Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A code-based approach for labeling in complex irregular regions*

Zhi-long LI†, Jun-jie CAO, Xiu-ping LIU†‡, Zhi-xun SU
(*School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China*)
†E-mail: zhilongli@gmail.com; xpliu@comgi.com

Received Apr. 17, 2009; Revision accepted Aug. 17, 2009; Crosschecked Aug. 31, 2009

Abstract: Labeling information in a complex irregular region is a useful procedure occurring frequently in sheet metal and the furniture industry which will be beneficial in parts management. A fast code-based labeler (FCBL) is proposed to accomplish this objective in this paper. The region is first discretized, and then encoded by the Freeman encoding technique for providing the 2D regional information by 1D codes with redundancies omitted. We enhance the encoding scheme to make it more suitable for our complex problem. Based on the codes, searching algorithms are designed and can be extended with customized constraints. In addition, by introducing a smart optimal direction estimation, the labeling speed and accuracy of FCBL are significantly improved. Experiments with a large range of real data gained from industrial factories demonstrate the stability and millisecond-level speed of FCBL. The proposed method has been integrated into a shipbuilding CAD system, and plays a very important role in ship parts labeling process.

Key words: Labeling, Freeman codes, Region filling, Optimal direction, Shipbuilding
doi:10.1631/jzus.A0920214 **Document code:** A **CLC number:** TP391.72

INTRODUCTION

Labeling is useful in a variety of applications in computer aided manufacturing (CAM) such as steel metal, clothing, map annotating, shipbuilding, and the furniture industry. During production procedures in manufacturing industry, a large number of parts or products need to be labeled on their usable portions for further pipelining operations. A traditional method is used in some industries, where workers decide the layout from their experience. This lowers efficiency since these manual operations are usually tedious, time-consuming, and highly error prone. Also, the dyes used in some industries are highly volatile. Long term usage of them would do harm to workers.

We consider the following labeling problem. Given a part region with a complex irregular structure and information like characters or graphics, locate a set of positions in the region to label all input information. Labeling focuses on the existence of the solution no matter whether or not it is unique. The actual problem may be constrained in a sub-region of the given region. The labeled information must be within the region, avoiding intersecting with the region boundaries or interior features like lines. To obtain a comfortable layout for reading, information needs to be labeled continuously. When the information cannot be labeled in the region, the labeling approach needs to adjust the information to obtain a more suitable outline for the region.

This paper intends to discuss 2D labeling approach based on the encoding method. Inspired by modern computer graphics techniques, we identify the boundary encoding procedure as a region filling problem (Tang and Lien, 1988; Henrich, 1994) and use Freeman codes (Freeman, 1961) for encoding. Codes are enhanced for adapting to more generalized cases. Then, searching strategies are introduced to pack the labeling with user-driven constraints in a simple yet efficient way. Instead of the traversing of

‡ Corresponding author
* Project supported by the National Natural Science Foundation of China (Nos. 60873181, 60673006 and 60533060) and the Program for New Century Excellent Talents in University, China (No. NCET-05-0275)

searching directions, the least square method, which takes advantages of reasonable discretization, is adopted to improve the labeling speed.

Since most CAD software represents region boundaries of real world parts and products with a couple of basic graph elements such as line segments and arcs already, we use these elements to construct foundational components of our algorithm. It provides local control of each element during labeling, while maintaining the efficiency and simplicity with respect to geometric operations.

To control the labeling, users can specify the configuration of each key step in the procedure using the intelligible parameters placed on the interface, or let the program give an experiential configuration by itself according to the input data. Such a scheme enables flexible operations and further integrations for various requirements.

Our specific contributions are: (1) We develop a simple and efficient algorithm that can automatically accomplish the goal of labeling in complex irregular regions. (2) To the best of our knowledge, this is the first attempt to bring the encoding method into the labeling community. We enhance the codes gained by the encoding method to suit the actual cases. All detailed problems in the labeling procedure are solved based on the encoding results.

RELATED WORKS

Similar problems like label placement and nesting are extensively studied by researchers. However, their problem settings are different from ours. Hence, existing methods are not very suitable for our labeling problem. We introduce some samples of them and indicate the main detailed differences below.

Label placement problem

The aim of the label placement problem is to communicate the meaning of each graph object via text or graphic labels (Wolff, 2009). This technique is very useful for UML (unified modeling language) diagrams, ER (entity relationship) diagrams, map design, etc. Binucci *et al.*(2005) presented algorithms that compute labeled drawings in the orthogonal drawing convention with several additional optimization goals. Dogrusoz *et al.*(2007) described a general framework for modeling, drawing, editing, and automatic placement of labels respecting user constraints. They also presented a visualization tool, Graph Editor Toolkit, for further demonstration of their algorithm. Compared to many algorithms that exist for the label placement problem, Kakoulis and Tollis (2006) considered positioning multiple labels per graphic feature in a map or drawing and presented encouraging results. Lan *et al.*(2005) discussed the placement of Chinese annotation from the viewpoint of graphics. They classified the area features into three types and solved these problems with the help of long-diagonals. Mote (2007) offered a method for labeling point-features on dynamic maps in real time without pre-processing. A greedy randomized adaptive search procedure was proposed by Cravo *et al.* (2008) for point label placement. Alvim and Taillard (2009) proposed a new heuristic method based on the application of the POPMUSIC frame for a similar aim.

The labeling problem addressed here has three essential differences from the common label placement problem. Firstly, label placement often discusses computing drawings of graphs with labels on vertices and edges. Our labeling problem needs to be considered with respect to complex irregular regions. Secondly, although area label placement is specific to regions, it can be solved by line-based methods such as polygon diagonal since the regions are often big enough for laying the labels. This would obtain a separated result and lacks the visual continuity required here. Lastly, the whole procedure of label placement is an optimization of all candidates and the inputs are the fixed shaped labels. Conversely, the character inputs of our problem need to be adjusted to utilize the regions as much as possible.

