. Parameter settings for generic search are listed in Table 2. For two-dimensional cases, the initial population is 20, chromosome length is 20, and the maximum number of iteration is 20. In the cases of three dimensional data, the initial population is 30, chromosome length is 30, and the

maximum number of iteration is 10.

Table 2: Parameter settings of Schemes 2 and 3.

Parameter	Value	Description
Chromosome length	20 for x and y, 30 for x, y and z	Determined by accuracy requirement
Population size	20 for 2D, 30 for 3D	Related to efficiency of search
Selection	The first quarter of population is duplicated and last quarter is removed	High-quality individual has more probability to be kept
Probability of crossover	$pc = \begin{cases} 0.9 & f_i < f_m \\ (0.9 - (0.9 - 0.6) \times (f_{\max} - f_i)) / (f_{\max} - f_m) & f_i \geq f_m \end{cases}$	One-point and adaptive crossover
Probability of mutation	0.1	Control the diversity of population
Termination criteria	The maximum number of iteration is 20 for two-dimensional cases and 10 for three-dimensional cases	Keep a balance between accuracy and efficiency

5. Results and Discussions

The performance of our new method (Schemes 2 and 3) is compared with the traditional least-squares fitting (Scheme 1) in the following eight aspects. In addition, an exhaustive search over the search grids of genetic algorithm is implemented as a benchmark algorithm (Scheme 4), a particle swarm optimization (Scheme 5) is designed for a comparative study with respect to the genetic algorithms.

5.1 Efficiency

First, a comparison was made between the results of Scheme 1 (traditional least-squares fitting: best data point as constrained corner) and Scheme 2 (genetic search for optimally constrained point). Scheme 1 is conducted on the basis of traditional least-squares fitting, and the corner point is selected by using the discrete data point with the minimum fitting error within the approximate search space. In Scheme 2, the lower-leftmost data point of the approximate search space is set to be the lower-leftmost end point of the genetic search space. Fig. 14 demonstrates their different fitting errors. The error of the former is 22.94, while the error of the latter is 20.96. This means that our Scheme 2 reduces the fitting error by 9.5%. The coordinate values of their constrained corner are (9.80,9.93) and (9.89,10.28) respectively.

(a) traditional least-squares fitting (b) genetic search for constrained corner

Figure 14: Comparison of fitting results between Scheme 1 (traditional least-squares fitting: best data point as constrained corner) and Scheme 2 (genetic search for optimal corner).

As mentioned above, we study two ways to determine the genetic search space. One is the aforementioned Scheme 2 in which the lower-leftmost data point of the approximate space is set as a start point. The other is Scheme 3 in which the data point with the minimum fitting error is set to be at the center of genetic search. Fig. 15 illustrates the experiment results based on these two schemes. With the same set of discrete data points and the number of iterations, the fitting errors of Schemes 2 and 3 are respectively 20.12 and 17.15. The coordinate values of their constrained corner are (9.70,10.33) and (9.36,9.91), respectively. Note that the fitting error (20.12) of Scheme 2 in this paragraph is different from that (20.96) of Scheme 2 in the last paragraph, because different data sets were used. In this case, Scheme 3 approximately decreases the fitting error by 33.7% in comparison with the traditional Scheme 1 and by 17.3% from our Scheme 2.

(a) lower-leftmost data point
as one end point of search space

(b) minimum-error point as
the central location of search space

Figure 15: Comparison of Schemes 2 and 3 ((a) – Scheme 2, (b) – Scheme 3).

5.2 Robustness

Since there is randomness in the generation of the initial population of the genetic search, an average performance is more meaningful when an evaluation is conducted. In this paper, 20 executions were carried out with each of the three schemes. The experimental data is illustrated in Fig. 16. The fitting errors of two genetic searches are both better than that of the conventional

least-squares fitting (Scheme 1: best data point as constrained corner).

Figure 16: Experimental results of three different schemes (dashed line – the traditional least-squares fitting, circle symbol --- our Scheme 2, diamond symbol --- our Scheme 3)

From Fig. 16, it can be seen that the variations in the simulation results for Schemes 2 and 3 are approximately 8% and 10%, respectively. The maximum variations of these two schemes from their average values of fitting error are only 4% and 5%. Thus, our schemes are reasonably robust in terms of the fitting error.

5.3 Optimization Parameters

It is well known that the performance of evolutionary algorithms is heavily dependent upon the choice of optimization parameters. We could always design another layer of loop that traverses different settings of the optimization parameters. It is interesting to know the extent to which the optimization parameters affect the results in the context of this study.

