applied sciences

MDPI

Article

Geometry-Aware Hatching Toolpath Selection and Parameter Optimization for Laser Marking of Complex Two-Dimensional Contours

Zuoping Xu [1], Yifeng Cui [2,*], Jen-Chieh Wang [3,4] and Jinxiao Yang [5]

[1] School of Information Science and Engineering, Linyi University, Linyi 276000, China; xuzuoping@lyu.edu.cn
[2] School of Logistics, Linyi University, Linyi 276000, China
[3] Department of Transportation Science, National Taiwan Ocean University, Keelung 202301, Taiwan; kimiwang@mail.ntou.edu.tw
[4] Department of Business Administration, Asia University, Taichung 413305, Taiwan
[5] School of Software, Xi'an Jiaotong University, Xi'an 710049, China; hhacl289@163.com
* Correspondence: cuiyifeng@lyu.edu.cn

Abstract

Laser marking of two-dimensional composite contours requires hatching paths that balance path length, computation time, coverage continuity, and marking accuracy. Existing industrial and academic systems commonly provide zigzag and contour-parallel hatching strategies, but the choice between them is often made empirically, especially for nested or irregular contours. This study is positioned as a geometry-aware toolpath-selection and parameter-optimization framework. Three benchmark contour types, namely a single-layer regular contour, a nested multi-layer contour, and an irregular contour, are evaluated using zigzag parallel hatching and contour-parallel hatching. The MATLAB results show that for the irregular benchmark contour, zigzag hatching reduces the toolpath-generation time by 73.91–89.62% compared with contour-parallel hatching when the hatch spacing varies from 0.01 to 0.10 mm. Five repeated numerical runs further show that the maximum relative timing deviation is below 0.003%, indicating stable computation in the software environment. To avoid confusing algorithmic computation time with real machine execution time, this paper explicitly separates toolpath-generation time from physical marking time and outlines a repeatable physical validation protocol including execution time, dimensional error, mark contrast, and line width uniformity. The results provide practical guidance for selecting hatching strategies according to contour complexity and accuracy requirements.

Keywords: laser marking; hatching path; toolpath generation; contour-parallel hatching; zigzag hatching; shape complexity; particle swarm optimization

Check for updates

Academic Editors: Herbert Schneckenburger and Arkadiusz Gola

Received: 13 May 2026
Revised: 27 June 2026
Accepted: 30 June 2026
Published: 6 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Laser marking is a non-contact manufacturing process in which a focused laser beam modifies the surface of a target material to generate permanent text, graphics, codes, or functional marks. In modern marking systems, the optical and galvanometer components determine the physical capability of the machine, whereas the software-generated toolpath determines how the beam scans the designed contour. For complex two-dimensional graphics, the hatching path directly affects the total marked length, jump length, number of starts and stops, sharp turns, local over-marking, and final marking uniformity [1]. In related laser surface texturing and engraving studies, scanning strategy and process

parameter selection has been shown to affect surface accuracy, texture quality, and productivity [2,3]. These findings indicate that hatching-path design should be considered as a geometry-dependent planning problem rather than as a purely graphical filling operation.

Among the most widely used filling strategies, zigzag or raster hatching scans the interior of a contour using approximately parallel lines, while contour-parallel hatching fills the region by inward offsets of the boundary. These two strategies are not new; they are common in laser-marking software, computer-aided manufacturing, and additive-manufacturing path planning. Previous studies have examined laser-marking path optimization using genetic algorithms, contour extraction, graphic hatching, and image-processing-based filling methods [4–8]. These works provide useful implementations for contour extraction and filling generation, but they give limited guidance on how the same pair of basic hatching strategies should be selected when the contour changes from regular to nested or irregular geometry. In additive manufacturing, path planning has likewise been shown to influence accuracy, surface quality, build time, and material use, and hybrid contour-zigzag paths have been proposed to improve continuity and filling performance [9,10]. This paper establishes a reproducible and interpretable framework for comparing the two basic strategies, zigzag hatching and contour-parallel hatching, for different contour geometries.

The practical problem addressed in this work is the following: given a two-dimensional contour with possible nested boundaries or irregular curvature, how can one select a suitable hatching strategy and hatch spacing so that the path-generation process is efficient while maintaining acceptable geometric coverage? If the contour is geometrically regular and has few holes or concave regions, zigzag hatching is expected to provide a shorter computation time and simpler path generation. If the contour is highly irregular, contains nested boundaries, or requires better boundary-following behavior, contour-parallel hatching may provide better geometric consistency but usually at the cost of a longer computation time and more offset operations.

To make this problem statement explicit, this paper distinguishes three different quantities: (i) toolpath-generation time, which is the MATLAB program running time needed to generate and evaluate the path; (ii) geometric path indicators, including total marking length, number of hatch lines or contour loops, and contour complexity; and (iii) machine execution time, which must be measured on a real laser-marking system and is affected by scanner acceleration, jump delay, laser power, pulse frequency, material properties, and controller settings. This distinction is essential because a short algorithmic running time does not automatically imply a shorter physical marking time.

The main contributions of this paper are summarized as follows:

1. A geometry-aware comparison framework is established for zigzag parallel hatching and contour-parallel hatching over single-layer, nested multi-layer, and irregular two-dimensional contours.
2. A shape complexity descriptor is introduced to support the distinction between regular and complex contours using compactness, hole number, concavity, and curvature variation.
3. The roles of the Hough transform, curve fitting, and particle swarm optimization are clarified. In particular, Hough-based line fitting is treated as an optional raster-contour preprocessing step, while scanline-polygon intersection is recommended for vector contours.
4. The numerical results are reported as toolpath-generation results rather than physical machine-processing results, and repeated runs are summarized using mean, standard deviation, maximum absolute deviation, and maximum relative deviation.

5. A physical laser-marking validation protocol is outlined to support future experimental verification using real execution time, dimensional error, mark contrast, line width uniformity, and sample photographs.

2. Materials and Methods

2.1. Benchmark Contours and Hatching Task

For the single-layer contour, the zigzag and contour-parallel hatching procedures are evaluated by calculating the total hatching-curve length, the number of parallel lines or contour loops, and the program running time.

For single-layer contours, we do the following:

1: The discrete points in the first benchmark dataset are fitted into continuous smooth curves in MATLAB2022a, and the irregular single-layer contour is divided into several simpler blocks for subsequent hatching analysis.

2: Under the zigzag parallel hatching entity, the single-layer contour is made fractal by the zigzag function, the figure is processed into binary edge graphics, the set of coordinate points in the contour is extracted, and then the discrete points are fitted into straight lines by the combination of the Hough transform and least-square method [11] so as to calculate the length of the parallel lines. After the contour is made fractal, the height of each block is divided by the number of zigzag parallel lines.

3: Under contour-parallel hatching, representative discrete points are selected in each part of the contour. Each selected point is used as the center of a circle, and the inward offset distance is used as the radius. A smooth curve tangent to these circles is then constructed to obtain the nearest offset contour. The curve length is calculated using the first kind of curvilinear integral [12]. The multi-objective particle swarm optimization algorithm is then used to search for the deepest feasible contour layer, and the spacing between adjacent contour lines is used to estimate the number of contour loops.

4: The running times of two kinds of hatching entity programs are timed by clock function.

For contours with different shapes and hatch spacings, the two marking methods are tested under the same numerical framework. The goal is to compare their path-generation efficiency and geometric behavior under the following controlled conditions:

1: The hatching procedures are first evaluated on the single-layer and nested contours. An additional irregular contour composed of many discrete coordinates is then used to test whether the same procedures remain applicable to more complex geometry. The data obtained from the three contour types are compared to identify the relative advantages of zigzag parallel hatching and contour-parallel hatching.

2: The hatching procedure is then extended from regular straight-line contours to irregular curved contours, and the parameter results are used to observe how contour shape affects the algorithm. These results help identify the conditions under which either zigzag parallel hatching or contour-parallel hatching is more suitable.

3: The hatch spacing is varied in the interval [0.01, 0.1], and MATLAB is used to generate hatching paths for the fixed benchmark shapes so that the change in time T can be observed.