Nesting problem

Approaches to the nesting problem are dedicated to most efficiently cutting product patterns from raw materials. A sufficient survey in this field can be found in Bennell and Oliveira (2008). Ramesh Babu and Ramesh Babu (1999) proposed a hybrid approach employing both genetic and heuristic algorithms that aim to arrange different rectangular parts on multiple rectangular sheets. Wu *et al.*(2003) also used hybrid algorithms in combination with improved Babus'

heuristic (IBH) method to solve cutting problems for 2D rectangular parts on multiple plates. Al-Assaf (2003) proposed an approach based on extracting human intuitive thoughts and compared well with results achieved by expert human operators. Apart from research by Ramesh Babu and Ramesh Babu (2001), the above studies do not consider the situation where parts and sheets are multiple complexes with internal features or defective regions. A quick location and movement (QLM) algorithm was proposed by Lee *et al.*(2008) to solve the situation of irregular shapes nested on multiple irregular and sheets. It was demonstrated that this approach is comparable or superior to the above algorithms in both utilization and processing time. Egeblad *et al.*(2007) proposed a meta-heuristic neighborhood searching algorithm and provided a solution for 3D nesting. A genetic algorithm (GA) to solve the nesting problem of shoe making was given by Yang and Lin (2009). An immune GA was presented by Liang and Ye (2008) to overcome the shortages of premature constringency and low efficiency existing in GAs.

Nesting is different from labeling. First, the input shapes of nesting are fixed but they can be adjusted in labeling. Adjusting the input will take a rather long time for current nesting methods. Second, instead of material utilization efficiency, labeling needs to have more attention paid to the visual effect of layout and user interaction constraints. Finally, considering the difference between their aims, i.e., the difference between 'most use' and 'can use', nesting needs global optimization but labeling can be solved locally.

METHODOLOGY

In this section, we present the mechanism of our fast code-based labeler (FCBL) for locating positions in complex irregular regions. The overview of the approach is shown by a flow chart in Fig.1, and details of each step are described below.

Discretizing the region to grid settings

Usually, a region in a 2D plane is surrounded by one exterior boundary and several interior boundaries. All the boundaries are closed and presented by elements (i.e., line segments and arcs). Denote the given region by Ω and suppose it is placed on a rectangular sheet R which is laid out horizontally. R is discretized to a grid (in this paper, 'grid' refers to the whole chessboard-like rectangular region and 'cell' refers to the unit sub-region in grid) R'. Ω takes up the corresponding cells with its discrete points and all these occupied cells make up of Ω'. In practice, we choose one information character's width w_{char} and height h_{char} as the unit to discretize R in horizontal and vertical directions, respectively. It means that the cell size is determined by the user defined characters (for instance, if one wants to label characters with a font size of 23 pt×25 pt, then each cell in the grid is 23 pt×25 pt), and grid resolution will change according to the given character size. Elements are discretized using the global unit $\text{unit}_{\text{global}}=\min(w_{\text{char}}, h_{\text{char}})$; namely, line segments are discretized to points with interval $\text{unit}_{\text{global}}$ while arcs with arc-length $\text{unit}_{\text{global}}$ (Fig.2, left). In this way, each cell contains one discrete point in general cases. An extreme case like the sharp arcs can be ignored because such sub-regions cannot satisfy the labeling requirement. Meanwhile, the bottommost $y_{\min}$, topmost $y_{\max}$, leftmost $x_{\min}$, and rightmost $x_{\max}$ coordinates of these discrete points are recorded. R' is actually restricted to the grid with ($x_{\min}$, $y_{\min}$) as its left-bottom and ($x_{\max}$, $y_{\max}$) as its right-top.

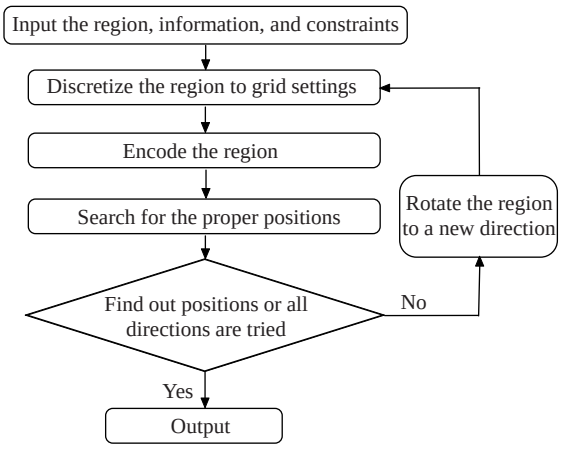


Fig.1 Flow chart of our approach

Discretization in this manner is based on the following considerations. First, information is composed of characters mostly (Figs.3a and 3b), so the labeling result (Fig.3c) can be used without further modifications. Moreover, any other graphics information can be replaced with a couple of characters' surrounding rectangles (Fig.3d). In addition, it is

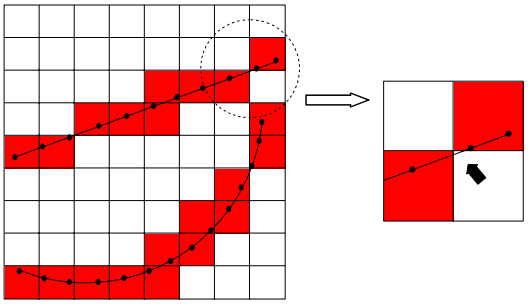


Fig.2 Discretizing the region, i.e., the line segments and the arcs, to points (left). Sharp-angled case (right) may happen somewhere

