Team Number:	2020230010136
Problem Chosen:	A

2020 APMCM summary sheet

With laser, the hatch tool of laser marking machine can be used to realize the work of making two-dimensional graphics. However, the selection and layout of hatch methods and paths greatly affect the difficulty and efficiency of incubation. The hatch problem of the laser marking machine on the two-dimensional closed graph can be equivalent to the problem of filling the two-dimensional closed curvilinear polygon with the curves connected by scattered points. Here, we provide two Python-based filling algorithms. They respectively correspond the zigzag hatch and the contour parallel hatch. Both of these two algorithms utilize vectors to the greatest extent on the basis of inward shrinking of the initial graphics to process the graphics lattice data, and use the distance between two points and the argument of vector as the basis for determining. Importantly, the algorithm automatically filters the lattice data at the sharp points of the figure after rational simplification, optimizes the shrinking of the arc with a very small radius of curvature. It also can ensure that the graphics formed after filtering can meet the requirements. This method breaks the limitation of the traditional algorithm that can only handle simple geometric figures. It provides the possibility to deal with complex curve polygons, and optimize the laser marking machine's refined processing of complex graphics. Meanwhile, we avoid solving a large number of linear equations in the algorithm and improve its efficiency. However, because of the filtering of the lattice, the graphics we get may have some unexpected deformation. Therefore, we still need some extra program to adjust it.

Keywords: Python filling algorithm vector zigzag hatch contour parallel hatch

Contents

I. Introduction

The sharpest knife, laser, has been widely used to mark specified 2D-compound curve graph. Efficiency is an important indicator for this craft. To maximize the efficiency, we need to make the hatch curves generated automatically and quickly by laser marking machine keep parallel to the boundary lines of the figures, and distribute evenly.

1.1 Problem Restatement

The curve used by laser etching has a certain width. In order to simplify the calculation, we approximate it to a one-dimensional curve, and set the spacing of the curve to achieve the filling effect. Taking into account the moving efficiency of the robotic arm in actual industrial production, the curve used to fill the graph must be as continuous as possible. Based on the above discussion, we can equate the problem as the optimal solution for filling a two-dimensional curve polygon with a continuous curve.

1.2 Previous Work

In previous work, we have learned that there has been a work of depicting a vector diagram by calculating basic data through DSP. In the processing of bitmaps, due to the large storage space occupied by bitmap files, a small amount of pixel data is selected according to the set resolution during the processing of the marking software. Such processing greatly reduces the amount of marking data of the bitmap file and improves its marking speed. Increasing the resolution of the picture can improve the fineness of the graphic marking, but due to the limitation of the scanning speed, the marking time increases accordingly.

1.3 Problem Solving Strategies

For the two hatching methods mentioned in the question, we originally proposed two different general algorithm. They can solve such problems universally. For the zigzag hatching, we first indent the graphics formed by the original lattice. And by finding the extreme points of the graphics, a dividing line parallel to the x-axis is made according to the traversal direction to intersect the graphics at one point. Therefore, we divide the entire graphics into a number of areas. Finally, we do zigzag hatching from top to bottom in each partition. For the contour parallel hatch, we will iterate the steps of

indentation. In the process of hatching, we will remove points that will cause overlap in the coverage area and add some points between two points far away. When the last curve generated by the iteration is about to cross, stop the iteration and partition the area formed by the last curve. Finally, we continue to iterate until the filling accuracy requirement is reached.

II. The Description of the algorithms

2.1 How do we use algorithms to implement the whole process?

The applicable objects of the algorithms in this paper are most of the two-dimensional geometric figures with curved polygons as the boundary. Both algorithms use given points data to generate appropriate point coordinate data through a certain transformation, and then connect them in a certain order to obtain the incubation path. They have a certain approximation, and through reasonable judgment conditions, the data of the generated points are appropriately screened, and points are automatically added between two points far away to complete the hatch path.

2.1.1 *The direction parallel hatch*

2.1.2 *The contour parallel hatch*

The method we use is to form two vectors through two points before and after a certain point. The straight line where the two vectors are located is translated by one precision unit into the graph, and the resulting intersection point is the mapping of the point of the new path generated inward by this point. In order to get the coordinates of the new point, we will unitize the vectors formed by the original three points, and then multiply it by the ratio of the sine of the angle between the accuracy and the vector. Then add the two vectors after processing to obtain the vector needed for transformation. The coordinates of the original point are added to the transformation vector to obtain the coordinates of the new point. According to the data given in the question, the order of the points on the outer boundary is counterclockwise. In order to ensure that the newly generated points are inside the original graphics, all new points must be guaranteed to be on the right side of the vector where their origin is located. We judge by the angle of two vectors and the angle between them. When this corner protrudes outside the image, the newly generated point appears inside the corner. When this corner is sunken inside

the image, we add a negative sign to the transformation vector to make it inverted. Then the newly generated point appears on the outside of the corner. Through this algorithm, we can make both convex and non-convex polygons shrink inward. Performing this step in a loop, you can get the coordinate data of a new set of points. This set of data will not only be used to generate a layer of contours, but also will be used to generate the next set of lattices.