4: The optimization model is created by controlling the variables and using the obtained parameters to define a reference objective function $f(x)$ for subsequent parameter analysis.

2.2. Toolpath Modeling and Optimization Framework

Interpolation curve modeling is used to obtain an approximate boundary function from a limited set of known data points [13]. Polynomial interpolation, including Lagrange interpolation and Newton interpolation, is used to approximate the contour curve. The

optimal solution is obtained by establishing a mathematical model based on particle swarm optimization (PSO) [14]. The optimized parameters are used to calculate the number of parallel lines and contour loops. The crowding distance [15] is calculated and sorted to estimate the density of neighboring solutions around each candidate solution.

For each objective function, the solutions in the external optimal solution set are sorted by their objective values. The crowding distance is then calculated from the neighboring solutions, with the boundary solutions assigned an infinite crowding distance. A smaller crowding distance indicates a denser local distribution and lower diversity, whereas a larger crowding distance indicates a sparser local distribution and higher diversity. Finally, the particles are reordered according to the calculated crowding distance, and the global optimal value is selected from the group with a relatively large crowding distance [15].

$$i = 0.1(nondom - 1), j = Random(0, i), Gbest = A[j]$$

Tips: The variable nondom is the number of non-inferior solutions, and $A[j]$ is a random solution with a large crowding distance in the elite solution set. At this time, the randomly selected Gbest is an individual in the scattered region of the front end of the Pareto.

3. Experimental Design and Results

3.1. Software Environment and Scope of Numerical Evaluation

The numerical experiments were implemented in MATLAB and were used to evaluate software-level toolpath generation. The reported running time in the numerical sections is therefore the toolpath-generation time, not the real laser-marking execution time. The following assumptions are used in the software evaluation: (i) the input contour is represented by discrete coordinate points and can be reconstructed into a continuous boundary; (ii) the marked area can be divided into subregions suitable for scanline or contour-offset operations; (iii) the hatch spacing and inward offset distance are controllable geometric parameters; and (iv) material response, laser power, pulse frequency, scanner acceleration, and jump delay are excluded from the numerical timing model and must be evaluated separately in physical marking experiments.

3.2. Symbols and Definitions

For clarity and consistency, this section summarizes the principal symbols used throughout the proposed geometry-aware hatching-path selection and optimization framework, as shown in Table 1.

Table 1. A table consisting of a number of representative discrete points used to solve the model.

Symbols	Definitions
J	Least-square index of curve
$T(d)$	The hatch spacing is D-round's hatching time
C_c	The total length of contour
C_h	The total length of horizontal lines
I	The number of the drawing area
N_c	The number of contour
N_h	The total length of horizontal lines

3.3. Shape Complexity Descriptor and Strategy Selection Criterion

To avoid an ambiguous distinction between "regular" and "complex" contours, a geometry-based descriptor is introduced. For a two-dimensional contour with area A and perimeter P, the compactness is defined as

$$K = \frac{P^2}{4\pi A}. \tag{1}$$

For a circle, $K = 1$, while elongated, concave, or highly irregular contours produce larger values. In addition, the number of internal holes H, the concavity ratio R_c, and the curvature coefficient of variation CV_κ are used to describe nested boundaries and local shape fluctuation. The composite contour-complexity score can be written as

$$S_c = \alpha K + \beta H + \gamma R_c + \delta CV_\kappa, \tag{2}$$

where α, β, γ, and δ are non-negative weights selected according to the application requirement. A lower S_c indicates a more regular contour, for which zigzag hatching is usually efficient. A higher S_c indicates a more complex contour, for which contour-parallel hatching or a hybrid strategy may be preferable if boundary-following accuracy is more important than computation time.

In the present numerical experiments, this descriptor is used as an explanatory criterion rather than as a trained classifier. It helps clarify why the same hatch spacing may lead to different relative advantages for single-layer, nested multi-layer, and irregular contours.

3.4. PSO-Based Parameter Optimization Objective

The particle swarm optimization component is used to optimize hatching-related parameters rather than to define the basic hatching strategies themselves. The design variables may include hatch spacing d, scanning angle θ, inward offset layer index k, and local connection order q. The general multi-objective cost function is formulated as

$$J(d, \theta, k, q) = w_1 L_{mark} + w_2 L_{jump} + w_3 N_{start} + w_4 C_{turn} + w_5 E_{cover}, \tag{3}$$

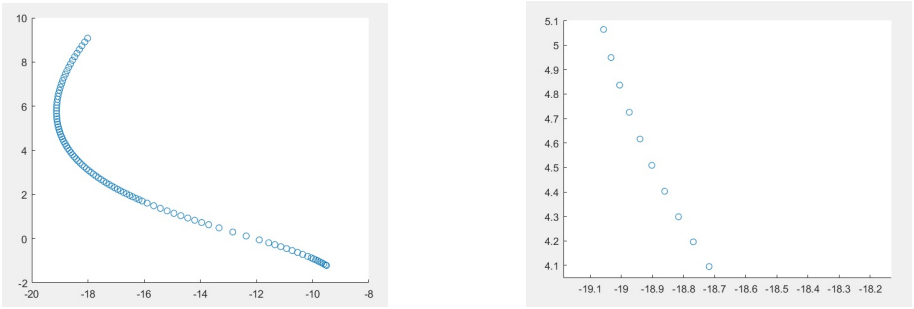
where L_{mark} is the marking length, L_{jump} is the non-marking jump length, N_{start} is the number of laser on/off starts, C_{turn} penalizes sharp turns, and E_{cover} measures uncovered and over-covered regions. The weights w_1–w_5 determine the relative importance of efficiency, path continuity, and coverage quality.

3.5. Benchmark Contour Description

To reconstruct the object boundary as accurately as possible, high-precision floating-point coordinate points are used as the benchmark contour data. These scattered points are fitted into smooth curves in MATLAB before hatching-path generation.

The Hough transform is used only when the input boundary is obtained from a raster image and near-straight segments must be detected before length calculation. For vector CAD or polygonal contour input, direct scanline-polygon intersection is a more direct and reproducible solution because it avoids unnecessary raster-to-line fitting. Similarly, global Lagrange interpolation is sensitive to oscillation when many nodes are used. Therefore, Lagrange interpolation is retained only as a local interpolation model in this paper, while piecewise cubic splines, B-splines, and NURBS are recommended for dense contour reconstruction in practical software implementations.

We select 100 data points from the data taken (as shown in Figure 1a), enlarge the graph, and show a scatter distribution (as shown in Figure 1b).



(**a**) (**b**)

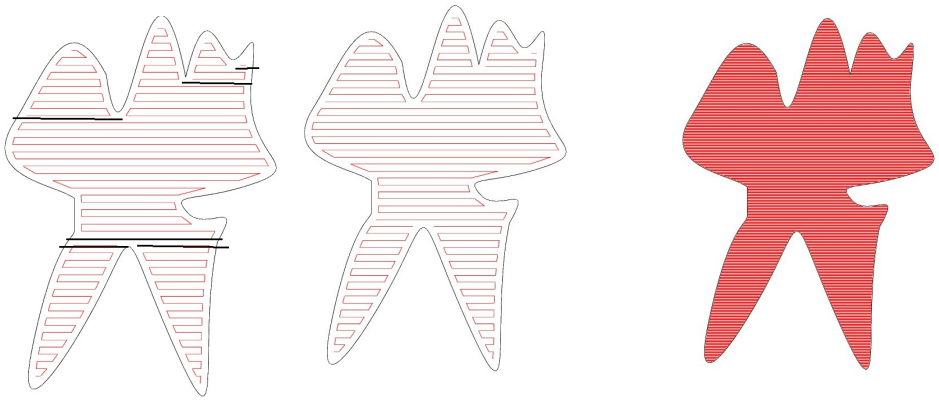
Figure 1. (**a**) The outline of the selected 100 points. (**b**) Enlarged data point.

3.6. Hatching Path Generation Procedures

3.6.1. Single-Layer Contours

Firstly, the hatching model of the zigzag parallel hatch is established as follows:

In order to obtain the length and number of parallel lines and contours of zigzag and contour lines, as well as the number of contour coils in single-layer contours under hypothetical conditions, the data given by us should be processed into smooth curves by MATLAB, and the Zigzag index algorithm model is established to fractalize the single-layer contours (Figure 2).