The cells which contain the discretized points are colored

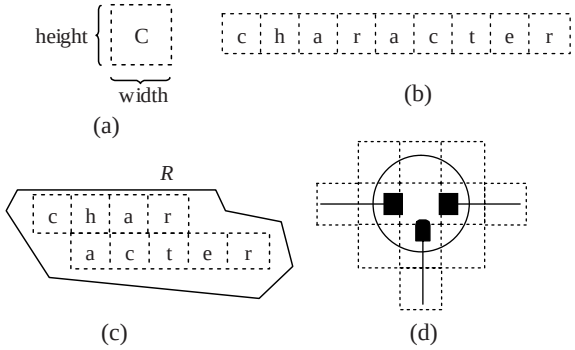


Fig.3 Characters actually constitute the information which needs to be labeled

(a) One character; (b) An information character row; (c) The character row in (b) is separated into two rows to fit the region size; (d) Covering graphics with character rectangles

convenient to adjust the grid resolution for easier labeling since the character size is easy to control. Second, $\text{unit}_{\text{global}}$ guarantees that the discrete points of the elements take up the cells of R' successively. It reflects the analogous connectivity in the continuous situation. Though a sharp-angled case appears (Fig.2, right) and may lead the searching step in subsection 'Searching for the proper positions' to touch the real boundaries sometimes, it can be disposed by adding just one character at both ends of each information character row. Given the fact that the main components of R are line segments and arcs, the implementing techniques would make use of drawing algorithms (Hearn and Baker, 1997) in raster graphics for a better discretization effect. Finally, the original region is discretized to a relatively simple grid version, enabling raster graphics techniques to be introduced directly.

Encoding the region

As a necessary condition of labeling, we need to distinguish region interior and exterior through cells in the grid, which is merely a region filling problem. We employ Freeman codes (Freeman, 1961; Tang and Lien, 1988; Henrich, 1994) to encode the boundary cells. Freeman codes describe the vector relation between two adjacent cells in the grid. For a closed discrete curve in the grid, this encoding scheme will give unique codes. Orienting the exterior boundary anticlockwise and the interior ones oppositely, the region filling algorithm starts to fill at −1, skips at 0, and stops at 1 in each row of the grid from left to right in a straight line according to their codes (Fig.4). It is very easy to decide whether a cell is inside, outside or on the region boundaries for each isolated row in the grid when filling in this scanning line way. This makes it unnecessary to consider the whole grid simultaneously.

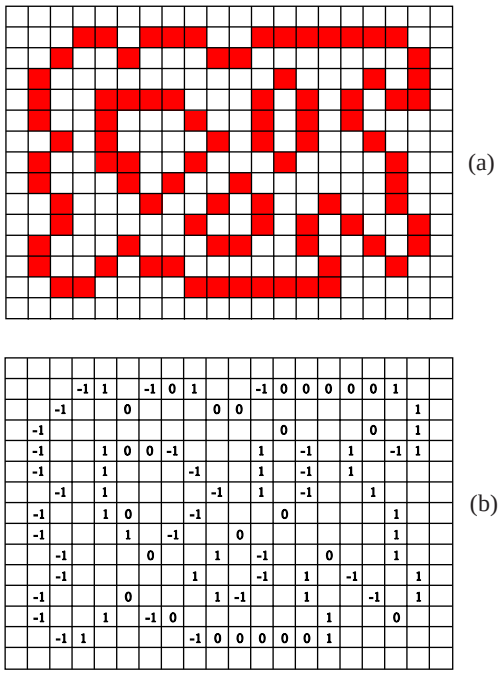


Fig.4 Codes for a complex region

(a) Initial region for labeling. The colored cells form the region boundaries. (b) Codes for the region in (a)

In practice, the data errors of the inputs, the relatively big discretization scale and some extended processing will lead to sharp handles or overlapping cases on the boundaries. This makes the encoding algorithm give a wrong description of the real region

boundaries. To deal with these problems we give the main codes enhancement strategies as follows.

1. Boundary overlapping. When interior boundaries overlap each other, accumulate the codes there and filter the sum by a sign function to obtain the new codes (Fig.5). When interior boundaries overlap the exterior one, use the exterior boundary codes as the new ones to ensure the priority of exterior boundaries.

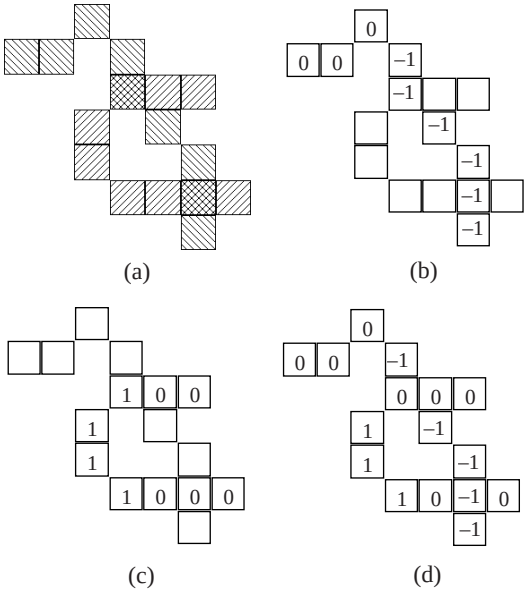


Fig.5 Interior boundaries overlapping case

(a) Left-bottom boundary filled with 45°-lines and right-top boundary filled with 135°-lines overlap at two cells filled with crossed lines; (b) Codes of the right-top boundary; (c) Codes of the left-bottom boundary; (d) Results obtained by summing the codes of these two boundaries

2. Sharp handles. The sharp handles case occurs when processing the elements with data errors. Cutting surgeries are necessary for providing the reliable codes. A two-step method is used to achieve this goal. First, record the position previous to the beginning of one sharp handle and the position next to the end of it. Next, recompute the codes at these two positions by themselves and set the middle ones to zero (Fig.6). Note in Fig.6a, to provide an intuitive illustration for the sharp handle cutting, the dashed, dotted, and solid arrows are used to demonstrate the directions which are followed when computing codes for upper element, lower element, and final reconnected boundary, respectively.