5.3.1 Search Space

An approximate search space is obtained by the line propagation in our algorithm. The definition of genetic search space affects its optimizing capability. There are two kinds of start points that are studied in this paper. Scheme 2 is defined by lower-leftmost data point that has the minimum coordinate in each coordinate direction among all the data points within the search region, while Scheme 3 is obtained by best data point that has the minimum fitting error among the approximate search space. Scheme 3 achieves a better result of optimization than Scheme 2 (Fig. 16).

With Scheme 3 chosen, the different sizes of search space do not significantly affect the genetic search optimization, as shown in Figs. 17. A larger search space leads to a relatively smaller fitting error when the search space is small ($x_s = 0.1, y_s = 0.2$). However, when the search space becomes bigger ($x_s = 0.4, y_s = 0.8$), the size plays an insignificant role in affecting the fitting error. Fig. 17 demonstrates that all fitting errors of three sizes are ranged between 17 and 19, leading to a 5% maximum variation from the average. This indicates that our algorithm of genetic search is effective and reliable, because of its strong invariance of search space size.

Figure 17: Test results with different search space sizes.

5.3.2 Chromosome Length

Fig. 18 illustrates the performance of our algorithm at different chromosome lengths. All the fitting errors of three lengths range from 18 to 20. This means that the maximum variation of the fitting errors from their average is approximately 5%.

Figure 18: Test results with different chromosome lengths.

5.3.3 Angular Search

In 3-D cases, the data point with the minimum fitting error is assigned as the central location of genetic search. If step (3) in Section 3.2 is not conducted in our algorithm (i.e., only a genetic search of constrained corner is implemented), the fitting error is 34.44, as in Fig. 19. If step (3) is performed (i.e., an additional genetic search of orientation is applied), the fitting error was reduced to 31.95, as shown in Fig. 20. In comparison, the fitting error of the conventional approach (i.e., Scheme 1) was 48.31. This means that our approach (Scheme 3) reduces the

fitting error by 51.2% from the traditional least-squares fitting (Scheme 1). As mentioned before, the genetic search for orientation can be conducted by using one of three search ranges in Eq. (16) individually. The numerical tests indicate that these three different types of angular searches generate almost the same magnitude of fitting errors, as depicted in Fig. 21.

Figure 19: Genetic search of a constrained corner.

Figure 20: Genetic search of constrained corner and direction vector.

Figure 21: Different angular genetic searches of direction vector.

5.3.4 Initial Population

For two-dimensional cases, Fig. 22(a) demonstrates the performance of genetic search with different initial populations. The averages of fitting errors are 16.5704, 16.4969, 16.9001 and 17.6127 with respect to initial populations 10, 20, 30 and 40, respectively. Size 20 is the best setting for initial population. In three-dimensional cases, the results of fitting errors are presented in Fig. 22(b) for different initial population sizes. The average errors are 37.8162, 35.1872, 34.2645 and 34.3481 respectively for populations 10, 20, 30 and 40. Population 30 is the best initial size.

The impact of initial populations on the fitting error becomes more severe in the three-dimensional cases, where a small population may cause a very large fitting error. The best population size is different between two-dimensional and three-dimensional cases. But, the sweet spot is limited in a range [20, 30].

Figure 22: Impact of initial populations on fitting errors.

5.3.5 Probability of Mutation

Fig. 23(a) illustrates fitting errors with different pm (probability of mutation) values. The averages of fitting errors are 16.4969, 16.7169 and 16.6587 with respect to pm 0.1, 0.15 and 0.2, respectively. Value 0.1 is the best setting for pm in two-dimensional cases. The fitting errors with different pm values in three-dimensional cases are given in Fig. 23(b). The average errors are 34.2645, 34.4954 and 34.4610 respectively for pm 0.1, 0.15 and 0.2. Value 0.1 is again the best setting for pm.

5.3.6 Other Parameters

Other optimization parameters are not good candidates as independent variables. The probability of crossover is automatically computed by a self-adaptation formula in Eq. (6). The maximum iteration number is fixed at 20 in this study. The termination of iteration is controlled by this maximum number and a termination condition in Eq. (17).

