


Research Article

Exploration of Laser Marking Path and Algorithm Based on Intelligent Computing and Internet of Things

Gang Lu ¹, Yide Liu,¹ Xue Yao,¹ Jiachen Yang,² and Cheng Jia²

¹Department of Electrical Engineering & Information Technology, Shandong University of Science and Technology, Jinan, Shandong, China

²Swinburne College, Shandong University of Science and Technology, Jinan, Shandong, China

Correspondence should be addressed to Gang Lu; 201903204415@sdust.edu.cn

Received 28 April 2022; Accepted 31 May 2022; Published 24 June 2022

Academic Editor: Arpit Bhardwaj

Copyright © 2022 Gang Lu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Nowadays, laser processing technology is being used more and more in various fields, and the requirements for laser control procedures are getting higher and higher. This paper aims to study the path generation problem of laser marking technology in order to improve the efficiency of laser marking as well as the protection of the marking material. Therefore, we creatively propose two-path generation methods, namely, sawtooth parallel and contour parallel, and design the boundary curve offset algorithm and domain partition intersection algorithm for the computer simulation of the two marking paths, respectively. Through the simulation, we discussed the efficiency and marking quality of the two path generation methods and gave the conclusion that the efficiency of the sawtooth parallel path generation method is greater than that of the contour parallel path generation method under specific parameters.

1. Introduction

The laser [1] is an important invention of the 20th century and has been called “the sharpest knife,” “the most accurate ruler,” and “the most unusual light.” Lasers have been increasingly used in industrial processes for a variety of machining operations such as marking, welding, drilling, cutting, heat treating, and painting. Laser marking [2] is the marking of logos, characters, symbols, and images with a laser on the product surface. It is a widely used processing method with the advantages of high processing efficiency, noncontact operation, no consumables, and low impact on product surface deformation [3] and solid marking content. The hatch tool of a laser marking machine can be used for hatch-specified 2D composite profile [4], but the setting of different hatch parameters has a great impact on the processing effect of different materials. Zigzag parallel [5] hatching and profile parallel [6] hatching are the two basic ways of marking road dynamics generation. And the design of different process processing and hatching algorithms has substantial significance on whether the operation efficiency

and accuracy of laser marking machine [7] can be improved.

Nowadays, in the field of laser marking, more research lies in the combination selection and optimization of process parameters [8]. Shivakoti [9] et al. investigated the selection of optimal laser beam micromarking process parameters using the fuzzy TOPSIS [10] method in the GaN laser beam [11] marking process and concluded that small pulse frequency [12], high current, and scanning speed lead to increased mark intensity. Some people have explored this area through Bessel curves [13]. And a connected Fermat spiral area filling algorithm (CFS) [14] has also been proposed, but its study has not been deeply applied to laser marking technology and cannot be applied for complex graphics [15]. Our research aims to fill this gap in the laser marking path generation algorithm.

To fill this gap, in order to fill this gap, we explore the laser marking path generation algorithm from the perspective of improving the efficiency of laser marking path generation, combining computer graphics [16] principles with the length of the laser marking path and the time of the generation algorithm as the main factors. Firstly, we

propose two-path generation methods, namely direction-parallel hatching and contour-parallel hatching. Parallel directional shaded lines, also called “zigzag” shaded lines, have paths that move along line segments parallel to the initially selected reference direction. The two marking path forms are shown in Figure 1. Based on this strategy, connected paths are obtained by connecting these parallel line segments so that they either all cross from right to left (or left to right) or alternate between left to right and right to left. In contrast, contour-parallel shading uses offset line [17, 18] segments based on boundary curves as smooth shading paths similar to boundary curves. Thus, contour-parallel shadows can be generated in a spiral fashion along a curve at a constant distance from the curve boundary. Later, the domain partition intersection algorithm and boundary curve offset algorithm are proposed for both methods and verified the feasibility and generation efficiency of both algorithms by computer simulation [19].

2. Method

In exploring the problem of laser marking path generation, we proposed the zigzag parallel and contour parallel [20] filling path approaches and used interpolation methods [21] to fit the original pattern at a high level to enhance the marking pattern.

Usually, the first function in an M-file is the main function, which can be followed by any number of subfunctions. The main function can be called by other functions outside that file, and the main function is called by the filename of the M file where the function is stored.

M files can include multiple functions, and functions other than the main function are called subfunctions. Subfunctions can only be called by the main function or by other subfunctions within that file. Each subfunction begins with a function definition statement and continues until the definition of the next function or the end of the file. The subfunctions appear in any order, but the main function must appear first.

2.1. Original Graph Curve Fitting

2.1.1. Definition of Curve Fitting. The spline curve [22] $S(x)$ is a segmentally defined equation. Given $(n + 1)$ data points and a total of n intervals, the cubic spline equation satisfies the following conditions:

- (A) In each segment interval $[x_i, x_{i+1}]$ ($i = 0, 1, \dots, n - 1$, x increasing), $S_{(x)} = S_{i(x)}$ is a cubic polynomial
- (B) The following relationship is satisfied:

$$S_{i(x)} = y_i \quad (i = 0, 1, \dots, n). \quad (1)$$

- (C) $S(x)$, the derivative $S'(x)$, and the second-order derivative $S''(x)$ are all continuous in the interval $[a, b]$, that is, the $S(x)$ curve is smooth. So n cubic polynomial segments can be written as



FIGURE 1: Graphical representation of zigzag parallel and contour parallel hatch.