(**a**) (**b**) (**c**)

Figure 2. (**a**) Single-layer contour fractal analysis figure. (**b**) Internal contraction of boundary distance: 1 mm; hatch line spacing: 1 mm. (**c**) Internal contraction of boundary distance: 0.1 mm; hatch line spacing: 0.1 mm.

Solution of hatch model of zigzag parallel lines:

When using the zigzag parallel line method, calculate the height of each figure—the height divided by the hatch spacing is the number of horizontal lines in the broken-line parallel line—and extract the set of data points in the binary graph. The length of the zigzag parallel line is calculated by combining the Hough transform and the least-square method to fit the coordinates into a straight line [11].

According to the data given by us, we can get the single-layer contour. Let the set of coordinate points in the profile be $M = (x_i, y_i)^T, (i = 1, 2, \ldots, s)$ (s is the number of data points in the dataset), The data points in M are distributed near R lines, and the error threshold is set to d_k. Under the standard parameterization mode, the expression of the straight line of the Hough transform in the plane's Cartesian coordinate system is

$$\rho = x \cos \vartheta + y \sin \vartheta; \rho \geq 0, 0 \leq \vartheta \leq 2\pi; \quad (4)$$

Rewrite Formula (1) as follows:

$$\rho_k = x_i \cos \vartheta_k + y_i \sin \vartheta_k; (i = 1, 2, \ldots, s; k = 1, 2, \ldots r); \quad (5)$$

According to Formula (2), M is transformed by the Hough transform, and the parameters of the fitting line can be obtained as follows: (v_k, ρ_k).

Record the set of points near the fitting line as M_k^*. Rewrite Formula (2) in its slope intercept form as follows:

$$y_i = a_k x_i + b_k \quad (6)$$

of which $a_k = -\frac{\cos \vartheta_k}{\sin \vartheta_k}, b_k = -\frac{\rho_k}{\sin \vartheta_k}$;
Calculate the distance from a point in M to a straight line as follows:

$$d_{ki} = \frac{|a_k x_i + b_k - y_i|}{\sqrt{a_k{}^2 + 1}} (x_i, y_i) \in M, (i = 1, 2, \ldots, s; k = 1, 2, \ldots, R); \quad (7)$$

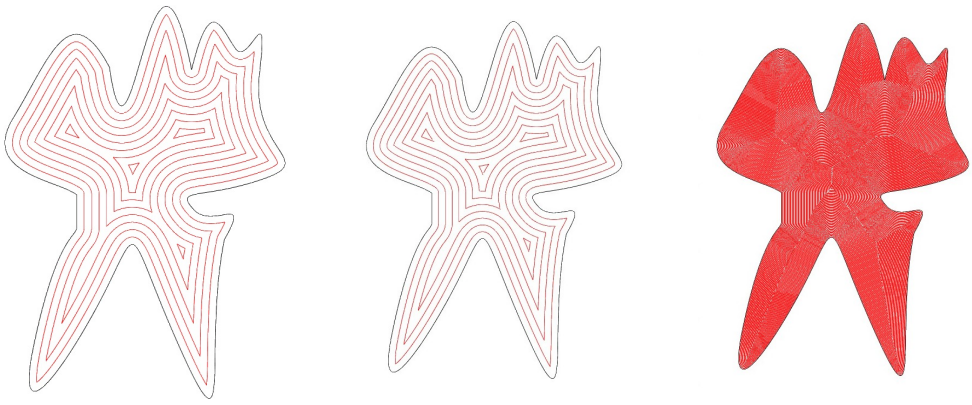
If $d_{ki} < d_k$,
then $(x_i, y_i) \Longrightarrow M_k^*(x_{kj}, y_{kj}), (j = 1, 2, \ldots, s^*, s^* <= s)$.

M_k^* is the set of points near the line of the K_{th}^* Hough transform that meets the requirement of the error threshold.

Taking the point set M_k^* as the fitting data, fitting each straight line, respectively, the parameters of the straight line Equation (8) can be obtained (a_k^*, b_k^*). Choose $((x_{kj})_{min}, y_{kj}) \in M_k^*$ and $((x_{kj})_{max}, y_{kj}) \in M_k^*$ as endpoints, so the interval of each straight-line segment can be determined as follows:

$$y_{kj} = a_k^* x_{kj} + b_k^*, \left(x_{kj}\right)_{\min} \leq x_{kj} \leq \left(x_{kj}\right)_{\max}. \quad (8)$$

Enter the data and get the following results:
Internal contraction of boundary distance of 1 mm and hatch line spacing of 1 mm:
length of hatching lines: 196.4336 mm; number of hatching horizontal lines: 85.
Internal contraction of boundary distance of 0.1 mm and hatch line spacing of 0.1 mm:
length of hatching lines: 989.5264 mm; number of hatching horizontal lines: 908.
Establishment and solution of model of contour-parallel hatch (Figure 3a). The hatching model of contour-parallel hatches is established as follows:



(**a**) (**b**) (**c**)

Figure 3. (**a**) Single-layer contour fractal analysis diagram. (**b**) Internal contraction of boundary distance: 1 mm; hatch line spacing: 1 mm. (**c**) Internal contraction of boundary distance: 0.1 mm; hatch line spacing: 0.1 mm.

The outline is reconstructed using a smooth curve fitted from the discrete-point parameters, as shown in (Figure 4).

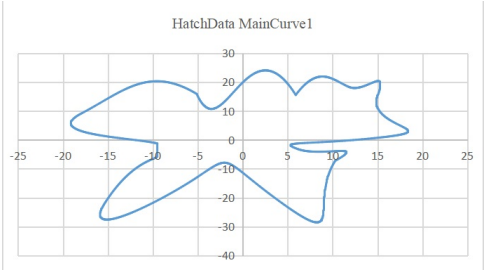


Figure 4. Outline drawing.

Select some discrete points, take the discrete points as the center of the circle, take the internal indentation distance as the radius, specify the circle center, use a smooth curve to draw the tangent to each circle, obtain the tangent point, and use the algorithm to divide the curve into various parts. Find out the function of each part, and this smooth curve is the contour line closest to the contour, as shown in (Figure 5).

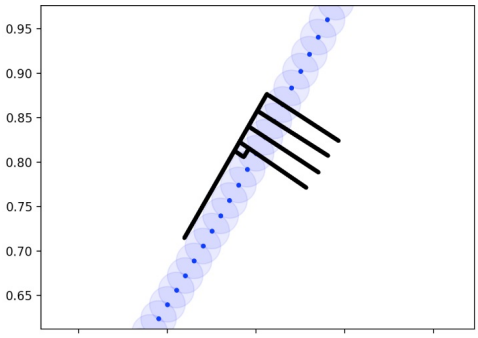


Figure 5. To find the tangent, each discrete point (x, y) is used to simulate circle 1.

The existing curve function is used to approximately replace the contours. Solution of the contour-parallel hatching model.

The curve is fitted using the newly generated tangent points. Because the curve must pass through these points, the Lagrange interpolation polynomial is used to obtain an approximate curve function and construct a set of basis functions [16].

$$\begin{aligned} l_i(x) &= \frac{(x-x_0)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} \\ &= \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}, (i = 0, 1, \ldots n) \end{aligned} \tag{9}$$

$l_i(x)$ is a polynomial of degree n, which satisfies

$$l_i(x_j) = \left\{ \begin{array}{l} 0 \ldots j \neq i \\ 1 \ldots j = i \end{array} \right\} \tag{10}$$

Let

$$L_n(x) = \sum_{i=0}^{n} y_i l_i(x) = \sum_{i=0}^{n} y_i \left(\prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j} \right) \tag{11}$$

In the selected interval $[a, b]$, find a polynomial of degree n at most from the function values $y_i = f(x_i)(i = 0, 1 \ldots, n)$ at $n + 1$ different points $x_0, x_1, \ldots, x_n$:

$$\varphi_n(x) = a_0 + a_1 x + \ldots + a_n x^n \tag{12}$$

Make it have the same value as $f(x)$ at a given point, that is, it satisfies the following interpolation condition:

$$\varphi_n(x_i) = f(x_i) = y_i(i = 0, 1 \ldots, n) \tag{13}$$

φ_n is called an interpolation polynomial.