5.4 Convergence

The convergence of genetic search for two-dimensional and three-dimensional cases is illustrated in Figs. 24 and 25, respectively. In the two-dimension case, there is a clear increase and obvious optimization at the first 10 iterations. After that, there is a tiny improvement until almost no change occurs (in Fig. 24(a)) or even a reduction appears (in Fig. 24(b)) at iteration 20. Similar convergence patterns were observed for all the two-dimensional cases. Hence, in this paper, the maximum number of iterations is set to be 20 for two dimensions. The termination criterion can be set to be

$$h = \frac{\max_{j=i+1}^{j=i+2}(f^{(j)}) - f^{(i)}}{f^{(i)}} \leq h_t, \quad (17)$$

where $f^{(i)}$ is the fitness at the i -th iteration, and is computed by Eq. (5) or (15). $\max_{j=i+1}^{j=i+2}(f^{(j)})$ denotes the maximum fitness in the set $\{f^{(i+1)}, f^{(i+2)}\}$. h is a fitness increase ratio, and h_t refers to a convergence threshold, which is set to be 0.15% for two-dimensional cases.

In two-dimensional cases, the increase ratio is 0.17% at iteration 19 in Fig. 24(a), and 0.19% at iteration 20 in Fig. 24(b). For position search in three dimensional spaces, there is an obvious improvement during the first 5 iterations and a slight improvement in the next 5 iterations in Fig. 25(a). After that, almost no change occurs. In the case of orientation search for a three-dimensional space, there is almost no improvement during the first five iterations. In Fig. 25(a), the increase ratio is 0.08% at the 10-th iteration. Therefore, for three-dimensional cases, the maximum number of iterations is 10, and the termination threshold can be set to be 0.1% increase.

(a) lower-leftmost point as the start point (b) minimum error point as the central point

Figure 24: Convergence of genetic search in two-dimensional cases.

(a) Position search

(b) Orientation search

Figure 25: Convergence of genetic search in three-dimensional cases.

5.5 Computation Time

The proposed genetic algorithm in two and three dimensional cases takes a reasonable amount of computational time, which is listed in Tab. 3. For a two-dimensional case, Scheme 1 takes 0.04s. Both Schemes 2 and 3 take a time shorter than 0.165s. These numbers are average values of their executions. In a three-dimensional case, Scheme 1, in which a discrete data point is assigned as the constrained corner, uses 0.3s. The running time for constrained corner search (Scheme 2) is 8.02s. Our line-fitting method for three dimensional space takes a little bit longer time. However, the performance improvement is potentially more significant, as in the last column of Table 3.

Table 3: Computation time and fitting error of genetic search schemes and traditional least-squares fitting.

Cases	Number of data points	Maximum number of iterations	running time (s)	Fitting error	Fitting Error Reduction (%)
Two dimension: Scheme 1 Discrete data point as constrained point	100	No genetic search	0.04	22.94	0
Two dimension: Scheme 2 Lower-leftmost point as start point	100	20	0.164	20.12	14
Two dimension: Scheme 3 Minimum error point as central of search	100	20	0.160	17.15	33.8
Three dimension: Scheme 1 Discrete data point as constrained point	100	No genetic search	0.30	48.31	0
Three dimension: Scheme 3 up to Step (2) Constrained corner search	100	10	8.02	34.44	40.3
Three dimension: Scheme 3 up to Step (3) Constrained corner plus orientation search	100	10(constrained corner) 5(direction vector)	15.8	31.95	51.2

5.6 Feasibility

Our test results indicate that it is possible to use genetic algorithm (GA) in handling geometric discontinuity. A simple line propagation method is designed for the fast determination of an approximate search space. The exact search boundary is constructed by a start-point position and search size. The good experimental performance of our schemes (Schemes 2 and 3) indicates that the genetic search proposed in this paper is a better way to locate exactly a constrained point at geometric discontinuity.

5.7 Flexibility

Our method is flexible to handle different sharp features with some modifications. Since it does not demand on the derivative information of lines or curves, many types of data points can be handled by the proposed approach. A similar approach can be extended to process the geometric discontinuity at the intersection of curves or the combination of curve and line.

5.8 Benefits

Although the conventional approach (Scheme 1) chooses the best data point within an approximate search space, it does not produce an optimal corner point. The genetic search algorithm developed in this paper can find a better location of multiple-line constrained corner with less fitting error, as shown in Fig. 16. It offers a better position to meet the stringent requirements on tolerance in high-precision industrial inspection and manufacturing. Furthermore, the presented genetic approach improves its search efficiency based on the line propagation in which an approximate search space is determined instead of using the whole set of data points. Another advantage of our approach is that it is especially suitable for applications in which too sparse or too dense data points appear at corner location.

