Computers, Materials & Continua CMC, vol.63, no.1, pp.509-519, 2020

A Fast Filling Algorithm for Image Restoration Based on Contour Parity

Yan Liu[1, *], Wenxin Hu[1, *], Longzhe Han[2], Maksymyuk Taras[3] and Zhiyun Chen[1]

Abstract: Filling techniques are often used in the restoration of images. Yet the existing filling technique approaches either have high computational costs or present problems such as filling holes redundantly. This paper proposes a novel algorithm for filling holes and regions of the images. The proposed algorithm combines the advantages of both the parity-check filling approach and the region-growing inpainting technique. Pairing points of the region's boundary are used to search and to fill the region. The scanning range of the filling method is within the target regions. The proposed method does not require additional working memory or assistant colors, and it can correctly fill any complex contours. Experimental results show that, compared to other approaches, the proposed algorithm fills regions faster and with lower computational cost.

Keywords: Region filling, image restoration, parity check, region growing.

1 Introduction

Region filling is important in computer graphics and image processing. For a given region boundary (or no boundary, just given the specified color), all pixels within the boundary range are modified to the specified color or pattern. First, the boundary of the region is obtained, and then the algorithm fills all the connected pixels of the region. For vector data structures, region filling is usually used as a representation of graphical output. After the vector line segment is constructed into a polygon, the region is filled.

There are two ways to render a target scene: raster images and vector images [Marius and Olli (2005)]. Raster graphics are matrices of the pixels, which are usually used in bitmaps. Vector graphics are also represented in raster, but they are produced by a set of geometric descriptions of shapes, or vectors, which are easier to scale and transfigure than raster graphics. Vector graphics also require much less storage space.

Popular vector representations of binary images include quad-trees [Charles, Azriel and Hanan (1980)], chain code [Ernesto (1999)], run-length code [Kim, Lee and Kim (1988)]

[1] School of Data Science and Engineering, East China Normal University, Shanghai, 200062, China.

[2] JiangXi Province Key Laboratory of Water Information Cooperative Sensing and Intelligent Processing, Nanchang Institute of Technology, Nanchang, 330099, China.

[3] Department of Telecommunication Institute of Telecommunication, Radio Electronics and Electronic Engineering, Lviv Polytechnic National University, Lviv, 79013, Ukraine.

[*] Corresponding Author: Wenxin Hu. Email: wxhu@cc.ecnu.edu.cn.

Received: 30 May 2019; Accepted: 16 July 2019.

and bin-code [Yao and Kuo (2000)]. They are obtained from raster graphics through a procedure called raster-vector transformation, while region filling is the inverse operation.

Liu proposed an efficient quantum perceptron algorithm based on the unitary weights [Liu, Gao, Liu et al. (2019)], which can realize basic quantum gate functions [Liu, Gao, Yu et al. (2018)]. The feature selection algorithm used in binary classification can help a user retrieve items from a database [Liu, Gao, Wang et al. (2019)]. They studied the effect of quantum noise on deterministic state preparation [Qu, Wu, Wang et al. (2017)] and designed a method based on quantum image expansion [Qu, Li, Xu et al. (2019)] and Grover search algorithm [Qu, Cheng and Wang (2019)].