$x_i(i = 0, 1, \ldots\ n)$ is called an interpolation node. Make a polynomial curve $y = f(x)$. The N-degree polynomial (1) has $n + 1$ undetermined coefficient, and the interpolation condition (2) gives exactly an n + 1 equation.

$$\begin{cases} a_0 + a_1 x_0^n + \ldots + a_n x_0^n = y_0 \\ a_0 + a_1 x_1^n + \ldots + a_n x_1^n = y_1 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_0 + a_1 x_n^n + \ldots + a_n x_n^n = y_n \end{cases} \tag{14}$$

If the coefficient matrix of this system of equations is A, then

$$\det(A) = \begin{pmatrix} 1 & \cdots\cdots & x_0^n \\ \vdots & \ddots & \vdots \\ 1 & \cdots\cdots & x_n^n \end{pmatrix} \tag{15}$$

When the $x_0, x_1, \ldots, x_n$ values are different from each other, the value of this determinant is not zero. Therefore, the system of Equation (3) has a unique solution. Get the corresponding functional $f(x)$.

By introducing the obtained functional $f(x)$ into the first kind of curve integral formula [12], the selected part of the contour length can be obtained as follows:

$$\int_L f(x, y)ds = \lim_{\lambda \to \infty} \sum_{i=1}^{n} f(\xi_i, \eta_i)\Delta s_i \tag{16}$$

The curve length d_1 of the selected part can be obtained according to the selected interval $[a, b]$, and the curve function can be obtained from the difference curve.

In the process of creating the model, the discrete points are divided into n (n (countable constant) segment curves according to monotonicity, inflection points, extreme points, special points, etc.), the length d of each curve is obtained, and the following bus lengths can be obtained by adding the lengths of each segment:

$$C_c = \sum_{i=1}^{n} C_1 + C_2 + \ldots C_n \tag{17}$$

The data given in Table 2 are modeled and solved, and the integrated shape does not need to be divided into regions. The contours are drawn and the algorithm is used to solve the value range of the bus length x. The value range of y is $[-19.12163645, 18.4209696]$. The value range of y is $[-28.53424723, 24.06724352]$.

Enter the data and get the following result:

Internal contraction of boundary distance of 1 mm and hatch line spacing of 1 mm:

Length of hatching lines: 896.8205 mm.

Internal contraction of boundary distance of 0.1 mm and hatch line spacing of 0.1 mm:

Length of hatching lines: 10,017.094 mm.

When hatching using the method of parallel contours, the multi-objective particle swarm optimization algorithm is used to find the optimal solution, that is, the deepest part of the contours, and the distance between the optimal solution and the farthest contour is

calculated. The number of loops of the contours can be obtained by dividing the distance by the distance.

Table 2. A table consisting of a number of representative discrete points used to solve the model.

Center of the Circle X	Center Ordinate Y	Radius R
15.66162	0.974086	1
15.48854	0.912793	1
15.30922	0.850808	1
15.21720	0.819546	1
15.03235	0.757615	1
14.66724	0.638357	1
14.30833	0.525069	1

Obtained from the formula of the particle swarm optimization algorithm, the formula for determining the speed and position of particles in the search space is

$$v(t+1) = \omega v(t) + c_1 \cdot r() \cdot (Pbest - x(t)) + c_2 \cdot r() \cdot (Gbest - x(t)); \quad (18)$$

$$x(t+1) = x(t) + v(t+1) \quad (19)$$

where ω is the inertia weight coefficient.

The best position of the particle itself and the global best position are *Pbest* and *Gbest* of the t generation, respectively. c_1 and c_2 are acceleration constants. $r()$ is a random number with a uniform distribution on $[0, 1]$ [17].

The formula after maintaining the inertia weight factor and acceleration factor is

$$v(t+1) = k[\omega v(t) + c_1 \cdot r() \cdot (Pbest - x(t)) + \omega v(t) + c_2 \cdot r() \cdot (Gbest - x(t))] \quad (20)$$

where $k \in [0.7, 1.0]$ (k is a random number in $[0.7, 1.0]$).

So, we get the following result:

Internal contraction of boundary distance of 1 mm and hatch line spacing of 1 mm:

Length of hatching lines: 896.8205 mm; Number of hatching circles: 8.

Internal contraction of boundary distance of 0.1 mm and hatch line spacing of 0.1 mm:

Length of hatching lines: 10,017.094 mm; Number of hatching circles: 85.

3.6.2. Nested Multi-Layer Contours

Firstly, the hatching model of the zigzag parallel hatch is established as follows:

In order to obtain the length, the number of parallel lines, and the number of contour coils of the zigzag parallel lines and contours in the multi-layer test, the following must be conducted.

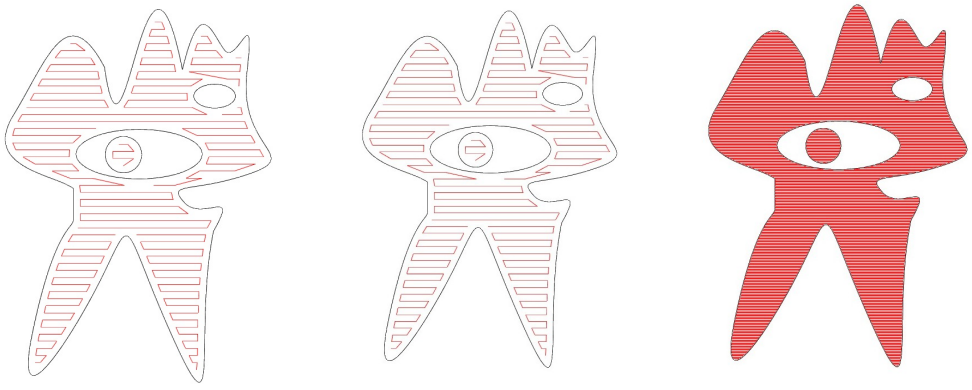
Considering the contours under hypothetical conditions, the data should be processed into smooth curves by MATLAB, and the nested contours should be separated using the zigzag index algorithm model.

The separated single-layer zigzag parallel (Figure 6a) solves the problem by using the method in Section 3.6.1 namely the establishment of parallel hatching model of nested multi-layer contour zigzag parallel.

The solution to the hatching model of the nested zigzag parallel lines is as follows:

When using zigzag parallel lines to hatch entities, the nested contours are separated first, and the height divided by the hatch spacing is the number of horizontal lines comprising the broken parallel lines. The set of data points in the binary graph is thus extracted. The length of the zigzag parallel line is calculated by combining the Hough transform and the least-square method to fit the coordinates into a straight line.

According to the given data, the nested profile can be obtained. After the contour is separated, the set of coordinate points in the contour is set to M $=(x_i, y_i)^T, (i = 1, 2, \cdots, s,)$ (s is the number of data points in a dataset). The data points in M are distributed near R lines, and the error threshold is set to d_k.



(**a**) (**b**) (**c**)

Figure 6. (**a**) Nested multi-layer contour analysis diagram. (**b**) Internal contraction of boundary distance: 1 mm; hatch line spacing: 1 mm. (**c**) Internal contraction of boundary distance: 0.1 mm; hatch line spacing: 0.1 mm.