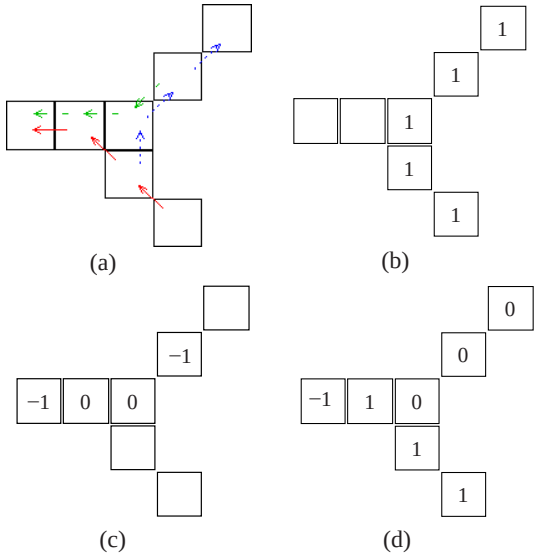


Fig.6 Sharp handles case

(a) A sharp handle where the dashed, dotted, and solid arrows are used to demonstrate the directions followed when computing codes for upper element, lower element, and final reconnected boundary, respectively; (b) Codes of the lower element; (c) Codes of the upper element; (d) The recomputed codes after reconnecting these two elements

3. Accumulating number. When interior boundaries spread out of the exterior one or boundary knots appear, one needs an accumulating number to guide searching. Initializing the number with 1, we accumulate it whenever the searching encounters a boundary code, start to count when this number becomes zero, and stop when it is non-zero. Moreover, when searching touches exterior boundary codes twice, reinitialize the number to 1.

Searching for the proper positions

With the codes gained in the above subsection, this subsection continues to discuss the searching strategy. Here we aim to locate a sub-region in the given region to label the multirow character information given by users. Besides, it is worth noting that for multirow characters, the layouts of the searching results need to be easy for human sight. Fig.7a gives an example of a bad layout. The words are laid separately, which makes it uncomfortable for human vision especially the blank row between ‘bad’ and ‘result’. On the contrary, Fig.7b shows the corresponding good layout. The words are laid collectively in the center-middle sub-region of R and lead to a continuous visual effect. Evidently, if all the valid

cells (i.e., the cells inside the discrete region Ω') are recorded, the optimization algorithm can be used to calculate the positions as a searching result. Considering the complexity, our searching method uses counting as its basic rule. This avoids recording the whole grid and provides more interactive schemes (searching under region, row amount, and/or character auto-adjusting constraints) for users. Details of our searching algorithm are described in the Appendix. In fact, searching under auto-adjusting and center-middle constraints with our method will give a visually pleasing result for common cases. A searching result is shown in Fig.7c. The characters occupy the cells which are the labeling results. The boundary of the part and the feature lines (their corresponding cells are also invalid) are kept untouched.

The boundary of the part is colored green and the feature lines (their corresponding cells are also invalid) are colored cyan. The red characters occupy the cells which are the labeling results. All the labeling result figures in this paper are shown in the same way.

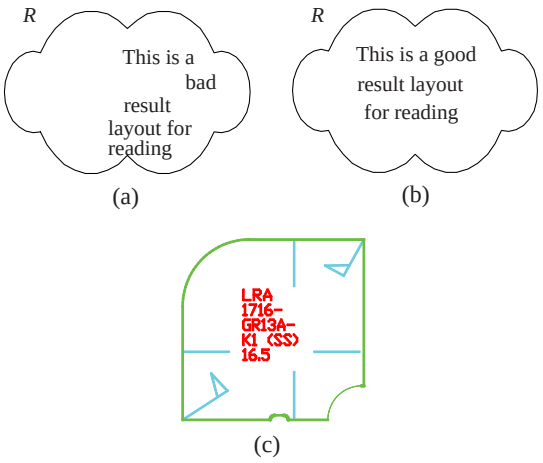


Fig.7 Labeling with visual effect considered
(a) A bad layout; (b) A good layout; (c) Searching result under auto-adjusting and center-middle constraints

Rotating the region to a new direction

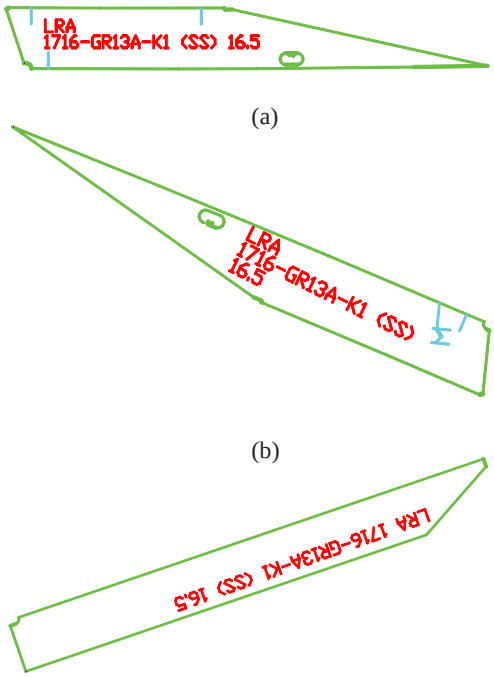
Basically the region for labeling is given in an arbitrary angle about the *x*-axis. Its optimal direction (which has the maximal region width) is the best choice for searching. The default horizontal searching will not always produce the exact result. In the non-horizontal case, we need to rotate the region, and then reapply the searching algorithm. The traditional method enumerates the possible cases to approximate this optimal direction by rotating the region by 1° and re-searching per time, which would be inefficient and lack precision. As our method discretizes the exterior boundary of the region to build a uniform grid, it can utilize the linear least square fitting technique (Bevington and Robinson, 1992; Draper and Smith, 1998). This is widely used for searching the best fitting line with respect to a sampled point set, to find the optimal direction with these cells in a straight line. Experiments show that at most five searching directions, namely horizontal, optimal, vertical, and bi-diagonal, are enough to calculate the final labeling result. Moreover, horizontal and optimal directions would be sufficient for most situations. To interpret this phenomenon, recall that the objective functional for optimizing may already reach the local minimum when the region is placed in horizontal or vertical directions, so fitting is trapped and is incapable of computing optimal directions in these situations. Applying the linear least square fitting to estimate the optimal direction reduces the rotating times and speeds up the algorithm. The opposite angle is outputted to guide the following computations.