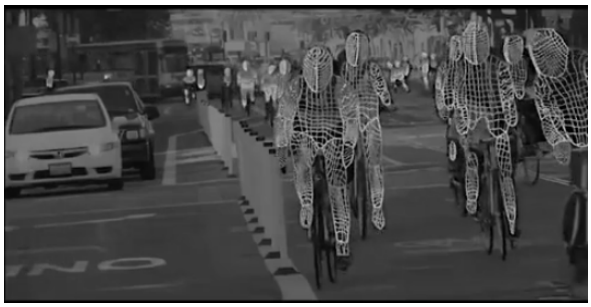

Figure 1: Scene in the computer vision

Pavlidis proposed a filling algorithm of Freeman code [Theo (1979)]. Bertalmio [Marcelo, Guillermo, Vicent et al. (2002)] proposed a method in which the information was propagated through the edge of the contour line in the occlusion area. Based on the idea of Bertalmio, Chan et al. [Chan and Shen (2001)] proposed a novel inpainting model based on the diffusion mechanism. However, the PDE model they adopted was merely suitable for small regions. To solve large region problems, Efros [Alexei and Thomas (1999)] proposed a non-parametric method for texture synthesis, which grew a new image from an initial seed. Recently, deep learning models have been applied to image completion. Convolutional neural network (CNN) [Deepak, Philipp, Jeff et al. (2016)] was used to generate the contents according to its surroundings, which become a novel solution for filling a region. However, CNN method relies on the data set.

The rest of this paper is organized as follows: Section 2 discusses related work; Section 3 describes the proposed method and its framework; Section 4 focuses on experimental results used to verify the proposed approach; and Section 5 concludes the paper.

2 Related work

Generally, filling algorithms are composed of two classes: parity check filling (edge or scan-line filling) such as method in Chang et al. [Chang and Leu (1990)] and [Frank and Wong (1994)], [Cai (1988)] and seed filling (also named connectivity filling or region growing). In parity check filling, a scanning line is drawn from the current pixel to the boundaries. Let n be the number of times the line intersects the boundaries. The pixel is an interior point if n is odd or otherwise is an exterior point. Seed filling begins from interior point (seed) and extends to its neighbor. The operation repeats until the border is met. Seed filling is able to fill complex contours. But selecting suitable seeds is difficult. And

searching for all interior pixels costs much time and large memory.

The method of Cai [Cai (1988)] was an early proposed parity check filling algorithm. The algorithm classified a contour point into *singular point*, *marking point* and "*no direction combination occurs*" according to its reaching codes and leaving codes. The method can fill images correctly.

Chang et al. [Chang and Leu (1990)] was another filling algorithm of parity check. In addition to the point classification defined in Cai's method, Chang named the *unsuitable point* to represent points inside the contour, indicating the occurrence of wrong code description. Chang's method used Merrill's y-axis data structure [Roy (1973)], which is an ascending (y, x) list, to store the marked points and singular points [Frank and Wong (1994)].

A typical seed filling algorithm was proposed in the work of Tang et al. [Tang and Lien (1988)]. Contour points was classified into *left point*, *right point* and x-tip point according to its reaching codes and leaving codes. The algorithm was correct and efficient.

Presently, Ren proposed a very simple and effective filling algorithm [Ren, Yang and Yang (2001)]. The method decomposed the contour into sub-contours. A sub-contour was named simple closed contour, whose contour points (colored with *boundary-color*) is passed only once. Once the *filling starting point* was found, the interior points in the same line was filled rightward until the *filling ending point* is met. The method is effective but couldn't fill internal contours.

Ren proposed the improved version in Ren et al. [Ren, Yang and Yang (2005)] later. Contour points were classified into *filling starting point*, *filling ending point* and *filling skipping point*, according to *sum_delta_y* (the sum of y-offsets of its reaching codes and leaving codes). The *filling starting point* on an inner contour lies on the right side of the boundary while the *filling starting point* on an outer contour lies on the left side of the boundary.

Ren's method is fast in finding seeds automatically and its time consumption is nearly constant without considering the shape of contours. But redundant filling will occur in the case of region with holes, where the filling of holes will overlap. As Fig. 2 illustrates, assuming *Area1* is the region between an outer contour and an inner contour, *Area2* is the region enclosed by the inner contour. N_{out} and N_{in} denote the number of outer contour codes and inner contour codes, respectively. To obtain correct filling result, Ren's method has to fill the region *Area1+Area2* first, and then fill the region *Area2*. It can be seen that *Area2* is filled twice, thus results in redundant filling.