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad (2)$$

where $i = 0, 1, \dots, n - 1$, and a_i, b_i, c_i, d_i represent the $4n$ unknown coefficients.

2.1.2. Request a Solution. Interpolation and continuity:

$$\begin{aligned} S_i(x_i) &= y_i, \\ S_i(x_{i+1}) &= y_{i+1}, \end{aligned} \quad (3)$$

where $i = 0, 1, \dots, n - 1$.

Differential continuity:

$$\begin{aligned} S'_i(x_{i+1}) &= S''_{i+1}(x_{i+1}), \\ S'_i(x_{i+1}) &= S''_{i+1}(x_{i+1}), \end{aligned} \quad (4)$$

where $i = 0, 1, \dots, n - 2$.

Differential equation of a spline curve:

$$\begin{aligned} S_i(x) &= a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \\ S'_i(x) &= b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2, \\ S''_i(x) &= 2c_i + 6d_i(x - x_i). \end{aligned} \quad (5)$$

Bring the following step size h into the conditions of the spline curve:

$$h_i = x_{i+1} - x_i. \quad (6)$$

Thus, we can deduce

$$\begin{aligned} a_i &= y_i, \\ a_i + h_i b_i + h_i^2 c_i + h_i^3 d_i &= y_{i+1}, \\ b_i + 2h_i c_i + 3h_i^2 d_i - b_{i+1} &= 0, \\ 2c_i + 6h_i d_i - 2c_{i+1} &= 0. \end{aligned} \quad (7)$$

We assume that

$$m_i = S''_i(x_i) = 2c_i. \quad (8)$$

Thus, we can deduce the following result:

$$2c_i + 6h_i d_i - 2c_{i+1} = 0. \quad (9)$$

It is equivalent to the following equation.

$$m_i + 6h_i d_i - m_{i+1} = 0. \quad (10)$$

Thus, the following formula is derived as

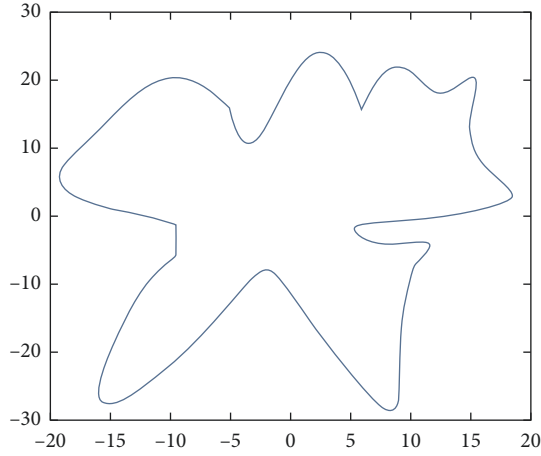


FIGURE 2: Simulation graphics.

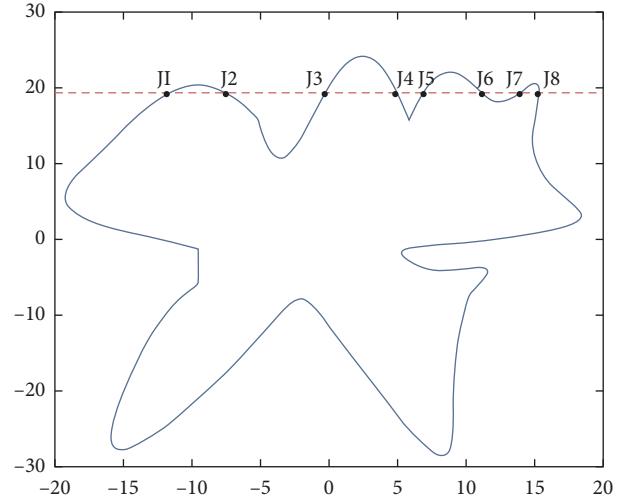


FIGURE 4: Scan for delivery.

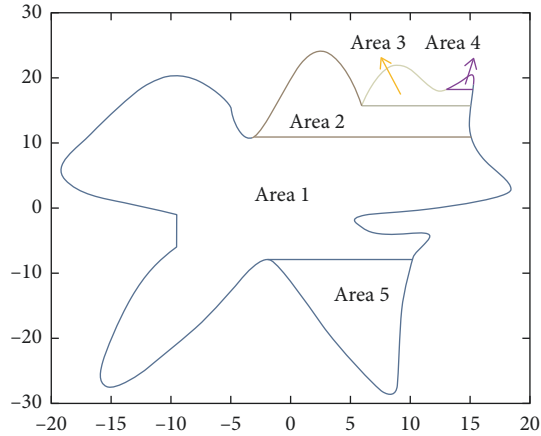


FIGURE 3: Domain zoning.