2.2 The Reasonableness of approximating

2.2.1 *Rounding of data point coordinates*

In the zigzag hatching, since the parallel lines must be parallel to the x-axis, we must make the ordinates of the data points obtained after indentation correspond to the same. This will cause deformation in the part with a smaller radius of curvature in the graph. However, due to the limitation of hatching accuracy, we have no way to fill these parts (the remaining space is not enough to make the distance between parallel lines reach the specified value), so this kind of deformation will not be reflected in hatching.

2.2.2 *Deleting and adding some special points*

In contour parallel hatch, during the iteration of the shrinking operation, we found that there are two situations that do not meet the requirements. First, the points generated after the iteration are too close to the curve before the iteration; second, the curves generated after the iteration form a closed triangle. Our algorithm will automatically identify these two situations and delete or add data points. It will cause slight deformation of the graph, for example, a smooth curve will produce some edges and corners after iteration, resulting in some small vacancies. However, when the accuracy meets the requirements, the finished product is not partially enlarged, but from the overall perspective, it is difficult to see that the indented graphics has deformed, and the existence of these vacancies cannot be seen. Moreover, the purpose of our indentation is to generate contour lines to fill the graphics, and this deformation will not interfere with our ultimate goal. So our approximation is reasonable.

2.3 The local optimization and the overall optimization

In computer system design, Amdahl's law is generally used to describe the effect of improving the performance of a certain part of the system. When a certain part of the

system is improved, we can get the speedup of the program:

$$S = 1/((1-a) + a/k) \tag{1}$$

Among them, a is the ratio of the execution time required for this part to the total execution time of the program, and k is the ratio of the execution time required for this part after optimization to that before optimization.

2.4 Algorithmic Complexity

The complexity of the algorithm is divided into time complexity and space complexity. Since the number of variables in the article is relatively unchanged, we mainly discuss time complexity here.

III. Models

3.1 Basic Model

3.1.1 *Terms, Definitions and Symbols*

The signs and definitions are mostly generated from queuing theory.

3.1.2 *Assumptions*

3.1.3 *The Foundation of Model*

1) The utility function

- The cost of :
- The loss of :
- The weight of each aspect:
- Compromise:

  3) The overall optimization and the local optimization

- The overall optimization:
- The local optimization:
- The optimal number of :

3.1.4 *Solution and Result*

1) The solution of the integer programming: 2) Results:

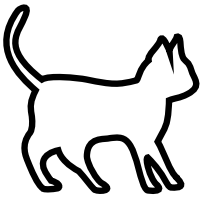

Figure 1 关注我们公众号，学习更多知识

3.1.5 *Analysis of the Result*

- Local optimization and overall optimization:
- Sensitivity: The result is quite sensitive to the change of the three parameters
-
- Trend:
- Comparison:

3.1.6 *Strength and Weakness*

Strength: The Improved Model aims to make up for the neglect of . The result seems to declare that this model is more reasonable than the Basic Model and much more effective than the existing design.

Weakness: Thus the model is still an approximate on a large scale. This has doomed to limit the applications of it.

IV. Conclusions

4.1 Conclusions of the problem

-
-
-
-

4.2 Methods used in our algorithm

4.2.1 *The direction parallel hatch*

The zigzag hatch is embodied in our algorithm as a problem of reordering the data points. For the data points generated after indentation, we partition them according to the geometric shapes they form, and order the points on each area according to the ordinate. By connecting the data points we ordered we get several sets of zigzag curves.

4.2.2 *The contour parallel hatch*

Here we have adopted a method similar to a binary tree. Through a series of calculations on the original data points, a new set of data points is obtained; the same calculation is performed on the new data points to obtain the next set of data points. There is a correspondence between the data points in each group. But unlike the binary tree method, this method has a one-to-one correspondence between the old and new data points. Therefore, some areas will produce some points that destroy the shape of the figure, and our algorithm will delete them based on the geometric properties of the figure formed by these points.

4.3 Applications of our algorithm

The algorithm we proposed realizes the filling of two-dimensional closed curve polygons through zigzag hatching and contour parallel hatching. It is expected to be used in laser marking after further optimization. In addition, we will increase the dimensionality of the algorithm in future work to enable it to fill the three-dimensional closed space. This can open exciting perspectives for 3d printing, laser engraving, etc.

V. Future Work

5.1 Ideas to solve existing problems

Zigzag, as the most common hatching method at present, has many sharp turns. For shapes with complex borders or hollow, this disadvantage will be very obvious.The contour parallel algorithm can avoid too many sharp turns, but it leads to more contours and the contours are not connected. For the problems of these two hatching methods, we

can use the Fermat spiral filling to solve it. The Fermat spiral consists of two intertwined sub-helices, one inward and the other outward. The Fermat spiral has three advantages:

1) Similar to the contour parallel hatch, the Fermat spiral only has a sharp turn at the center.

2) Some Fermat spirals can continuously fill the 2D area; traditional spirals are either inward or outward, and Fermat spirals are both inward and outward, and can meet at the boundary.

3) To facilitate the connection between the spirals, the start and end nodes of the Fermat spiral can be selected arbitrarily at the boundary.