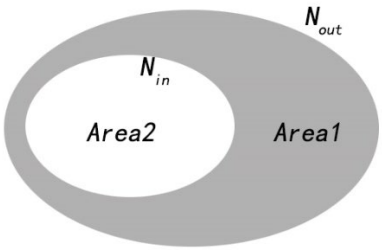

Figure 2: Region with a hole

3 The novel filling algorithm based on contour parity

To improve filling efficiency, we developed a technique using pairing points of the region's boundary in this work.

In Fig. 2, *Area1* should be filled and *Area2* should be skipped. According to our analysis, the contour points are classified into left point (L), right point (R) and duple point (LR). Their definitions are as follows:

L point: There is a region to be filled on its right in the same row.

R point: There is a region to be filled on its left in the same row.

LR point: The boundary has only one pixel in the current row, so the point is both L point and R point.

That is, whether a boundary point is L point, R point or LR point mainly depends on the direction of the contour.

Chain code is a typical image representation which includes the information of contour direction. And the proposed algorithm used Freeman chain code for the experiments. The contour tracing obeys the right-edge tracing rule [Theo (1979)], tracing inner contours clockwise and outer contours counterclockwise. Both inner and outer contour lines are conventionally 8-connected.

Liu designed a Freeman chain coding's status transition of eight directions [Liu and Gu (2014)]. In the transition, two consecutive codes (location *a* and location *b*) are used to judge the contour direction and tracing status, as shown in Fig. 3.

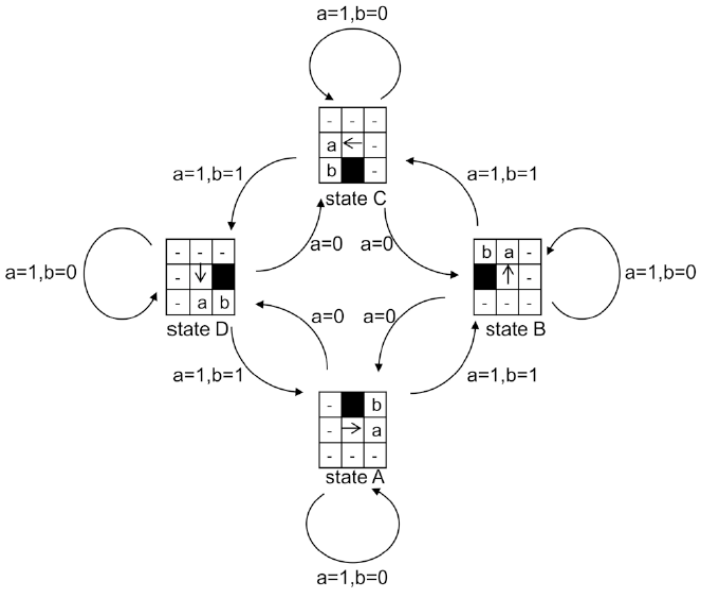

Figure 3: Status transition of Freeman chain codes

The chain codes and the L/R/LR points correspond in direction. Therefore, we can use two consecutive codes C_i and C_{i+1} to obtain contour point type. The chain coding-outline lookup table is shown in Tab. 1 below.

Table 1: The L/R/LR point lookup table

C_i \ C_{i+1}	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	×	R	R	R	R	—	—	×
1	×	R	R	R	R	LR	—	×
2	—	R	R	R	R	LR	LR	—
3	—	R	R	R	R	LR	LR	LR
4	L	—	—	×	×	L	L	L
5	L	LR	—	×	×	L	L	L
6	L	LR	LR	—	—	L	L	L
7	L	LR	LR	LR	—	L	L	L

It is important to note that the number of L points is equal to the number of R points in each row. The sum of the inner contour points is exactly the same as the outer contour points.

In every scanning row the filling begins from left points to the nearest right point. This way holes inside a region are skipped, and redundant filling can be avoided.

The process of the proposed algorithm is:

Step 1: Scans each contour including inner and outer contours and finds R and LR points. The points are marked with *fill-color*, as shown in Figs. 4(a) and 4(b).

Step 2: Scans every contour again. Finds all L points, marks them, and then fills rightward to the nearest marked R points., as shown in Figs. 4(c) and 4(d). LR points are skipped because they are already marked in Step 1. The algorithm terminates when all the contour pixels have been filled.