5.9 Exhaustive Search

Exhaustive search herein means a complete traversal over the search grids of genetic algorithms as in Fig. 6(b). This provides a fair ground for a comparison between least-square fitting and genetic algorithm. We consider this exhaustive search as a benchmark algorithm (Scheme 4). It is expected that the fitting error would be the same for Schemes 4 and 2 or 3. The main difference is the computational time. Tab. 4 shows the numerical results of exhaustive search and genetic search. When chromosome length is very small (i.e., 6), the benchmark algorithm actually takes less time to complete the search. The reason for this phenomenon is that the genetic search always has an overhead in its computational time. The advantage of genetic search becomes more obvious with the increase in chromosome length. In all the test cases, the fitting errors are almost the same between these two types of searches.

Table 4: different fitting error and running time between benchmark algorithm and genetic search

Chromosome length	Benchmark algorithm		Genetic search	
	Fitting error	running time(s)	Fitting error	running time(s)
6	17.153	0.052	17.153	0.138
8	16.978	0.164	16.978	0.140
10	16.863	1.663	16.863	0.142
12	16.735	16.027	16.735	0.163
14	16.721	26.121	16.721	0.165
16	16.714	>200	16.714	0.172

5.10 Particle Swarm Optimization

Particle swarm optimization (PSO) is a typical method of swarm intelligence based

approaches. It is interesting to know how PSO performs against GA in the context of this study. In our PSO system, each particle is treated as a point in two-dimensional/three-dimensional space. For instance, the k -th position of the i -th particle is represented as $q_i^k = (x_i, y_i)$ for two dimension and $q_i^k = (x_i, y_i, z_i)$ for three dimension. They are updated in the same way by the following equations:

$$v_i^{k+1} = w \times v_i^k + c_1 \times r_1 \times (p_i - q_i^k) + c_2 \times r_2 \times (p_g - q_i^k), \quad (18)$$

$$q_i^{k+1} = q_i^k + v_i^{k+1}, \quad (19)$$

where $i = 1, 2, \dots, N$ and N is the size of initial population (i.e., the number of data points within the approximate corner region). k is the iteration number of particle swarm optimization.

w is the inertia weight, and is set to 0.6 in this paper. c_1 and c_2 are two positive constants, and called the cognitive and social parameters, respectively; both of them are 2 in this study. r_1 and r_2 are random numbers that are uniformly distributed within a range $[0, 0.2]$; since our search space is reduced after the high-quality initialization, we provide a smaller range instead of traditional $[0, 1]$.

The item $c_1 \times r_1 \times (p_i - q_i^k)$ in Eq. (18) represents that each individual has its memory and tracks its own travelling experience; p_i is the best previous position of the i -th particle. The item $c_2 \times r_2 \times (p_g - q_i^k)$ means that the PSO system shares the same goal of the whole particle set, and each particle adjusts its position towards the best particle among the k -th population; p_g is the position of the globally-best individual. The fitness is defined in Eq. (15).

Fig. 26 illustrates the results of comparison between PSO search and genetic search. In both two-dimensional and three-dimensional cases, the fitting errors are in a very close range. This indicates that both search schemes achieve a similar fitting error in the context of this study. The average fitting errors for PSO and GA are 17.54 and 17.39 in Fig. 26(a), while the errors are 33.8 and 33.23 respectively for PSO and GA in Fig. 26(b). Since the fitting errors of these two types are in a close range, hybrid schemes are not further explored in this study.

5.11 Real-world Laser-Scanning Data

Figure 27 shows a real-world mechanical part represented by a point data cloud, which was acquired via an industrial laser scanner. Through different cross-sections of this point cloud, six test data sets of laser-scanning data were obtained, and five of those are depicted in Figure 28. Numerical results are listed in Table 5, which indicates an average improvement of 23.8% in reducing the fitting errors.

Figure 27: Real-world laser scanning data for a mechanical part.

Table 5: Fitting errors of real-world laser-scanning data.

