アルゴリズム 66−12
（1999. 1. 27）

Repairing Flaws in a Picture Based on a Geometric Representation of a Digital Image

浅野哲夫[1]，伊藤大雄[2]，木村宗市，嶋津茂昭[3]

[1]北陸先端科学技術大学院大学（石川県辰口町）
[2]豊橋技術科学大学（豊橋市）
[3]大日本スクリーン製造（京都市）

高精細印刷においては，入力画像に含まれる微少なキズの修復が要求されることが多い．周辺の模様をコピーすることによりキズ領域を書き換えるコピーブラッシと呼ばれる操作によって修復することも可能であるが，非常に精密で根気のいる操作であるので自動化が望まれている．このような処理を高速に実現するには従来の行列表現ではなく，画像を地形図のように考えて等高線で表現する方法が適していると考えられる．この方法では，キズ領域と交差する等高線をいかに自然に繋ぎ直すかが問題である．等高線の再連結に必要な経路の長さを最小にする連結パターンを求めるための効率の良いアルゴリズムを提案する．鍵になるのは，最小重みの完全マッチングと連結経路の幾何的性質である．

Repairing Flaws in a Picture Based on a Geometric Representation of a Digital Image

Tetsuo Asano[1] Hiro Ito[2], Souichi Kimura and Nariaki Shimazu[3]

[1]JAIST (Tatsunokuchi-cho, Ishikawa)
[2]Toyohasi University of Technology, (Toyohasi)
[3] Dainippon Screen MFG. Co., Ltd.,(Kyoto)

In high-quality image printing it is sometimes required to repair flaws contained in a given image. A simple way for such repair is to paste a flaw region with white and then to move those pixels in the neighborhood by using a tool called an copy-brush. Since it is a very fine operation, it causes great effort to human operators. It is not easy to automate this operation in the existing matrix representation of an image. In our geometric representation of an image as a collection of contour lines for intensity levels this problem is naturally defined as one of reconnecting those contour lines disconnected by a flaw region. An efficient algorithm for reconnecting contour lines is presented based on perfect matching and observations on geometric properties of interconnection paths.

1 Introduction

Recent remarkable development in the printing machine technology is toward more high quality with low costs. In the sense the most important problem is how to automate as many of printing processes based on human operators as possible. In this paper we present an automatic method for one of the tasks, repairing a flaw region in an image, which has been considered to be very hard to be automated. We also demonstrate its effectiveness by experimental results.

In commercial films even a tiny flaw must be removed. Removing electric poles is also the case. An usual way in these cases is to specify a region to be removed as a flaw region and copy pixels from surrounding region by using so-called a "copy-brush" operator. As far as we remain within a framework of matrix representation of images, it is not so easy to automate

this operation. The method we propose is based on a different representation of an image, called "contour representation", which represents an image by a collection of contour lines concerning their intensity levels. This is just like a terrain map which is obtained by regarding an intensity level at each pixel as height at the corresponding location. This representation admits geometric (and thus global) treatment of an image. This is a fundamental idea behind our approach.

Assuming the contour representation of an image, the above-stated flaw repairing problem is naturally defined as the following geometric problem. That is, specified a flaw region by a simple polygon and removed all of contour lines included in it, those contour lines intersecting the boundary of the flaw region become disconnected. If we reconnect those disconnected contour lines as natural as possible within the flaw region, the resulting image is expected to look natural. In this paper, we first describe how to reconnect those disconnected contour lines based on perfect matching and observations on geometric properties of interconnection paths.

The authors do not intend to apply the algorithm in this paper to restore texture images. For those random-like images frequency-based based approach by Hirani and Totsuka [4] would be more appropriate.

2 Contour Representation of an Image

A discrete image is usually represented in a matrix form in which each element represents an intensity level of the corresponding pixel (in three matrices for a color image). Contour representation is a different way of representation of an image as a collection of contour lines with intensity levels as heights. A contour line for a level i is the boundary of a region consisting of pixels whose levels are greater than or equal to i.

Our contour representation is different from what is commonly used in computer vision in the sense that contour lines exist between pixels in our method while they pass through the center of pixels in the common one. More concretely, a contour line consists of horizontal and vertical lattice edges between pixels. This difference is important especially for the applications dealt with in this paper in the definition of regions.