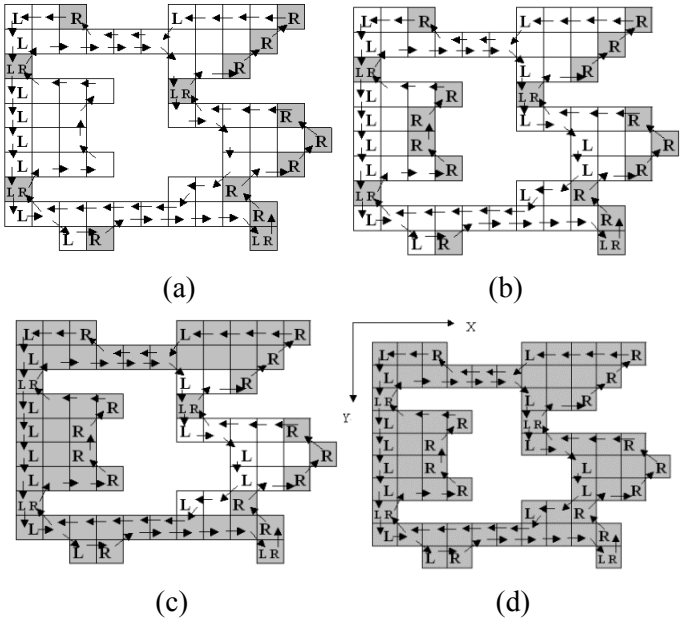

Figure 4: Filling procedure for a complex region

Conventional algorithms such as Ren's method have to fill contour points after all the interior pixels are filled. In contrast, the proposed method fills the contour pixels during the scan of the contour, which saves time.

The proposed method can be used to tackle regions presented by all kinds of chain code, such as Freeman chain code, crack code and vertex chain code. The algorithm based on Freeman chain code is implemented in Fig. 5 below.