5.2 Optimization of performance and efficiency

5.2.1 *Reducing the idling of the robotic arm*

In actual industrial applications, in addition to the running time of the algorithm, we must consider another time cost, the efficiency of the robot's movement. For the zigzag hatch, according to the complexity of the graphics, we have to partition them. This results in a certain distance between the start and end of each curve. It takes a certain amount of time for the robot arm to move between them. In order to reduce this waste, we need to make a more reasonable design for the partition and the angle between the parallel line and the x-axis. In the same way, there will be multiple converging centers in contour hatching. Reducing the number of converging centers as much as possible means reducing the blank movement time of the robotic arm.

5.2.2 *Reducing the amount of data*

In the actual laser marking process, the original image often contains a large number of data points. To delete redundant or isolated data points can shorten the running time of the program.

5.2.3 *Program optimization*

1)Algorithm optimization: An excellent algorithm can reduce the complexity of the algorithm by several orders of magnitude, and its average time-consuming is generally less than other algorithms with high complexity There is another situation where the algorithm itself is relatively complicated, its time complexity is difficult to reduce, and its efficiency does not meet the requirements. The measures that can be taken are:

maintaining the effect of the algorithm to improve efficiency, and discarding part of the effect in exchange for a certain efficiency.

2)Code optimization:

a. Avoid multiplication (division) and redundant calculations inside the loop

b. Avoid too many dependencies and jumps inside the loop

c. Convert floating-point operations to integer operations

d. Exchange space for time

e. Pre-allocated memory

3)Instruction optimization Instruction optimization generally uses a specific instruction set to quickly implement certain operations. Another core idea of instruction optimization is packed operations.

5.2.4 *Multi-threaded operation*

Due to the limitation of single-threaded, our code can only run in a single thread. However, we found that after partitioning, the iteration of each region, the correspondence between points, and the detection of each point are relatively independent, which makes it possible for some programs to run in multiple threads. Through multi-threaded processing, we can greatly reduce the running time of the program, thereby improving its performance.

VI. References

[1] Author, Title, Place of Publication: Press, Year of publication.

[2] author, paper name, magazine name, volume number: starting and ending page number, year of publication.

[3] author, resource title, web site, visit time (year, month, day).

[4] LaTeX资源和技巧学习 https://www.latexstudio.net

[5] LaTeX问题交流网站 https://wenda.latexstudio.net

[6] 模板库维护 https://github.com/latexstudio/APMCMThesis

[7] Kaposi, M. "IDIOPATHISCHES MULTIPLES PIGMENTSARKOM DER HAUT." Wiener Klinische Wochenschrift, vol. 4, no. 2, 1872, pp. 265273.

VII. Appendix

Listing 1: The matlab Source code of Algorithm

```
kk=2;[mdd,ndd]=size(dd);
while ~isempty(V)
[tmpd,j]=min(W(i,V));tmpj=V(j);
for k=2:ndd
[tmp1,jj]=min(dd(1,k)+W(dd(2,k),V));
tmp2=V(jj);tt(k-1,:)=[tmp1,tmp2,jj];
end
tmp=[tmpd,tmpj,j;tt];[tmp3,tmp4]=min(tmp(:,1));
if tmp3==tmpd, ss(1:2,kk)=[i;tmp(tmp4,2)];
else,tmp5=find(ss(:,tmp4)~=0);tmp6=length(tmp5);
if dd(2,tmp4)==ss(tmp6,tmp4)
ss(1:tmp6+1,kk)=[ss(tmp5,tmp4);tmp(tmp4,2)];
else, ss(1:3,kk)=[i;dd(2,tmp4);tmp(tmp4,2)];
end;end
dd=[dd,[tmp3;tmp(tmp4,2)]];V(tmp(tmp4,3))=[];
[mdd,ndd]=size(dd);kk=kk+1;
end; S=ss; D=dd(1,:);
```

Listing 2: The lingo source code

```
kk=2;
[mdd,ndd]=size(dd);
while ~isempty(V)
    [tmpd,j]=min(W(i,V));tmpj=V(j);
for k=2:ndd
    [tmp1,jj]=min(dd(1,k)+W(dd(2,k),V));
    tmp2=V(jj);tt(k-1,:)=[tmp1,tmp2,jj];
end
    tmp=[tmpd,tmpj,j;tt];[tmp3,tmp4]=min(tmp(:,1));
if tmp3==tmpd, ss(1:2,kk)=[i;tmp(tmp4,2)];
else,tmp5=find(ss(:,tmp4)~=0);tmp6=length(tmp5);
if dd(2,tmp4)==ss(tmp6,tmp4)
    ss(1:tmp6+1,kk)=[ss(tmp5,tmp4);tmp(tmp4,2)];
else, ss(1:3,kk)=[i;dd(2,tmp4);tmp(tmp4,2)];
```

```
end;
end
    dd=[dd,[tmp3;tmp(tmp4,2)]];V(tmp(tmp4,3))=[];
    [mdd,ndd]=size(dd);
    kk=kk+1;
end;
S=ss;
D=dd(1,:);
```