3 Flaw Repairing Problem

3.1 Properties of Contour Lines

A flaw region to be removed is specified as a simple polygon F. Then, each contour line intersecting the boundary of F is disconnected by the removal of F. Thus, the problem is how to reconnect those disconnected contour lines "naturally." Although it is not known what is the best way to reconnect those disconnected contour lines so that the resulting image looks natural. Our experience based on experiments suggests that minimization of the total length of chords to be added to reconnect them leads to reasonable results in many cases. Therefore, we settle our goal in this paper to propose an efficient algorithm for connecting contour lines intersecting a flaw region so that the total length of chords to be added in the region for their interconnection is as small as possible.

Intersections of contour lines with the boundary of a flaw region F, called "terminals", are numbered in a clockwise manner along the boundary as $v_0, v_1, \ldots, v_{n-1}$. Since a contour line is directed so that pixels of higher levels lie to its right, there are two types of terminals depending on the directions of their associated contour lines, that is, in one type they enter the flaw region F and in the other type they exit from F. The type of a terminal v_i defined like this is denoted by $\mathrm{type}(v_i)$. It is defined by $\mathrm{type}(v_i) = \mathrm{IN}$ for those terminals at which contour lines enter F and by $\mathrm{type}(v_i) = \mathrm{OUT}$ otherwise. When the contour line associated with a terminal v_i is the one between levels k and $k+1$, it is denoted by $\mathrm{level}(v_i) = k$ (see Fig. 1).

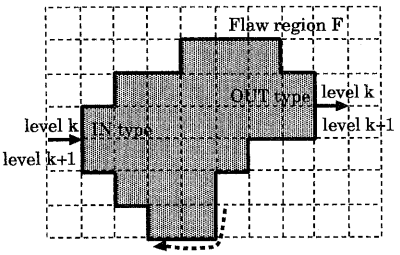


Figure 1: Classification of contour lines incident to a flaw region by their directions.

No two contour lines pass through the same place in a terrain map unless they are from a cliff. Thus, if we assume that no two contour lines touch each other, the terminals change their associated levels continuously along the boundary of a flaw region. Thus, a terminal of level k is adjacent to those of levels of $k-1, k$ or $k+1$. Here note that contour lines are directed so that higher level pixels lie to their right. This means the following constraints: That is, for a terminal v_i of level k with the type IN (OUT, respectively), its clockwise neighbor is of level $k-1$ ($k+1$, respectively) if it is of type IN (OUT, respectively) and k otherwise (if their types are different).

The above assumption does not hold in practice. It is rather common in a real image that more than two contour lines of different levels pass through the same location. To treat such an image just like a terrain map the following convention suffices: When two pixels of levels j and k ($j < k$) are adjacent on the boundary of F, $(k-j)$ contour lines pass through between them and so we create $k-j$ terminals associated with those levels (although their physical locations are the same). More precisely, if they are IN-type terminals, they are arranged clockwisely along the boundary of F in the decreasing order of their levels, and in the increasing order otherwise. No contradiction is caused by this convention.

We assume the ordering of those terminals along the boundary of F. For two terminals v_i and v_j, a set of terminals encountered when we trace the boundary of F clockwisely from v_i to v_j is denoted by $V_F(v_i, v_j)$. For any terminal v_i on the boundary of F there must be another terminal v_j so that v_i and v_j are interconnected by a contour line because any terminal is originally created by a contour line intersecting F. Such a terminal v_j is called a friend of v_i. Formally, it is defined by

$$v_j \text{ is a friend of } v_i \iff \text{level}(v_j) = \text{level}(v_i) \text{ and } \text{type}(v_j) \neq \text{type}(v_i).$$

By the property that intensity levels change continuously along the boundary of F, for any terminal there is a bounded interval along the boundary of F in which its friend exists. We assume $\text{level}(v_i) > \text{level}(v_j)$ below.