RESULTS AND DISCUSSIONS

We have experimented FCBL with various input data gained from industrial factories with the format of DXF (file_name.dxf). The CPU time reported was taken on a Pentium (R) D 2.8 GHz processor with 1 GB of memory. In practice, codes of the boundaries and other features like lines in the region are considered simultaneously according to their priorities. We have tried multiple configurations with the testing data and gained satisfying results. With the most often used configuration, FCBL used 28.362 s to finish the labeling for over 3000 data. All results met the requirements. The processing time for a single part in our experiments was calculated as the average of the results gained from batch processing the same data 500 times.

Fig.8 shows the labeling results of three similar parts with different angles about the positive *x*-axis. The regions were searched horizontally first, and we obtained the result of Fig.8a without any rotation. Then, Figs.8b and 8c were rotated to their optimal directions and re-searched to find the final results. As we can see from Fig.8, the information characters

were labeled along the optimal directions of their corresponding parts successfully. And they were auto-adjusted into two, three, and one rows to avoid touching the boundaries or feature lines. The resulting layouts were suitable for human sight. All information characters were labeled inside the region without the uncomfortable case as demonstrated in Fig.7a.



(a)

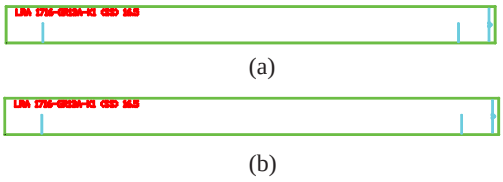
(b)

(c)

Fig.8 Results of similar data with different optimal directions

The optimal direction angles of the parts are: (a) 0°; (b) −26.7032°; (c) −160.268°

In Fig.9, the optional constraint, boundary reservation, was added into the processing. Boundary reservation means offsetting (Farouki and Neff, 1990; Pham, 1992) the boundaries along their exterior or interior normals by a constant distance. It is evident that the labeled characters in Fig.9b are closer to the top and left part boundaries than those in Fig.9a. During daily use or further manufacturing operations like cutting, the labeled characters in Fig.9a will have more durability in resisting wear originating from the part boundaries than those in Fig.9b. Furthermore, leaving more space to the part boundaries makes the labeled result possess a comfortable appearance for people.



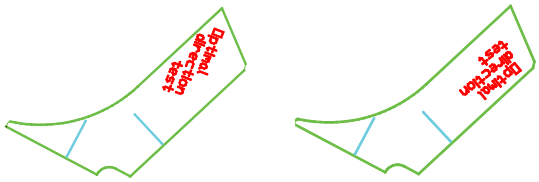
(a)

(b)

Fig.9 Results with (a) and without (b) boundary reservation

Fig.10 (see the next page) demonstrates the labeling results in a nesting layout. This is the most common case in which the labeling algorithm will be used. A global view of a set of parts with labeling results can make it convenient for result checking and machine painting, and hence save more time for production procedures. At first, the nesting layout was given, which means each part in the nesting layout was translated, rotated, or mirrored in the nesting step. At this time each part had different information characters, their production numbers, for labeling. All the parts were successfully labeled in a second using FCBL. Note in some parts the labeled characters touched the short feature lines, because we used the short feature line filtering constraint here.

Following the discussion in subsection ‘Rotating the region to a new direction’, we conducted experiments to study how the way we used to estimate the optimal direction influences the speed and accuracy of our labeling approach. A comparison example is given in Fig.11. The processing time was dramatically decreased in Fig.11b compared with that in Fig.11a. As the rotating times increased, least square estimation will save more and more time in this step. Furthermore, the estimated optimal direction in Fig.11b was more accurate than that in Fig.11a.



(a) (b)

Fig.11 Algorithm comparison for estimating optimal direction

(a) Rotating 1° per time estimation. Processing time: 25.2266 ms; optimal direction: −111°. (b) Least square estimation. Processing time: 3.4968 ms; optimal direction: −140.878°