Test Case	Algorithm	Number of data points	Maximum number of iterations	running time (s)	Fitting error	Fitting Error Reduction (%)
1	Three dimension: Scheme 1 Discrete data point as constrained point	184	No genetic search	1.0425	0.3895	0
	Three dimension: Scheme 3 up to Step (2) Constrained corner search	184	10	15.3126	0.2980	23.48
2	Three dimension: Scheme 1 Discrete data point as constrained point	331	No genetic search	1.5706	3.0998	0
	Three dimension: Scheme 3 up to Step (2) Constrained corner search	331	10	25.7418	1.9429	37.32
3	Three dimension: Scheme 1 Discrete data point as constrained point	192	No genetic search	1.0544	0.9460	0
	Three dimension: Scheme 3 up to Step (2) Constrained corner search	192	10	18.3421	0.6004	36.53
4	Three dimension: Scheme 1 Discrete data point as constrained point	268	No genetic search	1.4098	625	0
	Three dimension: Scheme 3 up to Step (2) Constrained corner search	268	10	47.6542	588.2353	5.88
5	Three dimension: Scheme 1 Discrete data point as constrained point	268	No genetic search	1.0863	1124.354	0
	Three dimension: Scheme 3 up to Step (2)	268	10	55.3615	1010.918	10.14

	Constrained corner search					
6	Three dimension: Scheme 1 Discrete data point as constrained point	373	No genetic search	1.9699	0.7880	0
	Three dimension: Scheme 3 up to Step (2) Constrained corner search	373	10	29.5512	0.5558	29.47

6. Conclusions

In this paper, we focus on an effective data rectification method for discrete points at geometric discontinuity. A genetic search algorithm for optimally-constrained multiple-line fitting is developed. Two-dimensional and three-dimensional cases are both considered in this study. A simple line propagation method is designed for the fast determination of approximate search space. The exact search boundary is constructed by a start-point position and search size. Two schemes (i.e., Schemes 2 and 3) are devised as a more effective way to locate the corner point, i.e., intersection point between lines. Numerical computations were conducted in different test cases with different optimization parameters. The good experimental performance of these two schemes indicates that the genetic search proposed in this paper is a better way to locate exactly a constrained point at geometric discontinuity than traditional least-squares fitting. The following conclusions can be drawn:

(1) The conventional least-squares fitting (Scheme 1) does not produce an optimal corner point, even though it chooses the best data point within an approximate search space. The genetic search algorithm in this paper can find a better location of multiple-line constrained corner with less fitting error (Fig. 16). Our Scheme 3 reduces the fitting error by 33.7% (a two-dimensional case) and by 51.2% (a three-dimensional case) in comparison with the traditional Scheme 1.

(2) The definition of genetic search space affects its optimizing capability. Scheme 3 (the data point with the minimum fitting error as the central location of search space) achieves a better result of optimization (17% less fitting error) than Scheme 2 (the lower-leftmost data point as a start point) (Fig. 16).

(3) Convergence patterns are observed in both two-dimensional and three-dimensional cases. A simple termination condition is designed with a termination threshold being 0.15% (2-D cases) and 0.1% (3-D cases).

(4) Our algorithm shows its robustness across 20 different initial populations with the maximum variation from the average values of fitting error being only 4% and 5% for Schemes 2 and 3, respectively.

(5) A very small chromosome length causes a large fitting error. When the length is large, it plays an insignificant role in affecting the fitting error (Fig. 18). The sweet spot of initial population ranges from 20 to 30, while the best mutational probability is 0.1 for the problems of this paper.

(6) Both genetic search and particle swarm search generate a similar fitting error in the context of this study.

(7) There is a penalty of computational time with our algorithm, compared to the conventional least-squares fitting (Tab. 3). On the other hand, our algorithm is much faster than an exhaustive search (Scheme 4 – a benchmark algorithm) in most cases of Tab. 4.

(8) Our proposed scheme 3 reduces the fitting errors of real-world laser-scanning data by 23.8% on an average basis over six test cases.

Acknowledgement

The first author was supported by a funding (No. 201206465020) from China Scholarship Council, and this work was in part supported by U.S. National Science Foundation DMI-0514900, CMMI-0721625, ECCS-1039563, and IIP-1445355.