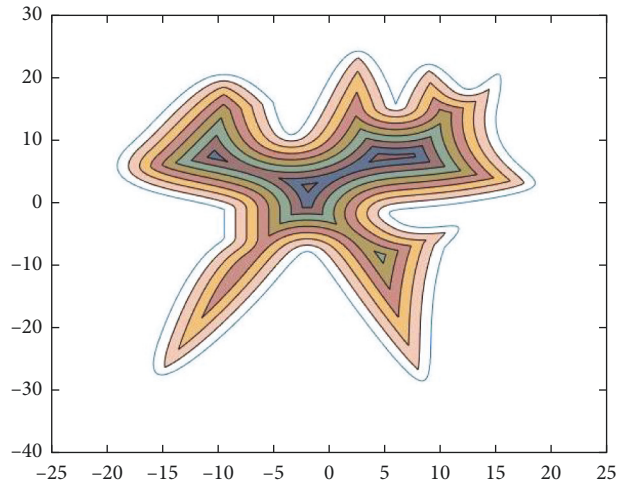


FIGURE 5: Results of simulation experiment I.

$$d_i = \frac{m_{i+1} - m_i}{6h_i}. \quad (11)$$

We substitute c_i, d_i into

$$y_i + h_i b_i + h_i^2 c_i + h_i^3 d_i = y_{i+1}, \quad (12)$$

we can derive the formula:

$$b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{2} m_i - \frac{h_i}{6} (m_{i+1} - m_i). \quad (13)$$

From the range of values of i , there are $(n-1)$ equations but $(n+1)$ unknowns m . To solve the system of equations, two additional equations are required. So some restrictions need to be placed on the differentiation of the two endpoints x_0 and x_n . Here, we use not-a-knot [23] to solve the problem.

Specify the cubic differential matching of the spline curve.

$$\begin{aligned} S_0'''(x_1) &= S_1'''(x_1), \\ S_{n-2}'''(x_{n-1}) &= S_{n-1}'''(x_{n-1}). \end{aligned} \quad (14)$$

After that we can derive the following conclusions:

$$\begin{aligned} h_1(m_1 - m_0) &= h_0(m_2 - m_1), \\ h_{n-1}(m_{n-1} - m_{n-2}) &= h_{n-2}(m_n - m_{n-1}). \end{aligned} \quad (15)$$

The new coefficient matrix of the system of equations can be written as

$$\begin{bmatrix} -h_1 & h_0 + h_1 & -h_0 & \cdots & \cdots & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & 0 & & \vdots \\ 0 & h_1 & 2(h_1 + h_2) & h_2 & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & -h_{n-1} & h_{n-2} + h_{n-1} & -h_{n-2} \end{bmatrix}. \quad (16)$$

TABLE 1: Experiment I algorithm analysis.

Function name	Number of calls	Total time (s)	Self-use (s)
createCanvas	1	0.007	0.002
GetInstance	1	0	0
hasBeenOpened	1	0.001	0.001
maybeShow	1	0.003	0.001
Polygon	30	0.039	0.017
doSetup	30	0.022	0.022
ToolBarController	1	0.005	0.004
Attributes	30	0.012	0.003
checkAttrs	30	0.001	0.001
checkClass	30	0.001	0.001
checkInputs	30	0.007	0.002
Axescheck	60	0.009	0.007
strcmpi ("parent", x)	60	0.002	0.001
Lunkuo	1	2.221	0.089
Cla	1	0.007	0.003
converStringToCharArgs	30	0.001	0.001
generateArgumentDescriptor	30	0.004	0.004
isCharOrString	90	0	0
Gobjects	92	0.013	0.009
claNotify	1	0.001	0.001
Clo	1	0.003	0.003
hasDisplay	1	0	0
Hold	30	0.023	0.014
isFigureShowEnabled	1	0	0
isPublishingTest	1	0	0
isStringScalar	60	0.001	0.001
markFigure	30	0.005	0.005
Newplot	31	0.066	0.041
Newplotwrapper	1	0.061	0.001
Nextstyle	30	0.01	0.01

2.2. *Introduction to Matlab Functions.* The polyshape function [24] creates a polygon defined by two-dimensional vertices and returns an object with attributes describing its vertices, solid regions, and holes.

The polybuffer function implements the creation of a buffer around a point, line, or polyshape object. Its boundary will be input to the polyshape object buffer polyin a distance d . For positive values of d , the solid area boundary polyin expands by unit d and the hole boundary shrinks by unit d . The negative values of d shrink the solid boundary and expand the hole boundary.

The Intersect function is the intersection of a polyshape object. It can return the intersection of a closed curve and a line, and determine which parts of the line are inside and outside the closed curve.

2.3. *Boundary Curve Offset Based on Contour Parallel Path.* Decompose the original figure into n simple figures, denote the centroid [25] of each simple figure, G_i represents the center point of each simple graph and the S_i represents

the area of each simple figure, yielding the central point coordinates of the original graph as

$$x = \frac{\sum_{i=1}^n G_{ix} S_i}{\sum_i^n S_i}, \quad (17)$$

$$y = \frac{\sum_{i=1}^n G_{iy} S_i}{\sum_i^n S_i}.$$

Thus, the maximum distance from any point (X, Y) on the road strength to the center of the path is obtained as:

$$d = \max\left(\sqrt{(X - x)^2 + (Y - y)^2}\right). \quad (18)$$

For the contour parallel line hatching problem, we use the buffer function polybuffer for computer simulation.

$$\text{polyout} = \text{polybuffer}(\text{polyin}, d). \quad (19)$$

Returns a polyshape object with a boundary that creates a buffer at distance d based on the input polyshape object polyin. For positive values of d , the solid area boundary of polyin is expanded by d units and the hole boundary is shrunk by d units. Negative values of d shrink the solid boundary and expand the hole boundary.

