Nonparametric Curve Extraction Based on Ant Colony System

Qing Tan[1, 2] **Qing He**[1] **Zhongzhi Shi**[1]

[1]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, China
[2]Graduate University of Chinese Academy of Sciences, China
{tanq, heq, shizz}@ics.ict.ac.cn

Abstract

Curve extraction is an important and basic technique in image processing and computer vision. Due to the complexity of the images and the limitation of segmentation algorithms, there are always a large number of noisy pixels in the segmented binary images. In this paper, we present an approach based on ant colony system (ACS) to detect nonparametric curves from a binary image containing discontinuous curves and noisy points. Compared with the well-known Hough transform (HT) method, the ACS-based curve extraction approach can deal with both regular and irregular curves without knowing their shapes in advance. The proposed approach has many characteristics such as faster convergence, implicit parallelism and strong ability to deal with highly-noised images. Moreover, our approach can extract multiple curves from an image, which is impossible for the previous genetic algorithm based approach. Experimental results show that the proposed ACS-based approach is effective and efficient.

1. Introduction

Curve extraction and recognition is basic and crucial in many fields such as image processing, image analysis and computer vision. In curve recognition, image segmentation algorithm is firstly used before extraction. Due to the complexity of raw images and the limitation of segmentation algorithm, binary images after segmentation always contain discontinuous curves and noisy pixels, which bring difficulty for curve recognition.

The well-known Hough transform (HT) is the most frequently used approach for curve extraction in a binary image which contains noisy pixels. In HT (Hough 1962), a linear point-to-line mapping is used to detect the largest sets of collinear points. A decade later, (Duda and Hart 1972) modified Hough's method to fit the situation that the intersection of lines could lie in ∞ in the coordinate. Since the emergence of HT, many scholars (Sklansky 1978, Brown 1983) proposed improved algorithms to detect lines and circular arcs in a binary image and there is still a wealth of literatures about the Hough transform in recent years (Chandan et al. 2008, Fernandes et al. 2008). However, all these HTs have the limitation that they can only detect parametric curves. But in actual applications such as pavement crack detection, river course detection in remote sensing images and other crack detection, curve equations are usually not known a priori before recognition. HT methods do not work in these situations.

Extracting curves from a binary image means to select a subset of pixels (real target pixels) from all nonzero pixels. So every pixel sequence can denote a curve and be treated as a feasible solution. In this way, curve extraction is converted to a combinatorial optimization problem and the curve could place no emphasis on whether it is a parametric one or not. From this point of view, some curve extraction approaches based on genetic algorithm (GA) were proposed. (Wei et al. 2005) proposed a niche genetic algorithm (NGA) based nonparametric curve extraction approach. (Saitoh 1998) extracted disconnected curve using genetic algorithm based on factors of closeness and continuity in perceptive grouping functions. These GA-based methods are effective in extracting the main part of the dominant curve. However, experiments indicate that their accuracy and computation efficiency are not high. Moreover, these methods can only detect the most dominant curve in the binary image. Considering the multiple curves, it needs further creative modification to fit them.

Ant colony optimization (ACO) (Dorigo et al. 1996, Dorigo and Stützle 2004) metaheuristic is a population-based approach inspired by the behavior of ant colony in real world. In ACO, solutions of the problem are constructed within a stochastic iterative process, by adding solution components to partial solutions. ACO algorithms have been successfully applied to several NP-hard combinatorial optimization problems such as traveling salesman problem (TSP).

This work is supported by the NSFC (No. 60933004, 60903141, 60975039), 863 NHTP (No.2007AA01Z132), NRPP (No.2007CB311004) and NSTSP (No.2006BAC08B06).

As one of the best implementation of ACO for TSP, ant colony system (ACS) (Dorigo and Gambardella 1997) has been shown to be competitive against optimization methods like GA and simulated annealing algorithm. In this paper, inspired by solving TSP utilizing ACS, we treat curve extraction as a problem of constructing path consisting of unfixed-length nonzero pixel sequence and propose an ACS-based curve extraction algorithm. Using state transition rule in ACS, the sequences are constructed step by step. Pheromone updating rule keeps the pheromone value in an ideal level to give ants more opportunities to find a better solution. Besides the traditional operators, searching termination function and path reversal rule are specifically designed in order to fit the curve extraction task. All these units work together to obtain the dominant curve. On this basis, the algorithm is extended to extract multiple target curves. We define the similarity between two curves based on the intersection of pixel sequences. All the high fitness curves whose similarity with others is smaller than a given threshold could be extracted.

The rest of this paper is organized as follows: Section 2 introduces the fitness function adopted in this paper. Then in Section 3, we present our ACS-based curve extraction algorithm and give detailed analysis on the related operators. Experimental results are given in Section 4, followed by our conclusions in Section 5.

2. Design of Fitness Function

In order to extract the most probable curve as human visualization does, the fitness function is designed according to human visual characteristics which could be summarized as follows.

(1) Along the most dominating curve, the density of nonzero pixels must be the highest one of all the envisaged curves in the image. In another word, the total length of the break points in the curve should be as small as possible.