1. $\text{type}(v_i) = \text{IN}$ and $\text{type}(v_j) = \text{IN}$:
   a friend of $v_i \in V_F(v_j, v_i)$, a friend of $v_j \in V_F(v_j, v_i)$.
2. $\text{type}(v_i) = \text{OUT}$ and $\text{type}(v_j) = \text{OUT}$:
   a friend of $v_i \in V_F(v_i, v_j)$, a friend of $v_j \in V_F(v_i, v_j)$.
3. $\text{type}(v_i) = \text{IN}$ and $\text{type}(v_j) = \text{OUT}$:
   a friend of $v_i \in V_F(v_j, v_i)$, a friend of $v_j \in V_F(v_i, v_j)$.
4. $\text{type}(v_i) = \text{OUT}$ and $\text{type}(v_j) = \text{IN}$:
   a friend of $v_i \in V_F(v_i, v_j)$, a friend of $v_j \in V_F(v_j, v_i)$.

Now, we turn to the problem of how to repair a flaw region by interconnecting terminals from disconnected contour lines in a "natural" way. Here we have to note the following constraints (see Fig. 2). When we say that two paths cross each other, it means that one lies in the both sides of the flaw region dissected by the other path.

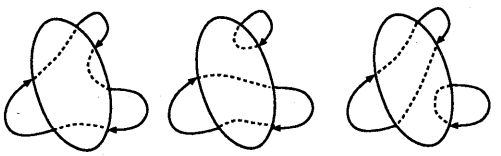
Constraint 0: Every contour line must become a closed loop without self-intersection (allowing touching itself).

Constraint 1: Two terminals to be interconnected must be of the same level.

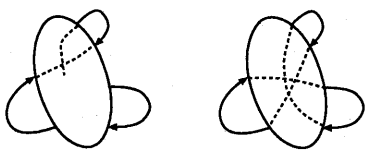
Constraint 2: Two terminals to be interconnected must be of different types.

Constraint 3: Terminals must be interconnected only within the flaw region.

Constraint 4: No two contour lines cross each other.



(a) legal connections



(b) illegal connections

Figure 2: Constraints on interconnection of disconnected contour lines.

3.2 Algorithm for Reconnecting Disconnected Contour Lines

The problem we consider in this paper is described as follows.

Problem 1 *When a region in an image represented by the contour representation is specified as a flaw region by a simple polygon, reconnect those contour lines disconnected by the flaw region so that all of the above-stated constraints are satisfied. In addition, the total length of paths to be added for the reconnection within the flaw region should be minimized.*

The previous solution to the problem by the authors was a greedy algorithm in which disconnected contour lines are interconnected one by one in the decreasing order of their associated area within the flaw region. More precisely, after ordering contour lines, based on the dynamic programming they found shortest interconnection paths under the constraints that they do not obstruct connectivity of remaining contour lines. So, it is a level-by-level optimization algorithm and it is questionable whether it obtains globally optimal interconnections.

The algorithm to be proposed in this paper tries to find globally optimal solution to the problem. The basic idea of the algorithm is matching.

First define a graph $G = (V, E)$ as follows: Terminals on the boundary of a flaw region F are *vertices* of the graph and when two terminals v_i and v_j satisfy the constraints 1 and 2, that is, their types are different from each other but their levels are just the same, an *edge* is drawn between corresponding vertices. The *weight* of an edge is a combination of the two *keys*.

The *first key* of an edge is the length of a shortest path interconnecting the two corresponding terminals within the flaw region, and is denoted by d_{ij}. Note that the distance is measured by the L_2 metric. The *second key* is the square root of their index difference (modulo n, the number of terminals). Although two different index differences can be considered for two terminals v_i and v_j, i.e., forward (or clockwise) and backward (or counter-clockwise) differences, we take their minimum as their index difference. Formally, the index difference ξ_{ij} between v_i and v_j is defined by $\xi_{ij} = \min\{(i-j+n) \bmod n, (j-i+n) \bmod n\}$.
By using these two keys, we define a weight of an edge (v_i, v_j) by $(d_{ij}, \sqrt{\xi_{ij}})$.

When we compare the weights of two edges, we first compare their first keys and then the second keys if the first keys are the same. The reason will become clear in the following discussions why the first keys are not sufficient.