References

- [1] M. Desbrun, M. Meyer, P. Schroder and A. H. Barr. Anisotropic feature-preserving denoising of height fields and bivariate data. 145-152. 2000. Graphics Interface.
- [2] M. Meyer, M. Desbrun, P. Schroder and A. H. Barr, *Discrete differential-geometry operators for triangulated 2-manifolds*. In *Visualization and Mathematics III*. pp. 35-57, Springer-Verlag, Heidelberg 2003.
- [3] U. Clarenz, U. Diewald and M. Rumpf. Anisotropic geometric diffusion in surface processing. 397-405. 2000. Proceedings of IEEE Visualization.
- [4] H. Zhang and E. L. Fiume, *Mesh smoothing with shape or feature preservation*. In *Advanced in modeling, animation, and rendering*. (Ed. J. Vince and R. Earnshaw) pp. 167-182, Springer, 2002.
- [5] Y. Ohtake, A. Belyaev and I. A. Bogaevski. Mesh regularization and adaptive smoothing. Computer Aided Design 33[11], 789-800. 2001.
- [6] K. Hildebrandt and K. Polthier. Anisotropic filtering of non-linear surface features. Computer Graphics Forum 23[3], 391-400. 2004.
- [7] T. Tasdizen, R. T. Whitaker, P. Burchard and S. Osher. Anisotropic geometric diffusion in surface processing. 125-132. 2002. IEEE Visualization 2002.
- [8] C. Bajaj and G. Xu. Anisotropic diffusion of subdivision surfaces and functions on surfaces. ACM Transactions on Graphics 22[1], 4-32. 2003.
- [9] T. Bulow. Spherical diffusion for 3D surface smoothing. IEEE Transactions on Pattern Analysis and Machine Intelligence 26[12], 1650-1654. 2004.
- [10] T. Tasdizen and R. T. Whitaker. High-order nonlinear priors for surface reconstruction. IEEE Transactions on Pattern Analysis and Machine Intelligence 26[7], 878-891. 2004.
- [11] B. Mederos, L. Velho and L. H. De Figueiredo. Smoothing surface reconstruction from noisy clouds. Journal of the Brazilian Computer Society 9[3], 52-66. 2004.
- [12] T. R. Jones, F. Durant and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. 943-949. 2003. Proceedings of the 30th Annual Conference on Computer Graphics and Interactive Techniques.
- [13] S. Fleishman, I. Drori and D. Cohen-Or. Bilateral mesh denoising. 950-953. 2003. Proceedings of the 30th Annual Conference on Computer Graphics and Interactive Techniques.
- [14] P. Choudhury and J. Tumblin. The trilateral filter for high contrast images and meshes. Christensen, Per. H. and Cohen, D. 186-196. 2003. Proceedings of the Eurographics Symposium on Rendering.
- [15] B. Li, R. Schnabel, R. Klein, Z. Cheng, G. Dang and S. Jin. Robust normal estimation for point

clouds with sharp features. *Computers & Graphics* 34[2], 94-106. 2010.

- [16] J. Shen, B. Maxim and K. Akingbehin. Accurate Correction of Surface Noises of Polygonal Meshes. *International Journal for Numerical Methods in Engineering* 64[12], 1678-1698. 2005.
- [17] J. Shen, D. Yoon, D. Zhao and Y. Song. Denoising Two-Dimensional Geometric Discontinuities. *International Journal of Computers and Applications* 32[2], 1-12. 2010.
- [18] X. Sun, P. Rosin, R. Martin and F. Langbein. Fast and effective feature-preserving mesh denoising. *IEEE Transactions on Visualization and Computer Graphics* 13[5], 925-938. 2007.
- [19] X. Sun, P. Rosin, R. Martin and F. Langbein. Random walks for feature-preserving mesh denoising. *Computer Aided Geometric Design* 25[7], 437-456. 2008.
- [20] A. C. Oztireli, G. Guennebaud and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum* 28[2], 493-501. 2009.
- [21] H. Avron, A. Sharf, C. Greif and D. Cohen-Or. L1-sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics* 29[5], 135-1-135-20. 2010.
- [22] O. Schall, A. Belyaev and H. Seidel. Feature-preserving non-local denoising of static and time-varying range data. 217-222. 2007. New York, NY, USA, ACM. *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*.
- [23] R. O. Duda and P. E. Hart. Use of the Hough transform to detect lines and curves in pictures. *Communications of the ACM* 15[1], 11-15. 1972.
- [24] H. K. Aghajan and T. Kailath. SLIDE: subspace-based line detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16[1], 1057-1073. 1994.
- [25] J. Liang, D. Liu, Y. Zhao and N. Song, *Multi-line fitting using two-stage iterative adaptive approach*. In *Intelligent Robotics and Applications*. pp. 144-153, 2012.
- [26] W. E. Deming, *Data reduction and error analysis for the physical sciences*, Wiley, New York 1943.
- [27] D. York. Least squares fitting of a straight line with correlated errors. *Earth and Planetary Science Letters* 5, 320-324. 1968.
- [28] D. York. Least-squares fitting of a straight line. *Canadian Journal of Physics* 44[5], 1079-1086. 1964.
- [29] G. H. Golub and C. F. Van Loan, *Matrix computations*, p. -694, The Johns Hopkins University Press, Baltimore, Maryland, USA 1996.
- [30] S. Van Huffel and J. Vandewalle, *The total least-squares problem: computational aspects and analysis*, SIAM, 1991.