2.4. *Domain-Distinct Intersection Algorithm for Self-Intersecting Graphs Based on Zigzag Parallel Paths.* For the serrated parallel path, we propose a regionalized closed curve [26] and a straight-line intersection algorithm. Firstly the original graph is regionalized according to the concave and convex points. The original graphics and the zoning graphics are shown in Figures 2 and 3 respectively. Afterwards, using a straight line $y = a$ from the top of the closed graph to the bottom in 1 mm steps in order to scan [27], using the intersect function to find the coordinates of the entry, and then according to the previous cycle can always find the coordinates of the intersection point of the line with the original graph each time $y = a$.

According to the above division, the region will be divided into many small blocks of all data points, each small block with an array to store the first (from left to right) of the intersection of the line, and then an array to store the second, because it is a sawtooth arrangement, so it must be the first from a small block to point to the second, and so on, the intersection of each small block will be the first from the small block of coordinates to point to the second coordinates.

The X and Y values of the coordinates of the intersection point, the next in each array position relationship will be the X and Y values of the coordinates of the intersection point of the next line and the intersection point of the line at the coordinates of the intersection point of

TABLE 2: Experiment II algorithm analysis.

Function name	Number of calls	Total time (s)	Self-use time (s)
createCanvas	1	0.007	0.002
GetInstance	1	0	0
hasBeenOpened	1	0.001	0.001
maybeShow	1	0.003	0.001
Polygon	314	0.309	0.139
doSetup	314	0.177	0.17
ToolBarController	1	0.005	0.004
Attributes	314	0.042	0.009
checkAttrs	314	0.002	0.002
checkClass	314	0.008	0.008
checkInputs	314	0.023	0.01
Axescheck	628	0.057	0.048
strcmpi ("parent", x)	628	0.009	0.006
Lunkuo	1	20.474	0.047
Cl	1	0.007	0.002
converStringToCharArgs	314	0.004	0.004
generateArgumentDescriptor	314	0.013	0.012
isCharOrString	942	0.002	0.002
Gobjects	942	0.077	0.059
claNotify	1	0.001	0.001
Clo	1	0.003	0.003
hasDisplay	1	0	0
Hold	314	0.181	0.112
isFigureShowEnabled	1	0	0
isPublishingTest	1	0	0
isStringScalar	628	0.003	0.003
markFigure	314	0.039	0.039
Newplot	315	0.147	0.067
newplot > ObserveAxesNextPlot	315	0.027	0.021

TABLE 3: Comparison of results of contour parallel experiments.

Experimental group	Time(s)					Average time	Number of laps	Average time ratio	Circumference
Number of experiment	1	2	3	4	5				
1	1.937	1.952	1.995	1.959	2.103	1.989	11	10.436	896.820
2	20.583	20.534	20.51	21.169	21.002	20.759	110		10017

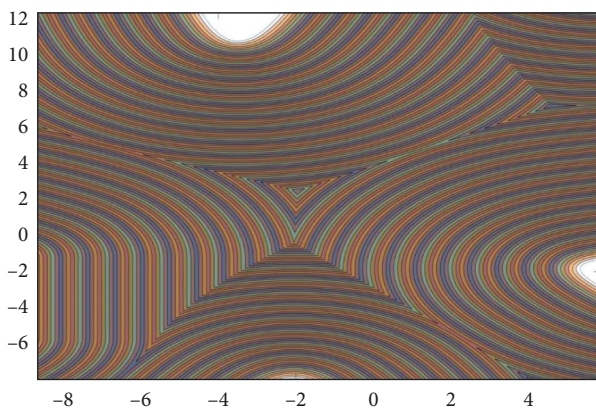


FIGURE 6: Simulation experiment II partial enlargement.

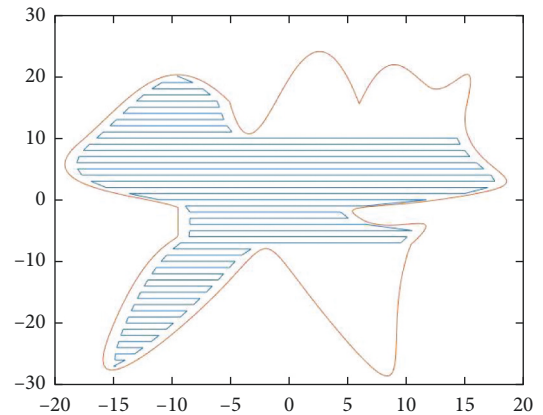


FIGURE 7: Region 1.