The graph defined above is bipartite and it is obvious that it has perfect matching since all the terminals can be interconnected by their original contour lines. It is also evident that any matching satisfies all of the above constraints 0 through 3. Thus, the problem of how to prevent crossing between different contour lines and to minimize the total length of contour lines at the same time is left. Here, we show that a perfect matching with minimum weight is a solution to this optimization problem.

Theorem 1 *Interconnecting terminals according to a perfect matching with minimum weight, no two contour lines cross each other.*

Proof: Let v_a, v_b, v_c, and v_d be terminals on the boundary of a flaw region, and assume that a path interconnecting v_a and v_b and that between v_c and v_d cross each other. Then, we want to show that both of the edges (v_a, v_b) and (v_c, v_d) cannot be included in a perfect matching with minimum weight. For this purpose, for any perfect matching M including both of the edges there exists a perfect matching with less weight.

By the definition, we have $\text{type}(v_a) \neq \text{type}(v_b)$, $\text{type}(v_c) \neq \text{type}(v_d)$, and $\text{level}(v_a) = \text{level}(v_b)$, $\text{level}(v_c) = \text{level}(v_d)$.

Before continuing the proof of the theorem, we need some basic observations.

Observation 2 *A shortest path interconnecting two terminals on the boundary of a flaw region F does not intersect itself.*

The observation follows from the fact that any path with self-intersection can be shortcut.

Observation 3 *A shortest path between two terminals on the boundary of a flaw region does not cross another shortest path between another pairs of terminals more than once.*

If they cross each other more than once, a closed region is formed between the first and second intersections p and q. Then, both of the clockwise and counter-clockwise paths between p and q along the boundary of the closed region are shortest paths. It contradicts to the uniqueness of a shortest path in a simple polygon.

Case 1: The four terminals are all of the same level:

By the definition, the shortest path between v_a and v_b crosses the shortest path between v_c and v_d. Let the intersection be p. Then, the arrangement of the four terminals must be either (v_a, v_c, v_b, v_d) or (v_a, v_d, v_b, v_c) clockwisely along the boundary of F. Otherwise, the two shortest paths must cross even number of times, which is impossible.

First consider the case of the arrangement (v_a, v_c, v_b, v_d). Then, the above shortest paths can be decomposed into four parts: the path $\mathrm{P}(v_a, p)$ from v_a to p, the path $\mathrm{P}(p, v_b)$ from p to v_b, the path $\mathrm{P}(v_c, p)$ from v_c to p, and the path $\mathrm{P}(p, v_d)$ from p to v_d. Changing their combinations to $\mathrm{P}(v_a, p) + \mathrm{P}(p, v_d)$ and $\mathrm{P}(v_c, p) + \mathrm{P}(p, v_b)$, the resulting interconnection pattern has different pairs of terminals to be interconnected without increasing the total length of paths. If v_a is of a different type from that of v_c, we should change the combination to $\mathrm{P}(v_a, p) + \mathrm{P}(p, v_c)$ and $\mathrm{P}(p, v_d) + \mathrm{P}(p, v_b)$.

Since $\mathrm{P}(p, v_a) + \mathrm{P}(p, v_d)$ is a path from v_a to v_d, its length is no shorter than a shortest path between them, that is, it is no shorter than d_{ad}. In other words, we have

$$d_{ab} + d_{cd} = |\mathrm{P}(v_a, p)| + |\mathrm{P}(p, v_b)| + |\mathrm{P}(v_c, p)| + |\mathrm{P}(p, v_d)| \geq d_{ad} + d_{cb}$$

if $\mathrm{type}(v_a) = \mathrm{type}(v_c)$, and

$$d_{ab} + d_{cd} \geq d_{ac} + d_{bd}$$

if $\mathrm{type}(v_a) \neq \mathrm{type}(v_c)$.

It is possible that equalities hold in the above inequalities. An example is shown in Fig. 3.

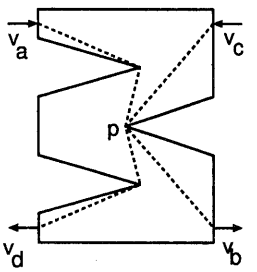


Figure 3: The case in which the sum of the shortest path lengths cannot be decreased.