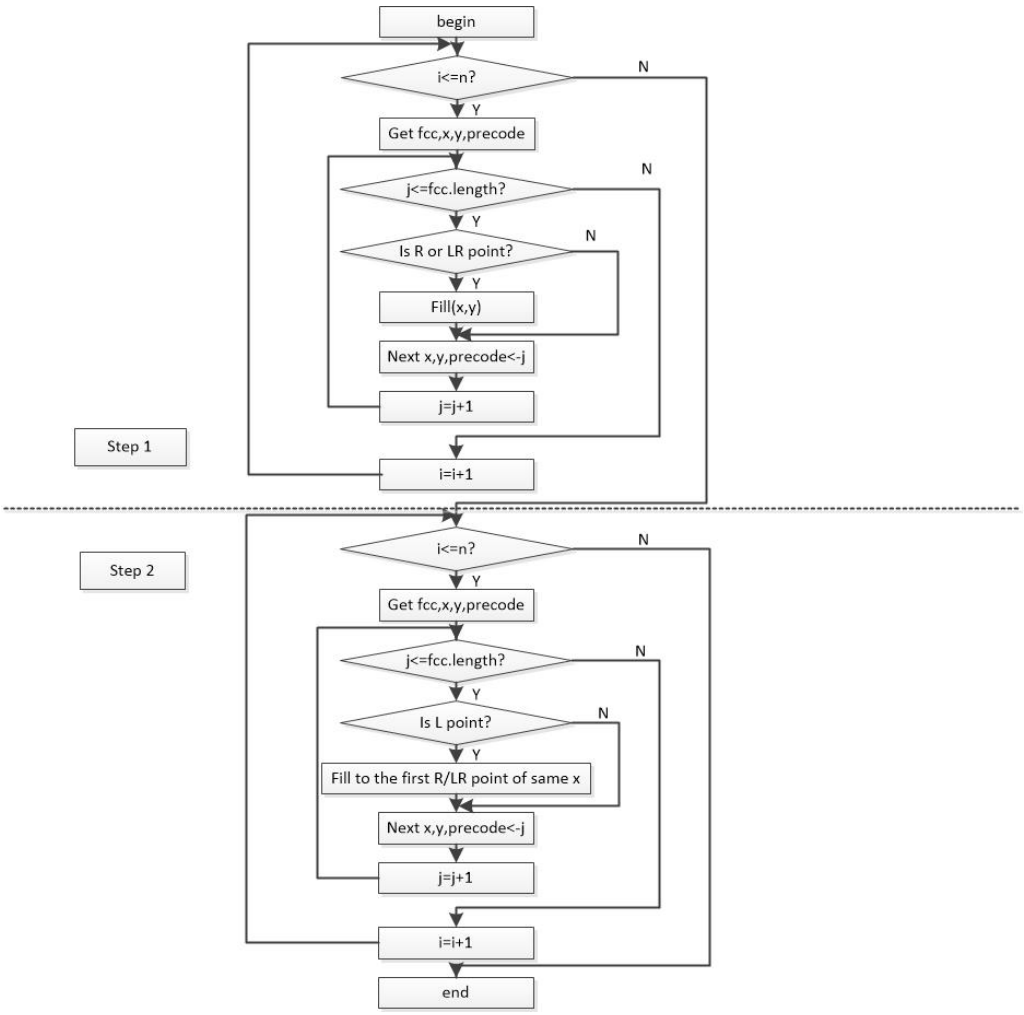

Figure 5: Filling method based on Freeman chain code

4 Experimental procedure

4.1 The analysis and comparisons of the methods

We conducted further experiments to compare the proposed method with a range of existing methods focusing on total time complexity as well as auxiliary memory and assisted color. We compared the proposed method with Cai's method, Chang's method, Ren's method, and Tang's method.

Since a contour point may be met at most four times during the operation, Cai [Cai (1988)] uses two workspaces of the same size as the original image, named A and B, to avoid incorrect classification. A stores marked points and B stores single points. Then lines between marked points in A are filled. Finally, the single points in B are added to A to produce the filling result. Time consumption in scanning the whole circum-rectangle (including filling) is $W \times H$ and the time of visiting the contour is $O(N)$. Therefore, the total time consumption is $W \times H + O(N)$. Both single and marked points used the same filling color, no assistant color was used.

According to the Chang's algorithm [Chang and Leu (1990)], a single point is recorded twice, a marked point is recorded once and a skipping point is not recorded. Furthermore, a single point will not be assigned as a marked point when it is visited the second time. Owing to the filling point list, the method avoids to scan pixels outside the contour. Time consumption of filling is Area. Sorting the array costs $\mathrm{O(NlogN)}$. The total time consumption is $\mathrm{Area} + \mathrm{O(NlogN)}$. No assistant color is used to distinguish pixels.

Tang's method [Tang and Lien (1988)] uses filling color I, and three assistant colors B, E and O. It first labeled boundary points in image A. Then an array T is allocated to store the processed row. During the filling, the method searches for seeds in each row to decide whether it should be labeled. The array not only stores boundary runs in each row, but also needs to sort them ascending by their coordinates. Time consumption of scanning the circum-rectangle (sometimes more than once) is $(1+\alpha)\mathrm{W} \times \mathrm{H}, \alpha > 0$, the time of the filling consumption is Area, and the time of determining seeds is $\mathrm{O(k \times N)}$, where k is the time required to find seeds and is uncertain. Thus the total time consumption of the Tang's filling method is $(1+\alpha)\mathrm{W} \times \mathrm{H} + \mathrm{Area} + \mathrm{O(k \times N)}, \alpha > 0$.

Ren's algorithm scans the contour five times. It colors all contour points in the first time, and calculates sum_delta_y of each contour point in the second time. It marks all points to fill in the third time and fills the pixels from start point to end point in the fourth time, finally fills in all contour points.