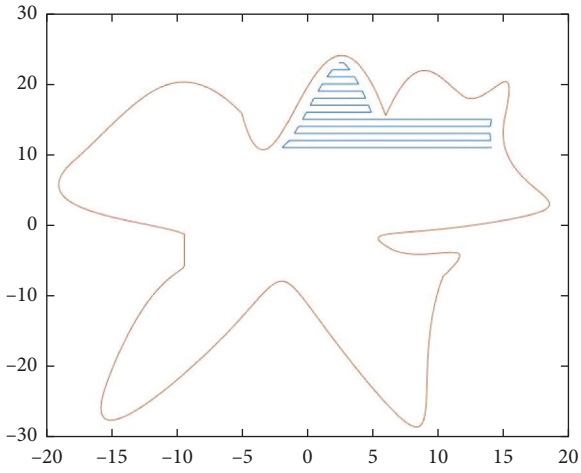


FIGURE 8: Region 2.

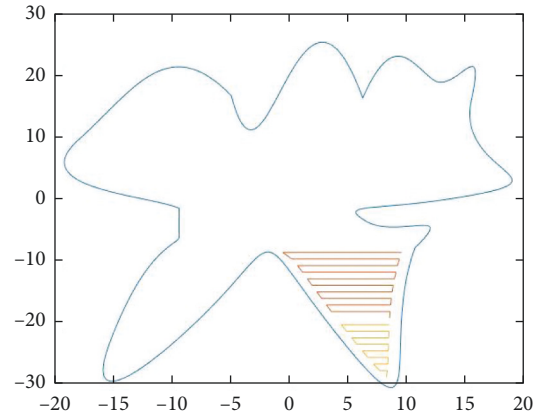


FIGURE 11: Region 5.

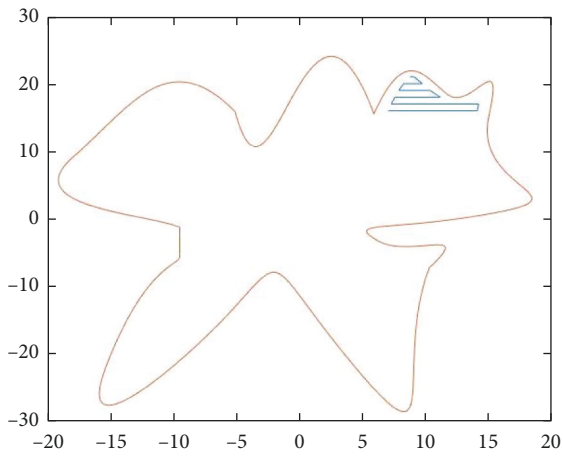


FIGURE 9: Region 3.

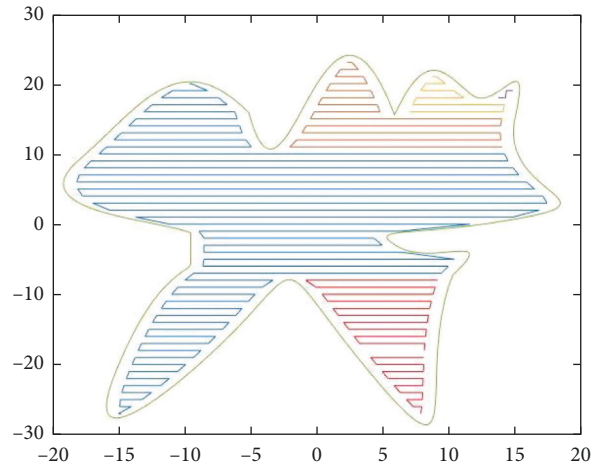


FIGURE 12: Zigzag parallel experiment.

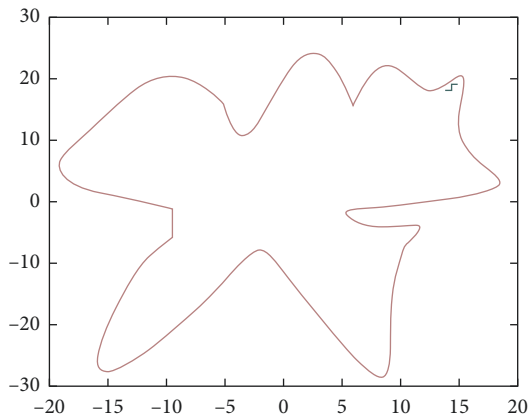


FIGURE 10: Region 4.

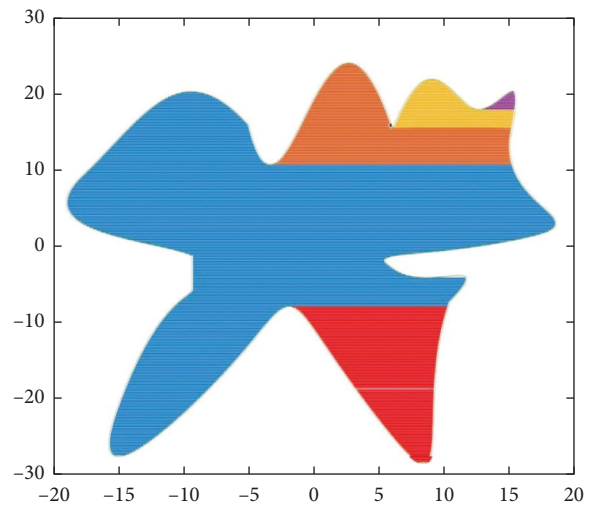


FIGURE 13: Zigzag parallel experiment II.