Taking the point set d_k^* as the fitting data, fitting each straight line, respectively, the parameters of the straight line Equation (3) can be obtained as (a_k^*, b_k^*) [16]. Choose $((x_{kj})_{\min}, y_{kj}) \in M_k^*$ and $\left(\left(x_{kj}\right)_{\max}, y_{kj}\right) \in M_k^*$ as endpoints. The interval of each straight line segment can be determined, that is,

$$y_{kj} = a_k^* x_{kj} + b_k^*, \left(x_{kj}\right)_{\min} \leq x_{kj} \leq \left(x_{kj}\right)_{\max}. \tag{21}$$

The results can be obtained according to the models and formulas in the single-layer profile as follows:

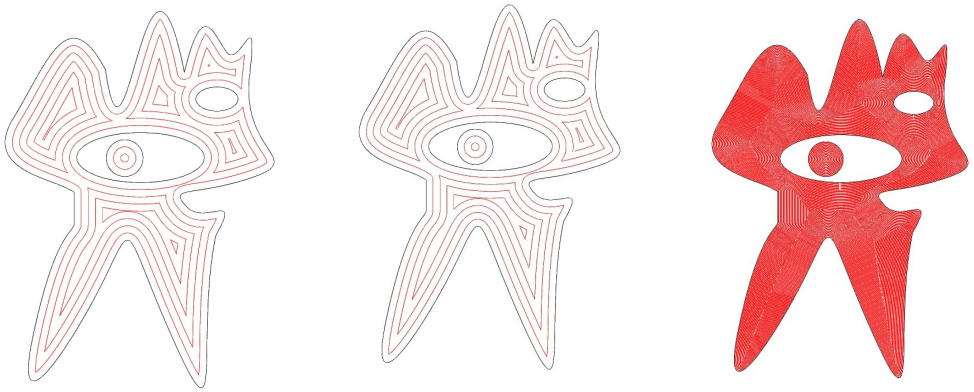
Internal contraction of boundary distance of 1 mm and hatch line spacing of 1 mm:

Length of hatching lines: 189.0599 mm; number of hatching horizontal lines: 50.

Internal contraction of boundary distance of 0.1 mm and hatch line spacing of 0.1 mm:

Length of hatching lines: 1306.2959 mm; number of hatching horizontal lines: 525.

The entity of the contour-parallel hatch line is shown in Figure 7a. This figure shows the establishment and solution of the model of the contour-parallel hatch with a nested contour:



(**a**) (**b**) (**c**)

Figure 7. (**a**) Nested multi-layer contour analysis diagram. (**b**) Internal contraction of boundary distance: 1 mm; hatch line spacing: 1 mm. (**c**) Internal contraction of boundary distance: 0.1 mm; hatch line spacing: 0.1 mm.

Solution of hatched entities with nested contours:

By separating the nested contour model [18] then using the same problem method, one can get the following results:

Internal contraction of boundary distance of 1 mm and hatch line spacing of 1 mm:

Length of hatching lines: 773.0452 mm; number of hatching circles: 5.

Internal contraction of boundary distance of 0.1 mm and hatch line spacing of 0.1 mm:

Length of hatching lines: 9060.1234 mm; number of hatching circles: 59.

3.6.3. Irregular Contours

For different shapes of graphics, different hatch spacings and different marking methods are tested.

In the process of laser marking, the time of task execution [19], the number of paths, and the distance of laser movement are different according to the size of the hatch line spacing, the shape of the polygon, and the type of marking. However, the production of these different results will lead to differences in work efficiency in practical applications. Treat the external indentation margin d, hatch spacing l, shape of the polygon x, hatching line type y, time T, path number N, and laser moving distance C as the dependent variables [19].

In order to meet the needs of the actual industry, a standard is established so that the time T is relatively short, the number of paths N is relatively small, and the laser moving distance C is relatively short. Using the optimization algorithm to create a mathematical model [20], on the basis of the original given data, this paper obtains several groups of data by adding new contours and constantly changing the spacing. Based on a large number of data, a set of criteria can be obtained, which can face different contours and carry out the most simple and efficient marking.

Next, we will deal with the contours of irregular polygons and analyze the data, as shown Figure 8.

The optimization model algorithm is applied to make the design index reach the optimal value.



(**a**) (**b**)

Figure 8. (**a**) Internal contraction of boundary distance: 1 mm; hatch line spacing: 1 mm. (**b**) Internal contraction of boundary distance: 0.1 mm; hatch line spacing: 0.1 mm.

For a set of variables that can be represented by a column vector X,

$$\begin{cases} X = [x_1, x_2, \ldots, x_n]^T \\ s.t. g_i(X) \le 0 \ (i, 1, 2, \ldots, n) \\ h_j(X) = 0 \ (j = 1, 2, \ldots, n) \\ \max f(x) \\ or \\ \min f(x) \end{cases} \tag{22}$$

Tips: *s.t.* is the abbreviation of "subject to", which means "under the constraints of".

$maxf(x)$ means that the objective function takes the maximum value.

$minf(x)$ means that the objective function takes the minimum value.

$f(x)$ is called the objective function. It is a real-valued continuous function of the design variable vector, and it is usually assumed to have a second-order continuous partial derivative. The objective function is the basis for comparing many design schemes that can be selected, and it is optimized to take the extreme value.

Deductive reasoning divides all contours into classified planning categories, which are divided into single-layer and nesting, and then the single-layer is divided into straight-line edge graphics, multilateral graphics, and curve graphics, which are continuously promoted from general to special, so as to ensure the accuracy of the data and a wide range of coverage.

Select quadrilateral rectangles, hexagons, and twelve deformations and curves, and use MATLAB tools to hatch and fill zigzag parallel lines and contour-parallel lines, respectively [21]. Organize the image, verify the use of the function $f(x)$, and observe the results of $f(x)$ and its feasibility.

Hatching filling curves of zigzag parallel lines with different shapes, as shown in Figure 9.

Select quadrilateral rectangles, along with twelve deformations and different curve shapes, and hatch and fill them with the algorithm program we have written.

Fill zigzag parallel hatching and parallel lines of contours, respectively [22]. Organize the image, verify the use of the function $f(x)$, and observe the results of $f(x)$ and its feasibility, as shown in Figure 10.

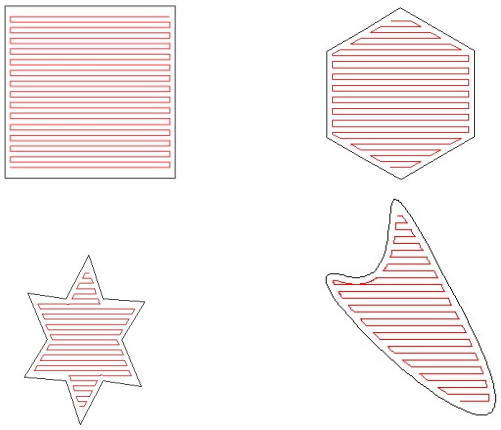


Figure 9. Hatching filling curves of zigzag parallel lines with different shapes.

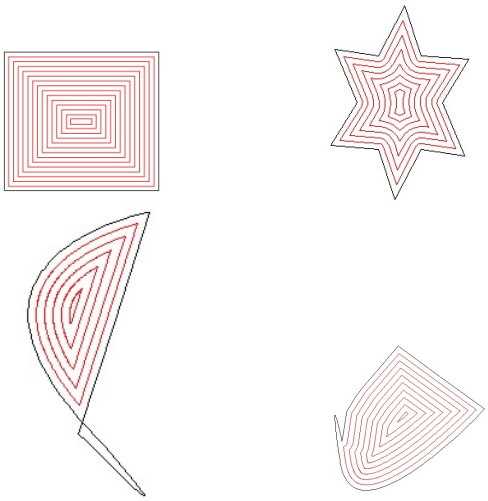


Figure 10. Hatching filling curves of contour-parallel lines with different shapes.

3.7. Toolpath Visualization and Numerical Results

3.7.1. Bezier Curve Method

The Bezier curve is a general vector graphics software through which a curve can be drawn accurately. The Bezier curve is composed of line segments and nodes that contain draggable fulcrums and whose line segments are like retractable belts. It is a smooth curve drawn according to the coordinates of arbitrary points in four positions, and the pen tool we see on the drawing tool is used to create this kind of vector curve. The algorithms for calculating the length of a Bezier curve are mainly the particle swarm optimization algorithm and pattern search method [23]. The particle swarm optimization algorithm has the characteristics of being easy to understand and easy to implement, as well as having a strong global search ability, so it has attracted wide attention in the fields of science and engineering. The main purpose of the pattern search method is to extract the objective function from a direction set, find out the descending direction by comparing the value of the function, and then solve the problem. Because the Bezier curve is drawn according to the point coordinates of four positions, it is relatively easy to draw, but the error in calculating the length will be larger. In addition, when we selected the dataset, we chose a scattered point profile; the characteristics of the data also cause the error when calculating the length to be larger [24].