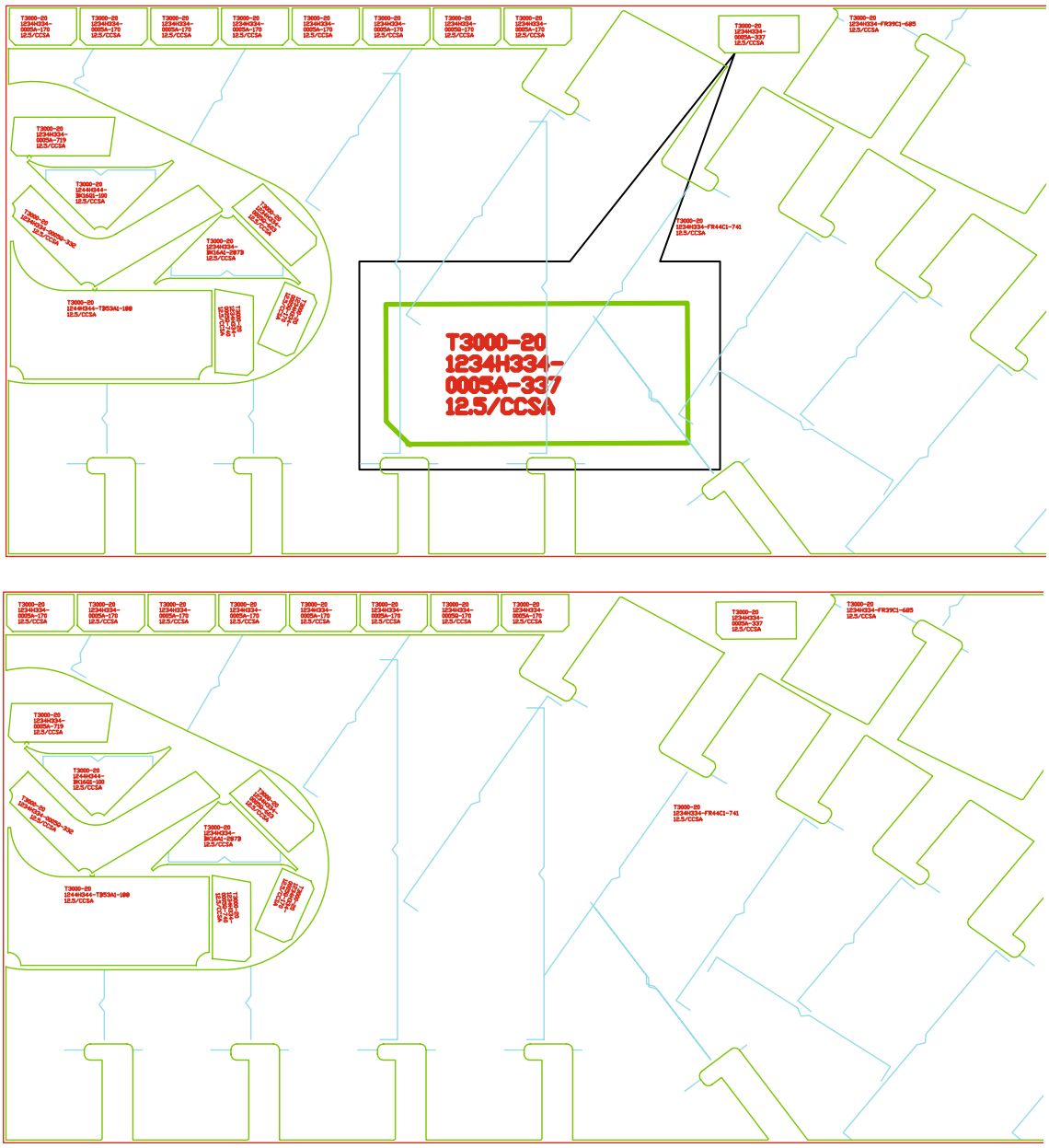


Fig.10 Labeling results on a nesting layout. The whole layout is divided into two for a better visual effect

Also we picked large scale data with different features for comparison under an identical configuration (Fig.12). As detailed in Table 1, size, grid, and the number of encoded cells of the parts influence the processing time to some extent.

To further investigate the influences of main factors in processing configurations, we gradually changed the corresponding factors during processing the picked data in Fig.12. When one factor was changing, all the other factors were fixed. The processing time under different numbers of the grid, encoded cells and information characters are shown in Fig.13. Fig.13a plots the processing time under different grid resolutions. The *x*-axis is the total number of cells in the grid, which was gained by multiplying grid width and grid height. In this case, the number of corresponding encoded cells increased with grid resolutions. Because FCBL did one more re-searching in processing D4 at 3500 and 4500, there was a protuberance in curve D4. Without that, curve

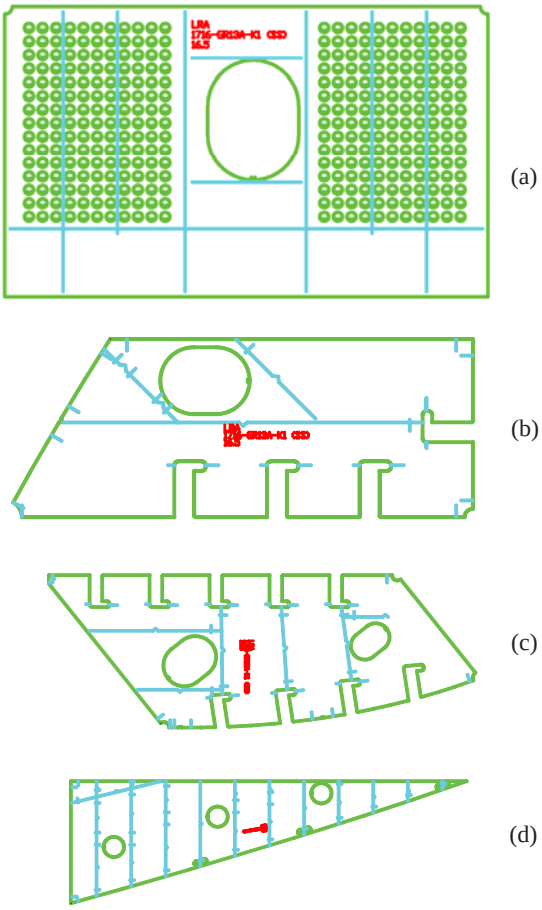


Fig.12 Results of large and complex data
(a) D1; (b) D2; (c) D3; (d) D4. The details of these data are given in Table 1

Table 1 Comparison of four large data

Data	Size (m×m)*	#Grid*	#Encoded	Time (ms)**
D1	3.16600×1.90609	51×30	1019	76.844
D2	4.14316×1.59750	66×25	374	6.052
D3	6.08507×2.19070	97×35	585	8.874
D4	9.57653×3.02457	153×48	785	9.984

* In a width×height format; ** The average processing time. #Encoded stands for the number of encoded cells in the grid