TABLE 4: Algorithm analysis based on zigzag parallel path simulation experiment I.

Function name	Number of calls	Total time (s)	Self-use time (s)
createCanvas	1	0.007	0.002
getInstance	1	0	0
hasBeenOpened	1	0.001	0.001
maybeShow	1	0.003	0.001
ToolBarController	1	0.005	0.004
axescheck	6	0.002	0.002
strcmpi ("parent", x)	6	0.001	0.001
c2_iuci	1	0.363	0.035
cla	1	0.006	0.002
gobjects	14	0.002	0.002
claNotify	1	0.001	0.001
clo	1	0.003	0.003
hasDisplay	1	0	0
hold	6	0.009	0.005
isFigureShowEnabled	1	0	0
isPublishingTest	1	0	0
isStringScalar	6	0	0
markFigure	6	0.002	0.002
newplot	7	0.059	0.041
newplot > ObserveAxesNextPlot	7	0.008	0.002
newplot > ObserveFigureNextPlot	7	0.001	0.001
newplotwrapper	7	0.063	0.002
area	153	0.012	0.007
intersect	51	0.034	0.028
numboundaries	408	0.011	0.01
numsides	102	0.004	0.003
perimeter	153	0.01	0.006
checkAndSimplify	51	0.204	0.166
checkArray	867	0.003	0.003
checkConsistency	408	0.001	0.001
checkinput	51	0.007	0.003
checkPointArray	51	0.003	0.003
NumHoles	102	0.002	0.002
NumRegions	102	0.003	0.003
getXY	51	0.004	0.004
isEmptyShape	408	0.014	0.002
isEqualShape	51	0.035	0.004
parseIntersectUnionArgs	51	0.001	0.001
polyshape	102	0.221	0.01
settings	1	0.001	0
settings	1	0.001	0.001
usejava	1	0.001	0

that previous line and the graph. The scan intersection is shown in Figure 4.

3. Results

We conducted two simulation experiments on the designed contour parallel path boundary offset algorithm using Matlab software. The first simulation experiment was set to the inner shrinkage boundary distance of 1 mm and the hatch line spacing of 1 mm, and the second simulation experiment was set to the inner shrinkage boundary [28] distance of 0.1 mm and the hatch line spacing of 0.1 mm. Under these two sets of parameters, the total length of the hatching lines of the serrated parallel and contour parallel hatching curves were calculated, and

the number of horizontal lines of the serrated parallel hatching and the number of circles of the contour parallel hatching were also calculated. The average running time (in ms) was calculated based on multiple runs of the hatching program, and the running time ratio of the program runs under different conditions was also calculated. The simulation results for the two parameters are shown in Figures 4 and 5, respectively.

3.1. Simulation of Boundary Curve Offset Algorithm Based on Contour Parallel Path

3.1.1. Algorithm Analysis Results. We have performed an accurate analysis of the running time [29] of this algorithm and have derived the number of function calls and the time during

TABLE 5: Algorithm analysis based on zigzag parallel path simulation experiment II.

Function name	Number of calls	Total time (s)	Self-use time (s)
createCanvas	1	0.007	0.002
getInstance	1	0	0
hasBeenOpened	1	0.001	0.001
maybeShow	1	0.003	0.001
ToolBarController	1	0.005	0.004
axescheck	6	0.002	0.002
strcmpi ("parent", x)	6	0.001	0.001
c2_iuci	1	2.476	0.08
cla	1	0.006	0.002
gobjects	14	0.002	0.002
claNotify	1	0.001	0.001
clo	1	0.003	0.003
hasDisplay	1	0	0
hold	6	0.009	0.005
isFigureShowEnabled	1	0	0
isPublishingTest	1	0.001	0.001
isStringScalar	6	0	0
markFigure	6	0.002	0.002
newplot	7	0.057	0.038
newplot > ObserveAxesNextPlot	7	0.008	0.002
newplot > ObserveFigureNextPlot	7	0.001	0.001
newplotwrapper	7	0.061	0.002
area	1593	0.103	0.048
intersect	531	0.283	0.251
numboundaries	4248	0.099	0.085
numsides	1062	0.026	0.021
perimeter	1593	0.092	0.044
checkAndSimplify	531	1.945	1.653
checkArray	9027	0.029	0.029
checkConsistency	4248	0.01	0.01
checkinput	531	0.023	0.011
checkPointArray	531	0.01	0.01
NumHoles	1062	0.012	0.012
NumRegions	1062	0.023	0.023
getXY	531	0.012	0.012
isEmptyShape	4248	0.119	0.021
isEqualShape	531	0.272	0.016
parseIntersectUnionArgs	531	0.004	0.004
polyshape	1062	2.043	0.075
settings	1	0.001	0
settings	1	0.001	0.001
usejava	1	0.001	0

TABLE 6: Analysis of each factor of the zigzag path simulation experiment.

Experimental group	Time					Average time	Average time ratio	Circumference	Number of articles
Number of experiment	1	2	3	4	5				
1	0.073	0.076	0.076	0.074	0.078	0.0754	1.0026	9176	957
2	0.075	0.075	0.075	0.076	0.077	0.0756		868	85

the operation of the algorithm by running it several times. The results are shown in Tables 1 and 2.

