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 Jia 2

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

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 \([x_i, x_{i+1}]\) \((i = 0, 1, \ldots, n - 1, x \) increasing), \( S(x_i) = S(x_{i+1}) \) is a cubic polynomial

(B) The following relationship is satisfied:

\[
S_i(x) = y_i, \quad (i = 0, 1, \ldots, n) \tag{1}
\]

(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

\[
S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \tag{2}
\]

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

2.1.2. Request a Solution. Interpolation and continuity:

\[
S_i(x_i) = y_i, \quad S_i(x_{i+1}) = y_{i+1}, \tag{3}
\]

Differential continuity:

\[
S_i'(x_i) = S_i''(x_{i+1}), \quad S_i'(x_{i+1}) = S_i''(x_{i+1}), \tag{4}
\]

where \( i = 0, 1, \ldots, n - 2 \).

Differential equation of a spline curve:

\[
S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \tag{5}
\]

\[
S_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2, \tag{6}
\]

\[
S_i''(x) = 2c_i + 6d_i(x - x_i). \tag{7}
\]

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

\[
h_i = x_{i+1} - x_i. \tag{8}
\]

Thus, we can deduce

\[
a_i = y_i, \quad a_i + h_i b_i + h_i^2 c_i + h_i^3 d_i = y_{i+1}, \tag{9}
\]

\[
b_i + 2h_i c_i + 3h_i^2 d_i - b_{i+1} = 0, \quad 2c_i + 6h_i d_i - 2c_{i+1} = 0. \tag{10}
\]

We assume that

\[
S_i'(x_i) = 2c_i. \tag{11}
\]

Thus, we can deduce the following result:

\[
m_i + 6h_i d_i - 2c_{i+1} = 0. \tag{12}
\]

It is equivalent to the following equation.

\[
m_i + 6h_i d_i - m_{i+1} = 0. \tag{13}
\]

Thus, the following formula is derived as
We substitute $c_i$, $d_i$ into

\[ y_i + h_i b_i + h_i^2 c_i + h_i^3 d_i = y_{i+1}, \]

we can derive the formula:

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

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.

\[ S_0''(x_1) = S_1''(x_1), \]
\[ S_{n-2}''(x_{n-1}) = S_{n-1}''(x_{n-1}). \]

After that we can derive the following conclusions:

\[ h_i (m_1 - m_0) = h_0 (m_2 - m_1), \]
\[ h_{n-1} (m_{n-1} - m_{n-2}) = h_{n-2} (m_n - m_{n-1}). \]

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

\[
\begin{bmatrix}
h_1 & h_0 + h_1 & -h_0 & \cdots & \cdots & 0 \\
-1 & h_0 & 2(h_0 + h_1) & h_1 & 0 & \cdots \\
0 & h_1 & 2(h_1 + h_2) & h_2 & 0 & \cdots \\
\vdots & 0 & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\
0 & \cdots & \cdots & -h_{n-1} & h_{n-2} & h_{n-1} & -h_{n-2}
\end{bmatrix}
\]
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 around a point, line, or polyshape object. It can return the intersection of a closed graph to the bottom in 1mm steps in order to scan 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 each array.

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

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

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

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.

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

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.
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             |
| Cla                           | 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               | 0.181          | 0.112             |
| isOpenFigureShowEnabled       | 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) | Number of laps | Average time | Average time ratio | Circumference |
|--------------------|---------|----------------|--------------|--------------------|---------------|
|                    | 1       | 2              | 3            | 4                  | 5             | 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 9: Region 3.

Figure 10: Region 4.

Figure 11: Region 5.

Figure 12: Zigzag parallel experiment.

Figure 13: Zigzag parallel experiment II.
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

| 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            |
| axescheck     | 6               | 0.002          | 0.002            |
| strcmpi ("parent", x) | 6 | 0.001 | 0.001 |
| c2_juci       | 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              | 0                |
| settings      | 1               | 0.001          | 0.001            |
| usejava       | 1               | 0              | 0                |
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             |
| axestrcheck                            | 6               | 0.002          | 0.002             |
| strcmpi("parent", x)                   | 6               | 0.001          | 0.001             |
| c2_juci                                | 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                         | 8               | 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          | 0.001             |
| settings                               | 1               | 0.001          | 0.001             |
| usejava                                | 1               | 0.001          | 0.001             |

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

| Experimental group | Time Number of experiment | 1    | 2    | 3    | 4    | 5    | Average time | Average time ratio | Circumference | Number of articles |
|--------------------|----------------------------|------|------|------|------|------|--------------|---------------------|---------------|-------------------|
| 1                  |                            | 0.073| 0.076| 0.074| 0.078| 0.0754| 0.074        | 1.0026             | 9176          | 957               |
| 2                  |                            | 0.075| 0.075| 0.075| 0.076| 0.0756| 0.0756       | 1.0035             | 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. Zakarenko, E. I. Pryakhin, V. V. Romanov, and N. N. Shchedrina, “Development of the technology of 0NBC 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. Mehdj, 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
X. Yang, "Filling algorithm of polygons and Embedded I. A. Blatov, A. I. Zadorin, and E. V. Kitaeva, "Cubic spline G. Pham, S.-H. Lee, and K.-R. Kwon, "Interpolating spline I. Hanhan and M. D. Sangid, "ModLayer: a matlab GUI T. Briand and P. Monasse, "fiV_heory and Practice of image 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. X. Yang, "Filling algorithm of polygons and Embedded polygons," Journal of Physics: Conference Series, vol. 1827, no. 1, p. 012146, 2021. T. Briand and P. Monasse, "Theory and Practice of image B-spline interpolation," Image Processing On Line, vol. 8, pp. 99–141, 2018. 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. 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. I. Hanhan and M. D. Sangid, "ModLayer: a matlab GUI Drawing Segmentation tool for visualizing and Classifying 3D