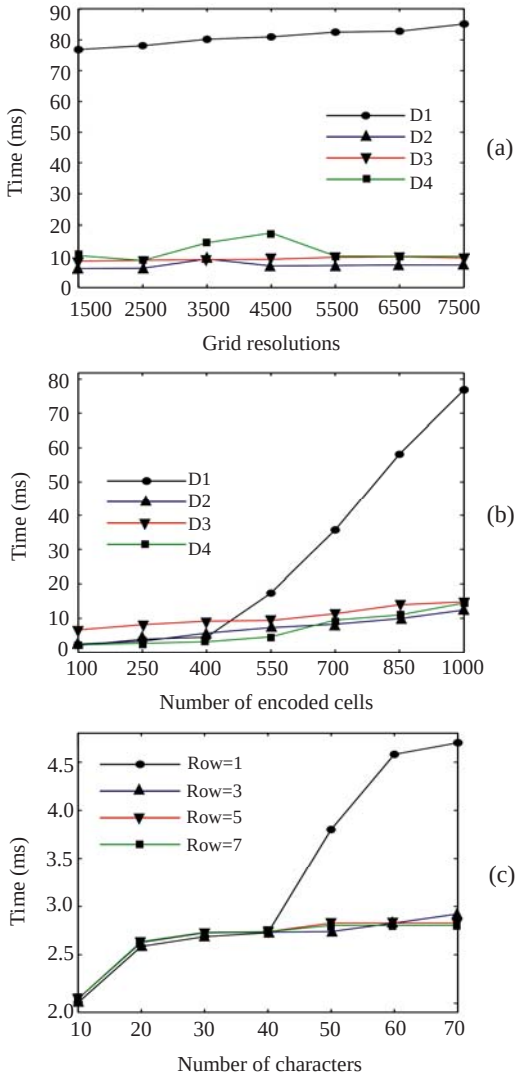


Fig.13 Relationship between processing time and the processing configurations
(a) Grid resolution; (b) The number of encoded cells; (c) The numbers of information characters with different auto-adjusted rows

D4 also increased as slowly as the others. Fig.13b shows how the number of encoded cells affects the time. Here the gird resolutions were fixed and the number of encoded cells was controlled by adding and removing interior boundaries and feature lines. Fig.13c illustrates the influence of a possible factor, the number of information characters, on processing time. The information characters were auto-adjusted into multiple rows to test the influence of different numbers of rows simultaneously. This test was built on the simplest version of D1 in Fig.12a with all interior boundaries and feature lines removed. Because the information characters were too long to be put in one row, one more re-searching was added to obtain the final result, and hence led to a big rise on the right side of the first curve (row=1). Our statistical data demonstrate that the most important factor on which FCBL depends is the number of encoded cells, i.e., the complexity of the target region.

CONCLUSION

An automatic algorithm FCBL is presented for solving the industrial labeling problem. It takes advantages of some techniques in computer graphics and improves them to fit specific problems. The use of code-based representation leads to a conceptually simple and effective method for intuitive labeling. Furthermore, a rapid optimal direction estimation reduces unnecessary computation cost by avoiding the tedious enumeration. Experiments demonstrate the fast speed and visually pleasing effect of FCBL.

Currently, FCBL has been developed using C++ and integrated into an industrial application, and the stable performance in actual manufacture validates its worth. In the future we would like to extend this technique for more complicated problems like the large-curvature with thin portion case. We also plan to improve the effect and efficiency of this work by including more geometric operations.

References

Alvim, A.C.F., Taillard, E.D., 2009. POPMUSIC for the point feature label placement problem. *Eur. J. Oper. Res.*, **192**(2):396-413. [doi:10.1016/j.ejor.2007.10.002]

Al-Assaf, Y., 2003. Human strategies based allocation of two-dimensional irregular shapes. *J. Intell. Fuzzy Syst.*, **14**(4):181-190.

Bennell, J.A., Oliveira, J.F., 2008. The geometry of nesting problems: a tutorial. *Eur. J. Oper. Res.*, **184**(2):397-415. [doi:10.1016/j.ejor.2006.11.038]

Bevington, P.R., Robinson, D.K., 1992. Data Reduction and Error Analysis for the Physical Sciences. WCB/McGraw-Hill, Boston.

Binucci, C., Didimo, W., Liotta, G., Nonato, M., 2005. Orthogonal drawings of graphs with vertex and edge labels. *Comput. Geom.*, **32**(2):71-114. [doi:10.1016/j.comgeo.2005.02.001]

Cravo, G.L., Ribeiro, G.M., Lorena, L.A.N., 2008. A greedy randomized adaptive search procedure for the point-feature cartographic label placement. *Comput. & Geosci.*, **34**(4):373-386. [doi:10.1016/j.cageo.2007.01.007]

Dogrusoz, U., Kakoulis, K.G., Madden, B., Tollis, I.G., 2007. On labeling in graph visualization. *Inf. Sci.*, **177**(12):2459-2472. [doi:10.1016/j.ins.2007.01.019]

Draper, N.R., Smith, H., 1998. Applied Regression Analysis. John Wiley & Sons, New York.

Egeblad, J., Nielsen, B.K., Odgaard, A., 2007. Fast neighborhood search for two- and three-dimensional nesting problems. *Eur. J. Oper. Res.*, **183**(3):1249-1266. [doi:10.1016/j.ejor.2005.11.063]

Farouki, R.T., Neff, C.A., 1990. Analytic properties of plane offset curves. *Comput. Aided Geom. Des.*, **7**(1-4):83-99. [doi:10.1016/0167-8396(90)90023-K]

Freeman, H., 1961. On the encoding of arbitrary geometric configurations. *IEEE Trans. Electron. Comput.*, **10**(2):260-268. [doi:10.1109/TEC.1961.5219197]

Hearn, D., Baker, M.P., 1997. Computer Graphics (C Version). Prentice Hall, New Jersey.