3.1.2. Analysis Results. According to the simulation experiment results, we counted the running time, the number of laps of the path, and the total length of the path for each simulation experiment, respectively. The results are shown in Table 3.

3.2. Simulation Results of Domain Partitioning Algorithm Based on Sawtooth Parallel Paths without Self-Intersecting Graphs

3.2.1. Simulation Results. We generate the marking path according to the area division as follows. Figures 6 to 11 show each of the five areas of the division. The final results are shown in Figures 12 and 13, respectively.

3.2.2. Analysis of Domain Partitioning Intersection Algorithm Based on Sawtooth Parallel Paths. For this algorithm, we conducted two sets of simulation experiments with different parameters and performed statistical analysis on the number of calls and time of each function in the operation of the algorithm, and the following results are obtained in Tables 4 and 5, respectively.

3.2.3. Zigzag Parallel Path Analysis. We statistically analyze the average running time, path length, and number of path entries of the zigzag parallel path algorithm, and the following results are obtained in Tables 6.

4. Discussion and Conclusion

Through the simulation solution, we can know that the average operation time of the zigzag parallel pattern filling and contour parallel pattern filling algorithms is 1 mm and 0.1 mm. After the fitting algorithm in this paper, we can get a nearly parallel straight line, while the contour pattern can be filled in parallel. The K value of the fitting function of the algorithm is close to 10, that is, under the same magnification, the zigzag algorithm can approach the previous value in time.

The algorithm in this paper can establish the length of two paths, that is, the distance that the laser sweeps through the whole closed figure, and the two paths are not very different [30].

However, the results of the simulation marking pattern can be seen, the simulation pattern based on the marking algorithm of the contour parallel path is more accurate, and the simulation pattern obtained by the marking algorithm of the sawtooth parallel path algorithm is relatively rough. Of course, this is also related to the design of our algorithm, and we believe that the accuracy of the serrated parallel path marking will be improved after the algorithm is continuously iterated and optimized.

Data Availability

The datasets for this study can be found in the (official website of APMCM) (<https://www.apmcm.org/detail/2403>).

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Authors' Contributions

G. L. and Y. L. contributed to the conceptualization of the study; G. L., Y. L., and X. Y. contributed to the methodology; G. L. created the software; Y. L. and X. Y. validated the software; X. Y., C. J., and J. Y. contributed to formal analysis; G. L. contributed to the investigation; Y. L. helped to collect the resources; X. Y. contributed to data curation; G. L. prepared the draft for writing—original the study; G. L. and J. Y. contributed to writing—review and editing the study; Y. L. contributed to visualization; J. Y. contributed to supervision; Y. L. and C. J. contributed to project administration; Y. L. contributed to funding acquisition. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

The authors sincerely thank the editors and reviewers for their efforts. The authors sincerely thank Geng Xiaoyang for laying the foundation for our achievement. The authors are grateful to Yifan Liu for his help in making it possible for me to submit this paper.

References

- [1] R. D. Haun, "Laser applications," *IEEE Spectrum*, vol. 5, no. 5, pp. 82–92, 1968.
- [2] E. A. Zakharenko, E. I. Pryakhin, V. V. Romanov, and N. N. Shchedrina, "Development of the technology of ONBC code formation on a metal surface by laser marking," *Journal of Physics: Conference Series*, vol. 1753, no. 1, Article ID 012003, 2021.
- [3] V. J. Logeeswaran, M.-L. Chan, Y. Bayam et al., "Ultra-smooth metal surfaces generated by pressure-induced surface deformation of thin metal films," *Applied Physics A*, vol. 87, no. 2, pp. 187–192, 2007.
- [4] H.-C. Kim, S.-g. Lee, and M.-Y. Yang, "An optimized contour parallel tool path for 2D milling with flat endmill," *International Journal of Advanced Manufacturing Technology*, vol. 31, no. 5-6, pp. 567–573, 2006.
- [5] K. F. A. Hussein, S. A. Mehdi, and S. A. Hussein, "Image Encryption based on parallel algorithm via zigzag Manner with a new Chaotic system," *Journal of Southwest Jiaotong University*, vol. 54, no. 4, 2019.
- [6] H. Abdullah, R. Ramli, and D. A. Wahab, "Tool path length optimisation of contour parallel milling based on modified ant colony optimisation," *International Journal of Advanced Manufacturing Technology*, vol. 92, no. 1-4, pp. 1263–1276, 2017.
- [7] M. Sheng, M. Zhou, and Y. Yang, "Design and implementation of laser marking machine control system," *Journal*