Time consumption of Ren's method in scanning pixels in the region is Area, and the time of visiting the contour is $\mathrm{O(N)}$, thus the total time consumption is $\mathrm{Area} + \mathrm{O(N)}$. For the target image, the algorithm visited the contour five times, so the time is $\mathrm{Area} + 5(\mathrm{N})$.

Ren's method is fast in finding seeds automatically, and its time consumption is nearly constant. But it handles each contour individually, redundant filling will occur in the case of region with holes, where the filling of holes will overlap. Take Fig. 2 as example, to obtain correct filling result, Ren has to fill the region $Area1 + Area2$ and fill the region $Area2$. The total time consumption is $Area1 + 2Area2 + 5N_{out} + 5N_{in}$. It can be seen that $Area2$ is filled twice, thus resulting in redundant filling.

In contrast to Ren's method, the proposed algorithm only visits the contour twice and scans the interior pixels (skipping holes) once, thus the total time consumption is $Area + 2(N)$. For the case illustrated in Fig. 2, the total time consumption is $Area1 + 2N_{out} + 2N_{in}$. In addition, because L and R points are marked with *fill-color* during the operation, no assistant color is used.

The time complexity of the four existing methods is compared to the proposed method in Tab. 2.

Table 2: Time complexity comparison

Method	Time complexity	Auxiliary memory	Auxiliary colors
Cai's method	$W \times H + O(N)$	$2 \times W \times H$	0
Chang's method	$Area + O(NlogN)$	0	0
Tang's method	$(1+\alpha)W \times H + Area + O(k \times N), \alpha > 0$	N	3
Ren's method	$Area + 5N$	0	3
Proposed method	$Area + 2N$	0	0

4.2 Experimental results

Our experiment takes Figs. 6(a)-6(c) and Fig. 7(a) as test images. And Fig. 7(b) is the filling result of Fig. 7(a). Because Ren's method has better performance than other established filling techniques, the experiments compare Ren's method and the proposed method, shown in Tab. 3.

(a) (b) (c)

Figure 6: Three test images

(a) (b)

Figure 7: Large complex image and the filling result

The average time of running the program 100 times is taken as the result, which shows that the proposed method outperforms Ren's method. Since Ren's method outperforms other methods,

one can deduce that the proposed method is faster than already-established approaches.

Table 3: Time comparison of the algorithms

Test image	Size	Ren's method (ms)	Proposed method (ms)
Fig. 6(a)	339×159	1792	1108
Fig. 6(b)	165×154	780	619
Fig. 6(c)	596×371	2741	1436
Fig. 7(a)	548×733	5424	2121

Because the proposed algorithm has no need to fill the holes in the image, it appears faster than Ren's method when the image has numerous large holes. Furthermore, it does not use additional memory or assistant color.

5 Conclusion

This paper proposes a novel filling method based on pixel parity. The proposed method presents a simple idea to find pairs of contour points for filling, which limits the scan within the region to be filled and does not fill holes redundantly. Further, no assistant color is used. Experiments results show that the time consumption of the proposed method is lower, regardless of the contour's shape. Experiments results also demonstrate that the proposed algorithm can fill complex contours with higher speed and less computational cost.

Acknowledgement: The research is jointly supported by the National Natural Science Foundation of China No. 61561035, and by Ukrainian government project No. 0117U007177 and the Slovak Research and Development Agency project number APVV-18-0214.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Alexei, E.; Thomas, L. (1999): Texture synthesis by non-parametric sampling. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1033-1038.

Cai, Z. G. (1988): Restoration of binary images using contour direction chain codes description. *Computer Vision, Graphics, and Image Processing*, vol. 41, no. 1, pp. 101-106.

Chan, T. F.; Shen, J. (2001): Non-texture inpainting by curvature-driven diffusions. *Journal of Visual Communication and Image Representation*, vol. 12, no. 4, pp. 436-449.

Chang, L. W.; Leu, K. L. (1990): A fast algorithm for the restoration of images based on chain codes description and its application. *Computer Vision, Graphics, and Image Processing*, vol. 50, no. 3, pp. 296-307.