(2) For a nonzero pixel sequence, the longer the span between initial pixel and terminal pixel, the more the probability of denoting the dominant curve.

Based on the above principles, some variables and fitness function are briefly defined as follows and more detailed information could be obtained in literature (Wei et al. 2005).

In a pixel sequence (*Curve*), the distance between the k_1th element and the k_2th element is

$$Dist(Curve(k_1), Curve(k_2)) = \sqrt{(x_{Curve(k_1)} - x_{Curve(k_2)})^2 + (y_{Curve(k_1)} - y_{Curve(k_2)})^2}, \quad (1)$$

where ($x_{Curve(k_1)}$, $y_{Curve(k_1)}$) and ($x_{Curve(k_2)}$, $y_{Curve(k_2)}$) are respectively the positions (row, column) of $Curve(k_1)$ and $Curve(k_2)$ in the image.

Definition 1: *Spn*— the span between initial nonzero pixel and terminal nonzero pixel of the *Curve* in the binary image, which could be calculated as follows:

$$Spn(Curve) = Dist(Curve(1), Curve(L_{Curve})). \quad (2)$$

Definition 2: *Lcv*— the actual length of the curve.

$$Lcv(Curve) = \sum_{k=1}^{L_{Curve}-1} Dist(Curve(k), Curve(k+1)), \quad (3)$$

where $Curve(k)$ and $Curve(k+1)$ are two neighbour pixels and L_{Curve} is the pixel number of the *Curve*.

The definition of distance is based on the premise that each pixel is a square with side length 1. The actual length of the curve *Lcv* could be partitioned into two parts: total length cross nonzero pixels and total length cross zero pixels. We use *Wnnc* to represent the total length cross nonzero pixels.

Definition 3: *Wnnc*— the weighted number of nonzero pixels in the curve. Suppose that each pixel is a square with side length 1, and the weighted '1' is the actual length of the line cross the square. The summation of the weighted '1' on the curve is called *Wnnc*.

$$Wnnc = \sum_{k=1}^{L_{Curve}} \frac{1}{\Theta(k)}, \qquad \Theta(k) = \begin{cases} \cos\theta(k) & \text{if } \theta(\mathrm{k}) \le \frac{\pi}{4} \\ \sin\theta(k) & \text{if } \theta(\mathrm{k}) > \frac{\pi}{4} \end{cases} \quad (4)$$

where $\theta(k)$ is the angle between x coordinate and the line segment of $Curve(k)$ and $Curve(k+1)$.

Definition 4: *Ftnf*— the fitness function of the curve.

$$Ftnf = \frac{Spn}{10 + Lcv - Wnnc}. \quad (5)$$

In the fitness function, *Lcv-Wnnc* is the total length cross zero pixels. The empirical value 10 is used for adjusting the acutance of *Lcv-Wnnc*. The function designed in this way is easy to compute and considers both two principles of human visual characteristics.

3. ACS-based Curve Extraction

In our proposed ACS-based Curve Extraction (ACE) algorithm, each nonzero pixel is analogous to a city in TSP, the target of our algorithm is to find a pixel sequence which owns the highest fitness and denotes the most dominant curve. Follow the pattern of solving TSP utilizing ACS, we randomly choose a nonzero pixel as the initial pixel of the sequence and then apply a state transition rule to add other nonzero pixels one by one to form a pixel sequence. The main difference between solving curve extraction task and TSP is that we do not visit all the nonzero pixels in the image but dynamically judge whether to stop adding a next pixel into the sequence.

3.1 ACE Algorithm

The main structure of ACE algorithm is similar with that of ACS algorithm when solving TSP. The procedure is described in Algorithm 1.

In each iteration, *m* ants are initially positioned on the

pixels. Each ant builds a pixel sequence (i.e., a feasible solution to the curve extraction task) by repeatedly applying the state transition rule. While constructing its solution, an ant also modifies the amount of pheromone on the visited edges by applying the local updating rule. Once all ants have terminated their solutions, the amount of pheromone on edges is modified again by applying the global updating rule. Different from ACS when solving TSP, a searching termination function is used in step 4 when the ant moves to a new pixel. It decides whether the ant will continue adding pixels into its current pixel sequence. Also a path reversal rule is used to reverse the current solution when the ant is stopped by searching termination function the first time. Detailed explanation will be given in the following subsections.

Algorithm 1: ACS-based curve extraction (ACE)

1. Initialize parameters and pheromone;
   Loop /* at this level each loop is called an iteration */
2. Put each ant in a random nonzero pixel as starting pixel;
   Loop
3. Each ant applies a *state transition rule* to choose a next pixel to visit, followed by a *local pheromone updating rule*;
4. Each ant calls the *searching termination function*;
5. If the return result is continue moving,
   Go to step 3 to incrementally build a solution;
6. Else /* the result is stop moving */
   If it is the first time to stop moving
   Apply *path reversal rule* and then go to step 3;
   Else /* it is the second time to stop moving */
   Stop moving for that ant;
   /* that ant has built a complete solution */
   Until all ants have built a complete solution
7. Calculate fitness of each solution and a *global pheromone updating rule* is applied to reinforce pheromone on the edges belonging to the best solution;

Until end condition of the algorithm is satisfied, usually reach a given iteration number.