- of *Physics: Conference Series*, vol. 1738, no. 1, Article ID 012128, 2021.
- [8] S. Vasanth and T. Muthuramalingam, "Multi Criteria Decision making of Power Diode based process parameters in laser beam machining using Taguchi dear methodology," in *Proceedings of the ATINER'S Conference Paper Series*, Athens, Greece, May 2018.
 - [9] M. Priyadarshini, P. P. Tripathy, D. Mishra, and S. Panda, "Multi characteristics optimization of laser drilling process parameter using fuzzy-topsis method," *Materials Today: Proceedings*, vol. 4, no. 8, pp. 8538–8547, 2017.
 - [10] A. Çalık, "A novel Pythagorean fuzzy AHP and fuzzy TOPSIS methodology for green supplier selection in the Industry 4.0 era," *Soft Computing*, vol. 25, no. 3, pp. 2253–2265, 2021.
 - [11] A. K. Mauraya, D. Mahana, P. Tyagi et al., "Structural and ultraviolet photo-detection properties of laser molecular beam epitaxy grown GaN layers using solid GaN and liquid Ga targets," *Physica Scripta*, vol. 96, no. 8, Article ID 085801, 2021.
 - [12] A. Bandari, "Beam engineering strategies reduce heat accumulation effects in high power, ultrashort pulse laser machining," *Scilight*, vol. 2020, no. 36, Article ID 361107, 2020.
 - [13] A. W. Lohmann, J. Ojeda-Castañeda, and A. Serrano-Heredia, "Bessel functions: parallel display and processing," *Optics letters*, vol. 19, no. 1, p. 55, 1994.
 - [14] Y. Zhang, H. Li, T. Wang, B. Liu, and G. Wang, "A hybrid tool-path with no pause generation algorithm for 3D printing," *Journal of Physics: Conference Series*, vol. 1754, no. 1, Article ID 012222, 2021.
 - [15] J. Wang, L. Qin, and W. Xu, "Flexible and high precision thermal metasurface," *Communications Materials*, vol. 2, no. 1, p. 89, 2021.
 - [16] S. Mukherjee, S. Mukherjee, D. P. Mukherjee et al., "Computer vision, graphics, and image processing," in *Proceedings of the ICVGIP 2016 Satellite Workshops, WCVA, DAR, and MedImage*, Springer Cham, Guwahati, India, Lecture Notes in Computer Science, Guwahati, India, December 2016.
 - [17] H. Luo, J. Luo, R. Li, and M. Yu, "Optimization algorithm design of laser marking contour Extraction and graphics hatching based on image processing technology," *Journal of Physics: Conference Series*, vol. 2173, no. 1, Article ID 012078, 2022.
 - [18] H. C. Kim, S. G. Lee, and M.-Y. Yang, "A new offset algorithm for closed 2D lines with Islands," *Transactions of the Korean Society of Mechanical Engineers A*, vol. 30, no. 2, pp. 141–148, 2006.
 - [19] L. Xia, Y. Hu, W. Chen, and X. Li, "Spot pattern separation in multi-beam laser pointing using a neural network," *Optics and Lasers in Engineering*, vol. 140, 2021.
 - [20] X. Yang, "Filling algorithm of polygons and Embedded polygons," *Journal of Physics: Conference Series*, vol. 1827, no. 1, p. 012144, 2021.
 - [21] T. Briand and P. Monasse, "Theory and Practice of image B-spline interpolation," *Image Processing On Line*, vol. 8, pp. 99–141, 2018.
 - [22] G. Pham, S.-H. Lee, and K.-R. Kwon, "Interpolating spline curve-based Perceptual Encryption for 3D printing models," *Applied Sciences*, vol. 8, no. 2, p. 242, 2018.
 - [23] I. A. Blatov, A. I. Zadorin, and E. V. Kitaeva, "Cubic spline interpolation of functions with high gradients in boundary layers," *Computational Mathematics and Mathematical Physics*, vol. 57, no. 1, pp. 7–25, 2017.
 - [24] I. Hanhan and M. D. Sangid, "ModLayer: a matlab GUI Drawing Segmentation tool for visualizing and Classifying 3D data," *Integrating Materials and Manufacturing Innovation*, vol. 8, no. 4, pp. 468–475, 2019.
 - [25] A. Hadir, K. Zine-dine, M. Bakhouya, and A. I. Technologies, "Applications. Improvements of centroid Localization algorithm for Wireless Sensor Networks," in *Proceedings of the 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (Cloud-Tech)*, pp. 1–6, IEEE, Marrakesh, Morocco, November 2020.
 - [26] Y. Yang, H. T. Loh, J. Y. H. Fuh, and Y. G. Wang, "Equidistant path generation for improving scanning efficiency in layered manufacturing," *Rapid Prototyping Journal*, vol. 8, no. 1, pp. 30–37, 2002.
 - [27] I. A. A. Al-Rawi, "Implementation of an efficient scan-line polygon fill algorithm," *Computer Engineering and Intelligent Systems*, vol. 5, pp. 22–28, 2014.
 - [28] K. W. F. Wan, F. C. Xia, and H. Tu, "An interfere coincidence processing algorithm for two-dimensional curve offset," in *Proceedings of the The 2nd International Conference on Information Science and Engineering*, pp. 4362–4365, IEEE, Hangzhou, China, December 2010.
 - [29] P. Tirado and O. Valero, "The average running time of an algorithm as a midpoint between fuzzy sets," *Mathematical and Computer Modelling*, vol. 49, no. 9-10, pp. 1852–1868, 2009.
 - [30] A. Tharwat, M. Elhoseny, A. E. Hassanien, T. Gabel, and A. Kumar, "Intelligent Bézier curve-based path planning model using Chaotic Particle Swarm Optimization algorithm," *Cluster Computing*, vol. 22, no. S2, pp. 4745–4766, 2018.