Charles, R. D.; Azriel, R.; Hanan, S. (1980): Region representation: boundary bodes from quadtree. *Communications of the ACM*, vol. 23, no. 3, pp. 171-179.

Deepak, P.; Philipp, K.; Jeff, D.; Trevor, D.; Alexei, A. E. (2016): Context encoders:

feature learning by inpainting. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536-2544.

Ernesto, B. (1999): A new chain code. *Pattern Recognition*, vol. 32, pp. 235-251.

Frank, Y. S.; Wong, W. T. (1994): An improved fast algorithm for the restoration of images based on chain codes description. *Computer Vision, Graphics, and Image Processing*, vol. 56, no. 4, pp. 348-351.

Kim, S. D.; Lee, J. H.; Kim, J. K. (1988). A new chain-coding algorithm for binary images using run-length codes. *Computer Vision, Graphics, and Image Processing*, vol. 41, no. 1, pp. 114-128.

Liu, W. J.; Gao, P. P.; Liu, Z.; Chen, H.; Zhang, M. (2019): A quantum-based database query scheme for privacy preservation in cloud environment. *Security and Communication Networks*, vol. 14.

Liu, W. J.; Gao, P. P.; Wang, Y.; Yu, W. B.; Zhang, M. (2019): A unitary weights based one-iteration quantum perceptron algorithm for non-ideal training sets. *IEEE Access*, vol. 7, pp. 36854-36865.

Liu, W. J.; Gao, P. P.; Yu, W. B.; Qu, Z. G.; Yang, C. N. (2018): Quantum relief algorithm. *Quantum Information Processing*, vol. 17.

Liu, Y.; Gu, G. Q. (2014): Efficient algorithm for obtaining connected components in bi-level images. *IEICE Electronics Express*, vol. 11, no. 3, pp. 20130748-20130748.

Marcelo, B.; Guillermo, S.; Vincent, C.; Coloma, B. (2002): Image inpainting. *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, vol. 27, pp. 417-424.

Marius, C. C.; Olli, S. N. (2005): Note: an algorithm for contour-based region filling. *Computers and Graphics*, vol. 29, no.3, pp. 441-450.

Qu, Z. G.; Cheng, Z. W.; Wang, X. J. (2019): Matrix coding-based quantum image steganography algorithm. *IEEE Access*, vol. 7, pp. 35684-35698.

Qu, Z. G.; Li, Z. Y.; Xu, G.; Wu, S. Y.; Wang, X. J. (2019): Quantum image steganography protocol based on quantum image expansion and Grover search algorithm, *IEEE Access*, vol. 7, pp. 50849-50857.

Qu, Z. G.; Wu, S. Y.; Wang, M. M.; Sun, L.; Wang, X. J. (2017): Effect of quantum noise on deterministic remote state preparation of an arbitrary two-particle state via various quantum entangled channels. *Quantum Information Processing*, vol. 16, no. 306, pp. 1-25.

Ren, M. W.; Yang, J. Y.; Sun, H. (2001): A new contour filling algorithm based on chain codes description. *Journal of Image and Graphics*, vol. 4, pp. 348-352.

Ren, M. W.; Yang, W. K.; Yang, J. Y. (2005). A new and fast contour-filling algorithm. *Patten Recognition*, vol. 38, no. 12, pp. 2564-2577.

Roy, D. M. (1973). Representation of contours and regions for efficient computer search. *Communications of the ACM*, vol. 16, no. 2, pp. 69-82.

Tang, G. Y.; Lien, B. (1988): Region filling with the use of the discrete green theorem. *Computer Vision, Graphics, and Image Processing*, vol. 42, no. 3, pp. 297-305.

Theo, P. (1979). Filling algorithms for raster graphics. *Computer Graphics and Image*

Processing, vol. 10, no. 2, pp. 126-141.

Yao, H.; Kuo, L. (2000): Region-filling algorithm on bincode-based contour and its implementation. *Computers and Graphics*, vol. 24, no. 4, pp. 529-537.