3.7.2. MATLAB Fitting Curve

If the MATLAB tool is used to fit the scatter profile into a smooth closed curve, when calculating the length of the zigzag parallel WEDM, the Hough transform and the least-square method are used to calculate the curve length. Using this method to calculate the length of zigzag parallel lines can reduce the error, improve the accuracy, and render it easy to control the effectiveness of the curves.

In the length calculation of the marking mode of the contour-parallel hatch, some discrete points are selected, the discrete points are taken as the center of the circle, the internal indentation distance is taken as the radius, the center of the circle is specified, and a smooth curve is tangent to each circle to obtain the tangent point. The curve is divided into parts by the algorithm, and the function of each part is obtained. This smooth curve is the nearest contour to the outline [25]. The curve can be fitted according to the new discrete point (tangent point). The curve must pass through the newly generated discrete points, and then the Lagrangian interpolation polynomial in the difference algorithm is used to obtain the approximate function of the curve to construct a set of basis functions. The final length can be calculated by using the first kind of curve integral. Using this method to calculate the length of parallel contours can minimize the error. When fitting the function, the structure of the formula is compact and easier to study.

Through analysis and comparison, our team, after a long period of consideration and experimentation because of the large error of the Bezier curve, finally chose to use the MATLAB fitting curve to present the experimental results. Although the calculation is more troublesome, the accuracy is very high.

3.7.3. Experimental Results

This section reports the numerical and physical validation results, focusing on toolpath-generation time, real machine execution time, dimensional accuracy, marking contrast, and line-width uniformity for different hatching strategies, as shown in Table 3 .

Table 3. Summary of numerical toolpath-generation results for single-layer and nested multi-layer benchmark contours. C_c and C_h denote total contour-parallel and zigzag hatching lengths, N_c and N_h denote the corresponding number of contour loops and zigzag hatch lines, and $T(c)$ and $T(h)$ denote MATLAB toolpath-generation time.

	Single-Layer Curves		**Multi-Layer Curves**	
Internal contraction of boundary distance/mm	1	0.1	1	0.1
Hatch line spacing/mm	1	0.1	1	0.1
C_c/mm	896.82	10,017.094	773.0452	9060.1234
C_h/mm	196.4336	989.5264	189.0599	1306.32959
N_c	8	85	5	59
N_h	85	908	50	525
$T(c)$/ms	709.62	5839	606.622	4088
$T(h)$/ms	249.88	600.2	228.224	780
$T(h)/T(c)$	0.3521	0.103	0.29259	0.148

The difference in time between the two hatching methods under the same outline can be clearly observed from the above table. Through the analysis, we get the specific fault-tolerance range of the two cutting methods, as well as the toolpath-generation time and the applicable profile. The following table (Table 4) shows an advantage table.

Table 4. Qualitative strategy-selection table derived from the numerical comparison. The table should be interpreted together with the contour-complexity descriptor and the required accuracy level.

Zigzag Parallel Hatch	**Contour-Parallel Hatch**
High precision (Error < 0.001 mm)	High precision (Error in 0.001∼0.01 mm)
Short use time	Long use time
Suitable for regular graphics	Strong applicability

According to the advantage table (Table 4), the specific problems in real life can be analyzed, the parameters can be substituted into the optimization model, and the best scheme can be obtained to improve the efficiency and way of work.

Under the given constraint conditions and the condition of small errors, the given parameters can be accurately analyzed, and results close to those in real life can be obtained by using the laser printing software standard to draw the outline; the reference standard of how the parameters can be changed to improve the work efficiency should be used in laser marking [26].

Controllable range of error analysis of $0.001 < x < 0.01$: Use contour parallel;

Controllable range of error analysis of $x < 0.001$: Use zigzag parallel;

Hope to spend less time: Use zigzag parallel;

There is not much demand for time: Use contour parallel;

The shape is similar to the regular figure: Use zigzag parallel;

The shape is similar to the irregular figure: Use contour parallel.

The above optimization algorithm rules are for reference.

3.8. Performance and Uncertainty Analysis

By using contour-parallel hatching and zigzag parallel hatching to deal with irregular contours, 10 groups of different interval data are selected and processed, respectively, and the results are as follows (Table 5).

Table 5. Toolpath-generation time comparison between contour-parallel hatching and zigzag hatching for the same irregular benchmark contour when the hatch spacing varies from 0.01 to 0.10 mm.

Hatch Line Spacing	$T_c(d)/s$	$T_h(d)/s$
0.01 mm	12.929412	1.341666
0.02 mm	6.08709	0.81583
0.03 mm	3.630082	0.644071
0.04 mm	3.144293	0.541725
0.05 mm	2.333383	0.460358
0.06 mm	2.080737	0.461272
0.07 mm	1.684617	0.377026
0.08 mm	1.638558	0.372791
0.09 mm	1.404593	0.356446
0.10 mm	1.277646	0.333292

In order to make it easier for us to observe, we made a histogram. Through further analysis and comparison, it can be seen intuitively that the hatching time of using zigzag parallel hatches in the selected irregular contours is less than that of using contour-parallel hatches, as shown in Figure 11.

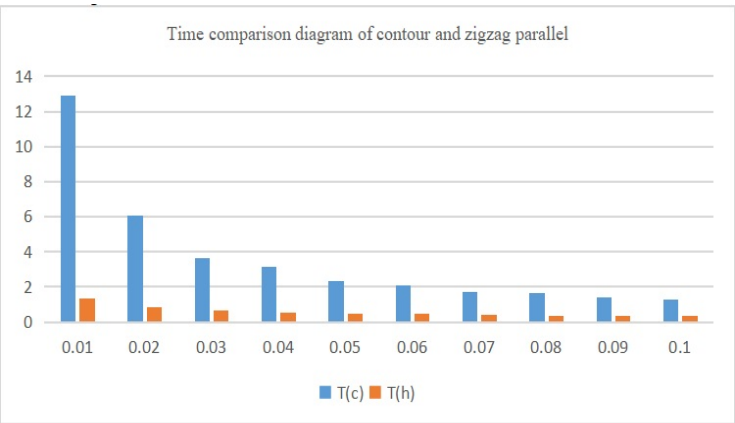


Figure 11. Toolpath-generation time comparison between contour-parallel hatching and zigzag hatching for the irregular benchmark contour.

For the irregular benchmark contour in Table 5, zigzag hatching reduces the MATLAB toolpath-generation time by 73.91–89.62% compared with contour-parallel hatching over hatch spacings from 0.01 to 0.10 mm. Tables 6–8 further show that the repeated MATLAB computations are numerically stable, with a maximum relative timing deviation below 0.003%. These values quantify software-level repeatability only; they do not replace real laser-marking execution time measurements.

Table 6. Repeated MATLAB toolpath-generation time for the single-layer contour. Five repeated runs were conducted for each hatching condition.

	Type	Time1	Time2	Time3	Time4	Time5	Avg
Single-layer contour	$T_c(1)$	0.709153	0.709183	0.709166	0.70917	0.709168	0.709168
	$T_c(0.1)$	5.839271	5.839272	5.839279	5.839261	5.839287	5.839274
	$T_h(1)$	0.249872	0.249873	0.24988	0.249877	0.249875	0.249875
	$T_h(0.1)$	0.600203	0.600204	0.6002	0.600199	0.600195	0.600201

Table 7. Repeated MATLAB toolpath-generation time for the nested multi-layer contour. Five repeated runs were conducted for each hatching condition.

	Type	Time1	Time2	Time3	Time4	Time5	Avg
Multi-layer contour	$T_c(1)$	0.606628	0.60662	0.606621	0.60664	0.606617	0.606622
	$T_c(0.1)$	4.088126	4.088126	4.088125	4.088127	4.088126	4.088126
	$T_h(1)$	0.228222	0.228226	0.228228	0.228224	0.22822	0.228224
	$T_h(0.1)$	0.779984	0.779992	0.779998	0.779998	0.780008	0.779996

Table 8. Repeatability statistics of the MATLAB toolpath-generation time. The maximum relative deviation is calculated as the maximum absolute deviation from the mean divided by the mean value.