On the other hand, taking the arrangement of terminals into accounts, we have the following equalities concerning the index difference. When the index difference between v_a and v_b is given by their backward difference and that between v_c and v_d is given by their forward difference, we have

$\xi_{ab} = \xi_{ad} + \xi_{db}$ and $\xi_{cd} = \xi_{cb} + \xi_{bd}$.

Thus, based on a simple observation that when $0 < p < q < r < s$ and $p + s \leq q + r$, we have $\sqrt{p} + \sqrt{s} < \sqrt{q} + \sqrt{r}$, the following inequalities follow.

$\sqrt{\xi_{ab}} + \sqrt{\xi_{cd}} > \sqrt{\xi_{ad}} + \sqrt{\xi_{bc}}$, and

$\sqrt{\xi_{ab}} + \sqrt{\xi_{cd}} = \sqrt{\xi_{ad} + \xi_{bd}} + \sqrt{\xi_{bc} + \xi_{bd}} > \sqrt{\xi_{ad} + \xi_{bd} + \xi_{bc}} + \sqrt{\xi_{bc}} \geq \sqrt{\xi_{ac}} + \sqrt{\xi_{bd}}$.

The proof for the case in which the index difference between v_a and v_b is given by their backward difference and that between v_c and v_d is given by the forward one is similar to that for the above case.

Difficulty arises when both of the indices are given in the same direction. For example, let us suppose that both of them are given by the backward indices. In this case, we have

$\xi_{ab} = (a - b + n) \bmod n$, $\xi_{cd} = (c - d + n) \bmod n$

and thus we have

$\xi_{ab} = \xi_{ad} + \xi_{db}$ and $\xi_{cd} = \xi_{ca} + \xi_{ad}$.

Therefore, if the type of v_a is different from that of v_c, interconnecting v_a with v_c and v_b with v_d results in smaller sum of their index differences and the sum of the square roots of the index differences also decreases.

On the other hand, if v_a and v_c are of the same type, it is not so obvious. In this case we are going to interconnect v_a with v_d and v_b with v_c. Then, concerning their index differences, we have the equality

$\xi_{ab} + \xi_{cd} \leq \xi_{ad} + \xi_{bc}$.

However, the equality implies

$\xi_{ad} < \min\{\xi_{ab}, \xi_{cd}\}$.

By the simple observation above we have

$\sqrt{\xi_{ab}} + \sqrt{\xi_{cd}} > \sqrt{\xi_{ad}} + \sqrt{\xi_{bc}}$.

Based on the observations above, it is seen that the combination (v_a, v_d) and (v_b, v_c) has smaller weight in the case of $\mathrm{type}(v_a) = \mathrm{type}(v_c)$, and that of (v_a, v_c) and (v_b, v_d) has smaller weight than that for (v_a, v_b) and (v_c, v_d) if $\mathrm{type}(v_a) \neq \mathrm{type}(v_c)$.

If the terminals are arranged in the order v_a, v_c, v_b, and v_d, we have similar discussion by reversing the directions of index differences.

The proof proceeds similarly for the remaining cases.

□

Based on the theorem, we can find an optimal interconnection pattern by the following algorithm. First, for each intensity level we enumerate all the crossings between contour lines of that level and the boundary of a flaw region, and after classifying those crossings into IN-type and OUT-type, for each pair of crossings of different types we calculate the length of a shortest path interconnecting them within the flaw region and their index difference. These informations are summarized as a bipartite graph. Since a flaw region is specified as a simple polygon, if we partition its inside into triangles, for each terminal we can compute the shortest path tree from the terminal (and the geodesic distances from the terminal to other terminals) in time proportional to the number of vertices of the polygon [3]. Thus, if we have n crossings and the flaw region is specified as an m-gon, we can build such a bipartite graph in time $O(n^2 + mn)$. Then, we find a minimum-weight matching for this bipartite graph. Some known algorithms are available for this purpose [2]. The best known algorithm runs in $O(n^{2.5})$ time, which leads to the following theorem:

Theorem 4 *Let n be the number of contour lines which cross a flaw region. Then, we can find an optimal interconnection pattern for contour lines disconnected by a flaw region in time $O(n^{2.5} + nm)$, where m is the number of vertices of a polygon which forms a flaw region.*