- [31] M. Krystek and M. Anton. A weighted total least-squares algorithm for fitting a straight line. *Measurement Science and Technology* 18[11], 3438-3442. 2007.
- [32] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, p. -994, Cambridge University Press, Cambridge 1992.
- [33] J. Orear. Least-squares when both variables have uncertainties. *American Journal of Physics* 50[10], 912-916. 1982.
- [34] M. Lybanon. A better least-squares method when both variables have uncertainties. No. NORDA-JA-330. 1984. Naval Ocean Research and Development Activity Stennis Space Center MS.
- [35] C. Moreno and H. Bruzzone. Parameters' variances of a least-squares determined straight line with errors in both coordinates. *Measurement Science and Technology* 4[5], 635-636. 1993.
- [36] B. C. Reed. Linear least-squares fits with errors in both coordinates. II: comments on parameter variances. *American Journal of Physics* 60[1], 59-62. 1992.
- [37] G. C. Cecchi. Error analysis of the parameters of a least-squares determined curve when both variables have uncertainties. *Measurement Science and Technology* 2[12], 1127-1128. 1991.
- [38] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. Optimization by simulated annealing. *Science* 220[4598], 671-680. 1983.
- [39] V. Cemy. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications* 45[1], 41-51. 1985.
- [40] N. Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association* 44[247], 335-341. 1949.
- [41] H. G. Beyer, *The theory of evolutionary strategies*, Heidelberg, 2001.
- [42] D. E. Goldberg, B. Korb and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems* 3[5], 493-530. 1989.
- [43] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.*, U. Michigan Press, Ann Arbor, MI, USA 1975.
- [44] H. G. Beyer and H. P. Schwefel. Evolution strategies -- A comprehensive introduction. *Natural Computing* 1[1], 3-52. 2002.
- [45] D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks* 5[1], 3-14. 1994.
- [46] K. Chellapilla. Combining mutation operators in evolutionary programming. *IEEE Transactions on Evolutionary Computation* 2[3], 91-96. 1998.

- [47] X. Yao, Y. Liu and G. Lin. Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation 3[2], 82-102. 1999.
- [48] J. Kennedy and R. Eberhart. Particle swarm optimization. 4, 1942-1948. 1995. Proceedings of IEEE International Conference on Neural Networks.
- [49] I. Rahman, P. Vasant, B. S. M. Singh and M. Abdullah-Al-Wadud. Hybrid swarm intelligence-based optimization for charging plug-in hybrid electric vehicle. 9012, 22-30. 2015. Proceedings of the 7th Asian Conference on Intelligent Information and Database Systems.
- [50] W. F. Abd-El-Wahed, A. A. Mousa and M. A. El-Shorbagy. Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems. Journal of Computational and Applied Mathematics 235[5], 1446-1453. 2011.
- [51] A. Carlisle and G. Dozier. An off-the-shelf PSO. 1, 1-6. 2001. Proceedings of the Workshop on Particle Swarm Optimization.
- [52] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11[4], 341-359. 1997.
- [53] T. Ganesan, I. Elamvazuthi, P. Vasant and K. Z. K. Shaari. Multiobjective optimization of bioactive compound extraction process via evolutionary strategies. 9012, 13-21. 2015. Lecture Notes in Computer Science. Proceedings of the 7th Asian Conference on Intelligent Information and Database Systems (ACIIDS 2015).
- [54] X. S. Yang, *A new metaheuristic bat-inspired algorithm*. In. pp. 65-74, Springer, Berlin, 2010.
- [55] X. S. Yang. Firefly algorithms for multimodal optimisation. 5792, 169-178. 2009. Lecture Notes in Computer Science. Proceedings of the 5th Symposium on Stochastic Algorithms.
- [56] X. S. Yang and S. Deb. Cuckoo search via Levy flights. 210-214. 2009. IEEE Publications, USA. Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009, India).
- [57] L. Wang. A hybrid genetic algorithm-neural network strategy for simulation optimization. Applied Mathematics and Computation 170[2], 1329-1343. 2005.
- [58] P. Vasant, *Hybrid mesh adaptive direct search genetic algorithms and line search approaches for fuzzy optimization problems in production planning*. In. pp. 779-799, Intelligent System Reference Library, 2013.
- [59] M. A. Lee and H. Takagi. Dynamic control of genetic algorithm using fuzzy logic techniques. 76-83. 1993. Morgan Kaufmann, San Mateo, CA. Proceedings of the 5th International Conference on Genetic Algorithms. Forrest, S.
- [60] F. Jimenez, G. Sanchez and P. Vasant. A multi-objective evolutionary approach for fuzzy optimization in production planning. Journal of Intelligent and Fuzzy Systems 25[2], 441-455.