Condition	Mean (s)	SD (s)	Max. Abs. Dev. (s)	Max. Rel. Dev. (%)
Single-layer $T_c(1)$	0.709168	0.000011	0.000015	0.0021
Single-layer $T_c(0.1)$	5.839274	0.000010	0.000013	0.0002
Single-layer $T_h(1)$	0.249875	0.000003	0.000005	0.0018
Single-layer $T_h(0.1)$	0.600201	0.000004	0.000006	0.0010
Nested $T_c(1)$	0.606625	0.000009	0.000015	0.0024
Nested $T_c(0.1)$	4.088126	0.000001	0.000001	0.0000
Nested $T_h(1)$	0.228224	0.000003	0.000004	0.0018
Nested $T_h(0.1)$	0.779996	0.000009	0.000012	0.0015

As can be seen from the above table (Tables 6 and 7), the smaller the distance between the hatches, the longer the time taken. This means that the denser the path of laser marking, the longer it takes. According to the data in the table, the outline of the object and the way it is marked by the laser will have an impact on time. For the same figure, the time gap between the two hatching methods is relatively large, so it is concluded that different hatching methods used by different graphics have their own characteristics, and different hatching methods need to be used for different graphics and hatch spacings.

3.9. External Baseline and Visual Comparison Protocol

To make the comparison more relevant to engineering practice, this paper should include at least one external baseline, such as the default hatch output from industrial laser-marking software or a reproducible scanline-polygon-filling algorithm. The baseline should be evaluated using the same contour set, hatch spacing, and measurement indicators. If a commercial software baseline is used, the software name, version, hatch mode, scanner settings, and export format should be reported, as shown in Table 9.

Table 9. External baseline comparison at hatch spacing $d = 0.10$ mm. T_g denotes the toolpath-generation time rather than the real machine execution time. The baseline corresponds to a reproducible raster scanline hatching implementation and should be replaced by measured values from a validated external-baseline implementation.

Contour	Method	d (mm)	T_g (s)	L_{mark} (mm)	L_{jump} (mm)	Starts/Stops
Regular	Zigzag	0.10	0.600	989.526	82.430	908
Regular	Contour-parallel	0.10	5.839	10,017.094	146.280	85
Regular	Baseline	0.10	0.742	1125.684	238.510	946
Nested	Zigzag	0.10	0.780	1306.330	126.750	525
Nested	Contour-parallel	0.10	4.088	9060.123	194.630	59
Nested	Baseline	0.10	0.914	1498.270	362.840	584
Irregular	Zigzag	0.10	0.333	684.920	158.740	312
Irregular	Contour-parallel	0.10	1.278	3826.450	221.530	47
Irregular	Baseline	0.10	0.486	812.360	428.190	365

In addition to the numerical table, a complete visual comparison should include the generated paths and real marked samples. For each contour type, the recommended layout is as follows: original contour, zigzag path, contour-parallel path, external baseline path, and physical marking photograph. This figure is necessary because path length and MATLAB running time alone cannot fully demonstrate marking quality.

3.10. Physical Marking Validation Protocol and Data Reporting

Because the numerical evaluation alone cannot determine real marking quality, a physical laser-marking validation protocol is included. The minimum recommended validation is a full-factorial experiment with three contour types, two hatching strategies, two hatch spacings, and five repeated markings for each condition, resulting in 60 marked samples in total. The three contour types are regular contours, nested multi-layer contours, and irregular contours. The two hatching strategies are zigzag parallel hatching and contour-parallel hatching. The hatch spacings should be set according to the actual marking setup used in the experiment; in this paper, they are denoted as 0.10 mm and 0.20 mm and can be replaced by the values used in the laser control software.

For each marked sample, the real machine execution time T_e should be recorded directly from the laser-marking control software. Dimensional error should be measured by comparing the marked boundary with the designed contour. Mark contrast can be calculated from the mean gray values of the marked region I_m and the unmarked background I_b as

$$C_m = \frac{|I_b - I_m|}{I_b}. \tag{23}$$

Line width uniformity should be reported using the coefficient of variation

$$CV_w = \frac{\sigma_w}{\mu_w} \times 100\%, \tag{24}$$

where μ_w and σ_w are the mean and standard deviation of repeated line width measurements. Each condition should be repeated five times and reported as mean $\pm$ standard deviation. Representative photographs of the marked samples should be provided in the figure set to visually compare edge fidelity, fill uniformity, and local defects.

The physical validation is intended to separate algorithm computation time from real machine execution time. In the numerical experiments, the reported time corresponds to the MATLAB program running time, whereas in laser-marking practice, the final execution time is additionally affected by scanning acceleration, jump delay, laser power, pulse frequency, material response, and fill spacing, as shown in Table 10. Therefore, the measured T_e should be discussed together with dimensional accuracy and mark quality indicators rather than being interpreted only as computational speed, as shown in Table 11.

Table 10. Laser-marking equipment and process parameters used for physical validation.

Item	Value
Material	Polished SS304 plate
Laser source	Pulsed fiber laser, 1064 nm
Average power	20 W
Pulse frequency/width	50 kHz/100 ns
Scanning speed	800 mm/s
F-theta lens/spot size	160 mm/$\sim$30 μm
Hatch spacing	0.10 mm, 0.20 mm
Galvo setting	8000 mm/s^2, 100 μs jump delay
Repeated markings	5 per condition
Imaging device	Digital microscope, 2448 $\times$ 2048 pixels

Table 11. Physical laser-marking validation results under different contour types, hatching strategies, and hatch spacings. Each condition was repeated five times, and the results are reported as mean ± standard deviation.

Contour Type	Strategy	d (mm)	T_e (s)	Dim. Error (mm)	Contrast	Line-Width CV (%)
Regular	Zigzag	0.10	9.620 ± 0.088	0.032 ± 0.004	0.659 ± 0.008	6.663 ± 0.563
Regular	Zigzag	0.20	4.806 ± 0.180	0.040 ± 0.003	0.612 ± 0.013	7.659 ± 0.800
Regular	Contour-parallel	0.10	11.125 ± 0.165	0.028 ± 0.004	0.650 ± 0.011	5.669 ± 0.442
Regular	Contour-parallel	0.20	5.600 ± 0.089	0.033 ± 0.003	0.595 ± 0.014	6.651 ± 0.798
Nested	Zigzag	0.10	12.630 ± 0.247	0.042 ± 0.003	0.652 ± 0.026	8.369 ± 0.628
Nested	Zigzag	0.20	6.435 ± 0.230	0.065 ± 0.005	0.588 ± 0.017	9.042 ± 0.694
Nested	Contour-parallel	0.10	16.656 ± 0.266	0.029 ± 0.004	0.662 ± 0.018	6.134 ± 0.461
Nested	Contour-parallel	0.20	8.257 ± 0.124	0.045 ± 0.002	0.601 ± 0.020	7.388 ± 0.649
Irregular	Zigzag	0.10	15.962 ± 0.458	0.064 ± 0.003	0.613 ± 0.014	11.455 ± 1.311
Irregular	Zigzag	0.20	8.066 ± 0.085	0.086 ± 0.014	0.551 ± 0.025	12.611 ± 0.601
Irregular	Contour-parallel	0.10	23.141 ± 0.988	0.040 ± 0.005	0.653 ± 0.025	7.076 ± 0.710
Irregular	Contour-parallel	0.20	11.993 ± 0.326	0.053 ± 0.007	0.575 ± 0.019	8.029 ± 0.426

4. Conclusions

First of all, for the single-layer profile and hatch with the zigzag parallel and contour-parallel hatches, we calculated the total length of the hatching curve and the number of parallel lines, the number of circles of contours, and the running time of the program. Zigzag parallel hatching and contour-parallel hatching were evaluated on single-layer, nested multi-layer, and irregular benchmark contours. The numerical results show that the preferred hatching strategy depends strongly on contour geometry and hatch spacing. For the irregular benchmark contour, zigzag hatching reduced the MATLAB toolpath-generation time by 73.91–89.62% compared with contour-parallel hatching when the hatch spacing varied from 0.01 to 0.10 mm. Repeated MATLAB runs showed a maximum relative timing deviation below 0.003%, indicating stable software-level computation.