4 Experimental Results

We have implemented our algorithm against several image data. Examples we have tested are to remove electric poles from a street, to remove claw's feet around eyes, and removing characters in a wine bottle without eliminating highlight part, and so on. The results were satisfactory in all those cases. One of the examples is shown in Fig. 4 in which the flaw region depicted by a white region is restored in the resulting image below. A similar result could be achieved by manual operations using so-called "copy-brush" which copy pixels in the surrounding region. A manual operation, however, takes time and cost, and the quality of the resulting picture depends heavily on experience of human operators. This is why automatic processing is required.

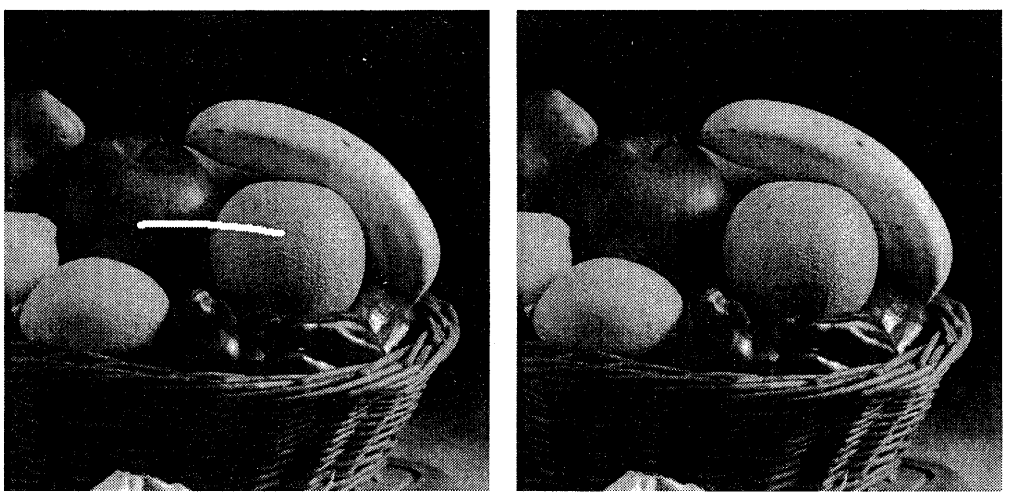


Figure 4: An experimental result. A picture with a flaw region above and the restored picture image below.

5 Conclusions

In this paper we have presented an algorithm for removing flaws in a picture and described experimental results. The key idea to the success is the use of perfect matching for global optimum. One important contribution of this paper is that the new algorithm can guarantee global optimal solutions in less time than the previous one. That is, our previous algorithm found an optimal solution for each level, but it does not guarantee an optimal solution for all levels.

The contour representation of an image as a collection of contour lines is a new representation proposed by the authors. It allows geometric treatment of an image, which is especially important for getting global view. This is another key to the success. One drawback of the representation is that it takes much storage. In fact, the total length of the contour lines may be much larger than the total number of pixels (or image size). In our application, however, we do not need to convert the whole image into its corresponding contour line representation. We are just required to have contour representation for the region specified as a flaw region. Thus, the overhead due to the representation is not so much. Conversion from contour representation to the conventional matrix form is straightforward.

Acknowledgment

The authors would like to their thanks to Kazuhiro Nakai for his valuable supports in our implementation.

References

[1] T. Asano and S. Kimura: "Contour Representation of an Image with Applications, " Proc. SPIE's International Symposium on Vision Geometry IV, pp.14-22, San Diego, July 1995.

[2] T.H. Cormen, C.E. Leiserson, and R.L. Rivest: "Introduction to Algorithms," The MIT Press, 1990.

[3] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan: "Linear time algorithm for visibility and shortest path problems inside simple polygons," in Proc. 2nd ACM Symp. on Computational Geometry, pp.1-13, 1986.

[4] A. N. Hirani and T. Totsuka: "Combining Frequency and Spatial Domain Information for Fast Interactive Image Noise Removal," Proc. SIGGRAPH'96, pp.269-276, 1996.