2013.

- [61] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu and L. M. Wang. An improved GA and a novel PSO-GA-based hybrid algorithm. *Information Processing Letters* 93[5], 255-261. 2005.
- [62] L. Y. Tseng and S. C. Liang. A hybrid metaheuristic for the quadratic assignment problem. *Computational Optimization and Applications* 34[1], 85-113. 2005.
- [63] D. H. Kim and J. H. Cho. Robust tuning of PID controller with disturbance rejection using bacterial-foraging-based optimization. *WSEAS Transactions on Systems* 3[9], 2834-2840. 2004.
- [64] M. Vazquez and D. Whitley. A hybrid genetic algorithm for the quadratic assignment problem. 135-142. 2000. Morgan Kaufmann, San Mateo, CA. *Proceedings of Genetic and Evolutionary Computation Conference (GECCO)*.
- [65] K. Ganesh and M. Punniyamoorthy. Optimization of continuous-time production planning using hybrid genetic algorithm-simulated annealing. *International Journal of Advanced Manufacturing Technology* 26[1], 148-154. 2004.
- [66] S. Minton, M. D. Johnston, A. B. Philips and P. Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and schedule problems. *Artificial Intelligence* 58[1], 161-205. 1992.
- [67] G. Dozier, J. Bowen and A. Homaifar. Solving constraint satisfaction problems using hybrid evolutionary search. *IEEE Transactions on Evolutionary Computation* 2[1], 23-33. 1998.
- [68] Y. M. Park, J. B. Park and J. R. Won. A hybrid genetic algorithm/dynamic programming approach to optimal long-term generation expansion planning. *International Journal of Electrical Power & Energy Systems* 20[4], 295-303. 1998.
- [69] R. K. Ahuja, J. B. Orlin and A. Tiwari. A greedy genetic algorithm for the quadratic assignment problem. *Computers and Operations Research* 27[10], 917-934. 2000.
- [70] Q. Zhang and S. Chang. An improved crossover operator of genetic algorithm. 2, 82-86. 2009. *Proceedings of Computational Intelligence and Design (ISCID'09)*.

Xue-Fei Shi received the B.S. degree in Automation Instrument from University of Science & Technology Beijing in 1994 and the M.S. degree in Detection Technology and Automatic Equipment from USTB in 2000. She achieved the Ph.D. degree in Control Science and Engineering from USTB in 2011. She has been a visiting scholar of University of Texas at Arlington for four months in 2006. She is currently a visiting scholar of University of Michigan at Dearborn. Her research interest includes detection and control technology, computer modeling and optimization technology and machine vision.

Jie Shen received the B.S. degree and M.S. degree in Vehcile Engineering from Jiangsu Institute of Technology, P.R. China in 1982 and 1985, respectively. He obtained the M.S. degree and Ph.D. in Computer Science from the University of Saskatchewan, Canada in 1997 and 2000.

From 2002 to 2008, he was an assistant professor with the University of Michigan – Dearborn. From 2008 to present, he was an associated professor and then a professor in the same institute.

Dr. Shen has published one book and more than 110 technical papers. He serves as the Editor-in-Chief for *International Journal of Modelling and Simulation* and as an editorial board member for *International Journal of Computer and Applications*. He is the co-chair of 2012 International Conference of Computer Aided Design, the organizer of Modeling and Simulation of Material Failure (IMECE 2015), and the workshop organizer of International Conference on Computational Science and Applications (ICCSA) in last three years. Dr. Shen contributed to the development of two worldwide-known commercial software packages: OPTISTRUCT® & HYPERMESH®, which were widely used among Fortune 500 companies. He is the inventor of the University of Michigan Invention 3805, which was licensed to a U.S. company of laser sensors.

David Yoon obtained his Ph.D. in Computer Science from Wayne State University in 1989 and currently is an associate professor in the Department of Computer and Information Science at the University of Michigan- Dearborn.

Dr. Yoon's research areas include computer graphics, geometric modeling, computer aided design and manufacturing.