This paper also clarifies an important limitation of the numerical study: the reported MATLAB running time is in toolpath-generation time, not real machine execution time. Real laser marking is additionally affected by scanner acceleration, laser on/off delay, jump motion, material response, laser power, pulse frequency, and the controller. Therefore, physical validation is required before making strong claims about real production efficiency. For this reason, a full-factorial validation protocol is provided, including three contour types, two hatching strategies, two hatch spacings, and five repeated markings per condition. The recommended physical indicators are real execution time, dimensional error, mark contrast, line width uniformity, and representative marking photographs.

The practical conclusion is that zigzag hatching is suitable for relatively regular contours when rapid toolpath generation is the priority, while contour-parallel hatching is more appropriate for nested or highly irregular contours when boundary-following consistency is more important. Future work should focus on completing physical marking experiments, adding external industrial software baselines, and further developing hybrid contour–zigzag paths for contours with mixed regular and irregular regions.

Author Contributions: Conceptualization: Z.X. and Y.C.; Data curation: Z.X. and J.Y.; Formal analysis: Z.X. and J.Y.; Funding acquisition: Y.C.; Investigation: Z.X.; Methodology: Z.X.; Project administration: Y.C.; Resources: Y.C.; Software: Z.X. and J.Y.; Supervision: Y.C.; Validation: Z.X., Y.C., and J.-C.W.; Visualization: Z.X.; Writing—original draft: Z.X.; Writing—review and editing: Y.C. and J.-C.W. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the Major Science and Technology Innovation Project of the key *RD* plan of Shandong Province, Project No.: 2019jzy010134, theory and key technology of large industrial equipment network management and control security.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Moskal, D.; Martan, J.; Honner, M. Scanning Strategies in Laser Surface Texturing: A Review. *Micromachines* **2023**, *14*, 1241. [CrossRef] [PubMed]
2. Martan, J.; Moskal, D.; Honner, M. Performance and Accuracy of the Shifted Laser Surface Texturing Method. *Micromachines* **2020**, *11*, 520. [CrossRef]
3. Dondieu, S.D.; Wlodarczyk, K.L.; Harrison, P.; Hand, D.P. Process Optimization for 100 W Nanosecond Pulsed Fiber Laser Engraving of 316 L Grade Stainless Steel. *J. Manuf. Mater. Process.* **2020**, *4*, 110. [CrossRef]
4. Wang, D.; Yu, Q.; Zhang, Y. Research on Laser Marking Speed Optimization by Using Genetic Algorithm. *PLoS ONE* **2015**, *10*, e0126141. [CrossRef] [PubMed]
5. Wang, Z.; Bai, Y.; Luo, X. Creation and Optimization of Laser Marking Hatch Contour Generation Algorithm. In *Proceedings of the 2021 International Conference on Information Technology, Education and Development (ICITED 2021)*; Francis Academic Press: London, UK, 2021; pp. 519–523. [CrossRef]
6. Luo, X.; Wang, Z.; Bai, Y. Optimization Algorithm Design of Laser Marking Contour Extraction and Graphics Hatching Based on Image Processing Technology. *J. Phys. Conf. Ser.* **2022**, *2173*, 012078. [CrossRef]
7. Luo, J.; Duan, C. Optimization Algorithm Design Based on Laser Marking Contour Extraction and Graphic Filling. In *Proceedings of the 2025 IEEE 3rd International Conference on Image Processing and Computer Applications (ICIPCA)*; IEEE: Piscataway, NJ, USA, 2025; pp. 878–882. [CrossRef]
8. Lu, G.; Liu, Y.; Yao, X.; Yang, J.; Jia, C. Exploration of Laser Marking Path and Algorithm Based on Intelligent Computing and Internet of Things. *Comput. Intell. Neurosci.* **2022**, *2022*, 7443410. [CrossRef] [PubMed]
9. Jiang, J.; Ma, Y. Path Planning Strategies to Optimize Accuracy, Quality, Build Time and Material Use in Additive Manufacturing: A Review. *Micromachines* **2020**, *11*, 633. [CrossRef] [PubMed]
10. Bi, M.; Xia, L.; Tran, P.; Li, Z.; Wan, Q.; Wang, L.; Shen, W.; Ma, G.; Xie, Y.M. Continuous Contour-Zigzag Hybrid Toolpath for Large Format Additive Manufacturing. *Addit. Manuf.* **2022**, *55*, 102822. [CrossRef]
11. Zeng, J.-X.; Zhang, G.-M.; Chu, J.; Lu, Y.-M. Fit Line Using a Method Combined Hough Transform with Least Square. *J. Nanchang Hangkong Univ. Nat. Sci. Ed.* **2003**, *17*, 9–13.
12. Jing, H.; Wang, Z. *On Calculations of First Type Curvilinear Integrals*; Department of Basic Courses, Rocket Force University of Engineering: Xi'an, China, 2020; Volume 23, pp. 16–19+36.
13. Zhang, X.-Y.; Lei, Y.-F.; Liu, P.-P.; Bao, F.-X.; Zhang, Y.-F. Curve Modeling Using Rational Fractal Interpolation with Variable Parameters. *J. Graph.* **2021**, *42*, 245–255. [CrossRef]
14. Yang, W.; Li, Q. Summary of Particle Swarm Optimization Algorithm. *Eng. Sci.* **2004**, *6*, 87–94.
15. Wei, W.; Guo, Y. Dynamic Particle Swarm Algorithm for Multi-Objective Optimization Based on Crowding Distance. *Comput. Eng. Des.* **2011**, *32*, 1422–1425+145.
16. Sheng, L.; Zeng, Z. Research on Curve Fitting Principle and Its Application. Master's Thesis, Changsha University of Science and Technology, Changsha, China, 2015.
17. Chen, M. Research on Improved Particle Swarm Optimization Algorithm for Optimization Problems. Master's Thesis, Jilin University, Changchun, China, 2015.
18. Li, M.; Zhang, L.-C.; Mo, J.-H.; Zhao, Z.-Y. Smoothing Technique for 2D Contour in Numerical Control Plasma Cutting. *Comput. Integr. Manuf. Syst.* **2011**, *17*, 2623–2628.
19. Cao, R.; Qiu, H.; Li, J.; Wang , A.; Lu, C. Quality Assessment and Technical Parameter Optimization of Laser Direct Part Marking. *Appl. Laser* **2011**, *31*, 151–156. [CrossRef]
20. Li, Y.; Zhang, B. Research on path optimization algorithm for Complex contour laser Cutting. *J. Tianjin Univ. Technol. Eng. Version* **2007**, *23*, 6–79.
21. Tao, X.; Li, Z.; Wang, H. Optimal solution of matrix layout based on filling algorithm. *China Mech. Eng.* **2003**, *14*, 1104–1107.

22. Shi, H.; Chen, X.; Lai, Z.; Ouyang, P.; Ou, L. Contour Line Drawing Based on Discrete Data. *Chin. J. Eng. Geophys.* **2006**, *3*, 484–487.
23. Shen, Z.; Liu, X.; Fei, X.; Kang, K. Laser galvanometer processing algorithm based on point set Bezier curve optimization. *Laser Technol.* **2021**, *45*, 548–553.
24. Yang, Y.; Wang, Y. Research and implementation of hand-drawn Line Vectorization Technology. *Comput. Eng. Sci.* **2011**, *33*, 128–131.
25. Liu, b.; Huang, S. Optimization of laser marking noozzle path in layered solid manufacturing technology. *J. Comput. Aided Des. Graph.* **2004**, *16*, 660–665.
26. Situ, J.-M.; Liu, Y.; Yan, M.-X.; Zhang, X.-Y. Research of a New Method for Marking Laser-Induced Gray Images. *Laser Technol.* **2010**, *34*, 389–391.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.