Henrich, D., 1994. Space-efficient region filling in raster graphics. *The Vis. Comput.*, **10**(4):205-215. [doi:10.1007/BF01901287]

Kakoulis, K.G., Tollis, I.G., 2006. Algorithms for the multiple label placement problem. *Comput. Geom.*, **35**(3):143-161. [doi:10.1016/j.comgeo.2006.03.005]

Lan, X., Jiang, Y., Lü, G., Deng, H., 2005. Automatic placement of GIS vector map annotation in area feature by long-diagonal. *Geo-Spat. Inf. Sci.*, **8**(4):276-281. [doi:10.1007/BF02838662]

Lee, W.C., Ma, H., Cheng, B.W., 2008. A heuristic for nesting problems of irregular shapes. *Comput.-Aided Des.*, **40**(5):625-633. [doi:10.1016/j.cad.2008.02.008]

Liang, L., Ye, J., 2008. A Solution of Irregular Parts Nesting Problem Based on Immune Genetic Algorithm. Int. Symp. on Computational Intelligence and Design, p.217-220. [doi:10.1109/ISCID.2008.25]

Mote, K., 2007. Fast point-feature label placement for dynamic visualizations. *Inf. Visual.*, **6**(4):249-260. [doi:10.1057/palgrave.ivs.9500163]

Pham, B., 1992. Offset curves and surfaces: a brief survey. *Comput.-Aided Des.*, **24**(4):223-229. [doi:10.1016/0010-4485(92)90059-J]

Ramesh Babu, A., Ramesh Babu, N., 1999. Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms. *Int. J. Prod. Res.*, **37**(7):1625-1643. [doi:10.1080/002075499191166]

Ramesh Babu, A., Ramesh Babu, N., 2001. A generic approach for nesting of 2-D parts in 2-D sheets using genetic and heuristic algorithms. *Comput.-Aided Des.*, **33**(12):879-891. [doi:10.1016/S0010-4485(00)00112-3]

Tang, G.Y., Lien, B., 1988. Region filling with the use of the discrete green theorem. *Comput. Vis. Graph. Image Process.*, **42**(3):297-305. [doi:10.1016/S0734-189X(88)80040-9]

Wolff, A., 2009. The Map Labeling Bibliography. University of Karlsruhe, Karlsruhe, Germany. Available from http://illwww.iti.uni-karlsruhe.de/map-labeling/bibliography/ [Accessed 19/08/09].

Wu, T.H., Chen, J.F., Low, C., Tang, P.T., 2003. Nesting of two-dimensional parts in multiple plates using hybrid algorithm. *Int. J. Prod. Res.*, **41**(16):3883-3900. [doi:10.1080/0020754031000149239]

Yang, H.H., Lin, C.L., 2009. On genetic algorithms for shoe making nesting: a Taiwan case. *Exp. Syst. Appl.*, **36**(2):1134-1141. [doi:10.1016/j.eswa.2007.10.043]

APPENDIX

The searching algorithm under region and multi-row constraints (searching order is top-to-bottom and left-to-right) is shown as follows (others are parallel).

Variables declaration

district: a rectangular sub-region in the given region where the searching is constrained.

$$district.\begin{cases} left, & \text{Column index of the leftmost cell in } district, \\ right, & \text{Column index of the rightmost cell in } district, \\ top, & \text{Row index of the topmost cell in } district, \\ bottom, & \text{Row index of the bottommost cell in } district. \end{cases}$$

seg: a $1\times h$ ($h\geq 1$) rectangular sub-region where the first character row will be labeled. This variable is important for providing visually pleasing results in our algorithm. Sub-regions for searching the residual character rows are changed following this variable.

$$seg.\begin{cases} begin\_pos, & \text{Column index of the beginning cell in } seg, \\ end\_pos, & \text{Column index of the ending cell in } seg, \\ row\_id, & \text{Row index of } seg. \end{cases}$$

cur_row: the row index of the current row (the grid row which is being searched).

real_end: the column index of the terminating cell in the current row.

count: the number of the continuous valid cells in the current row. Once the algorithm meets an invalid cell in searching, *count* is reset to 0.

cur_len: length of the current character row (the character row which needs to be labeled).

need_to_move: length of the residual characters in the current character row which have no place to be labeled. It is used when the algorithm steps into left shifting. Its value can be computed by *need_to_move=cur_len−count*.

Algorithm steps

1. Initialization. Input *district* for searching. Bound *cur_row* between *district.top* and *district.bottom*. Set *seg.begin_pos=district.left*, *seg.end_pos=district.right*, *seg.row_id=district.top*, and *real_end=district.right*.

2. Basic checking. Accumulate *count* from *seg.begin_pos* to check whether there are *cur_len* continuous valid cells in the *cur_row* row. Yes, i.e., *count*==*cur_len*, go to step 5. No, if 0<*count*<*cur_len*, go to step 3; if *count*==0 for the first time here, go to step 4.

3. Left shifting. Check the *need_to_move* cells on the left side of *seg.begin_pos*. If *count*==*cur_len* (i.e., all the checked cells are valid), go to step 5; if *count*==0, go to step 4.

4. Right shifting. Keep to shift right one cell to accumulate *count* (remain accumulate if *count*>0 at *real_end*). If *count*==0 at *real_end* (or at any cell right to *real_end*), set *cur_row=cur_row*+1 and redo steps 2~4. If *count*==*cur_len*, go to step 5.

5. Recording and updating. Record *cur_len* and the beginning position of the *cur_len* cells. Set *cur_row=cur_row*+1. For the first character row, update *seg* and *real_end* by the beginning and ending positions of the *cur_len* cells, respectively. Continue to search the next character row, and redo steps 2~4.

6. Outputting. During the whole flow, if all character rows can be labeled, return ‘true’; if one of the character rows cannot be labeled, set *cur_row*=*cur_row*+1 and redo this flow from searching with respect to the first character row; if searching reaches *district.bottom* or the number of the residual rows in *district* is smaller than that of the residual character rows, return ‘false’.

The auto-adjusting constraint can be achieved by adding the residual characters to the current character row and moving the unlabeled ones to the next character row during the searching.