3.2 State Transition and Pheromone Updating Rule

3.2.1 State Transition Rule. In ACE, the state transition rule is as follows: ant positioned on pixel i chooses the pixel j to move to by applying the rule given by Eq. (6).

$$j = \begin{cases} \arg\max_{l \in U}[(\tau_{il})^{\alpha}(\eta_{il})^{\beta}], q \le q_0 \\ V \qquad\qquad\qquad\quad ,q > q_0 \end{cases}, \tag{6}$$

V is a random variable selected according to the probability distribution given by Eq. (7).

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l \in U}(\tau_{il})^{\alpha}(\eta_{il})^{\beta}}, \tag{7}$$

where U is the set of pixels that not be visited by ant yet, q is a random number uniformly distributed in [0…1], and $0 \le q_0 \le 1$ is a parameter in ACS. Besides, τ_{ij} is the pheromone on edge (i, j), η_{ij} is the heuristic information provided by the optimization problem itself, α and β are parameters which determine the relative importance of pheromone versus heuristic information.

The crucial part in state transition rule is how to define the heuristic information η. Usually, heuristic information stands for the cost of adding new elements into the current slice of the solution. For example, heuristic information defined in TSP is reciprocal of the distance between two cities. In curve extraction problem, the accompanying *Lgap* value when adding a new pixel into the path is just the cost. Heuristic information is defined in Eq. (8).

$$\eta_{ij} = 1/(Lgap_{ij} + 1), \tag{8}$$

where $Lgap_{ij}$ is the length of the gap between current pixel i and pixel j. Ants prefer to move to pixels with high pheromone level and small *Lgap* value on the edges.

3.2.2 Pheromone Updating Rule. Pheromone τ_{ij} reflects the desirability of visiting pixel j from current pixel i. In ACS, pheromone updating includes two parts: local updating rule and global updating rule. Local updating rule is applied while ants construct solutions. When building a solution, ants visit edges and change their pheromone level by applying the local updating rule of Eq. (9).

$$\tau_{ij}^{new} = (1-\varepsilon)\tau_{ij}^{old} + \varepsilon\tau_0, \tag{9}$$

where $0 \le \varepsilon \le 1$ is a pheromone decay parameter and τ_0 is the initial pheromone level.

Global updating rule is applied when all ants have terminated their pixel sequences. The pheromone level is updated by applying the global updating rule of Eq. (10).

$$\tau_{ij}^{new} = (1-\rho)\tau_{ij}^{old} + \rho\Delta\tau_{ij}^{bs}, \quad \forall (i,j) \in T^{bs}, \tag{10}$$

$$\Delta\tau_{ij}^{bs} = \begin{cases} best\_fitness & \text{if } (i,j) \in \text{ global-best-path} \\ 0 & \text{otherwise} \end{cases}, \tag{11}$$

where $0 \le \rho \le 1$ is a decay parameter, and *best_fitness* is the fitness of the global best path from the beginning of the trial. It indicates that only those edges belonging to the globally best path will receive reinforcement.

3.3 Searching Termination Function and Path Reversal Rule

3.3.1 Searching Termination Function. In TSP, length of the solutions is a fixed number because all the cities must be visited once and only once. As to the problem of curve extraction, the curve is composed of only a subset of all the nonzero pixels. In order to fit for this feature, a searching

termination function is designed in our approach. In the process of constructing a pixel sequence, termination function will be applied whenever the ant moves to a new pixel. It judges the ant whether stop moving or continue adding a new pixel to the current partial sequence. In an ideal state, ants should stop when they move to the end points of the curve, otherwise should be allowed to continue moving to a new pixel. When all the unvisited nonzero pixels are far away from the current pixel, a great increment of weighted number of zero pixels (length cross zero pixels) will be added into the curve if ant continues moving to a new pixel, which will decrease the density of nonzero pixels and cause low fitness of the solution. When this situation happens, the ant should have more probability to stop at the current pixel.

Definition 5: *Lgap*— length of the gap between two nonzero pixels. Connecting two pixels refers to connect their centroids. *Lgap* equals the distance of the centroids minus the length cross the two pixels. It could be calculated when the positions (row and column in the image) of the pixels are given.

The searching termination function designed in our approach is given by Eq. (12), which gives the probability of stopping at the current pixel.

$$Stop() = \begin{cases} \sqrt{\dfrac{min\_Lgap}{ave\_Lgap}} & \text{if } min\_Lgap \le ave\_Lgap \\ 1 & \text{if } min\_Lgap > ave\_Lgap \end{cases}, \quad (12)$$

min_Lgap is the minimal *Lgap* value among those from current pixel to all the unvisited nonzero pixels. The more the *min_Lgap*, the more stopping probability will be returned. *ave_Lgap* is the average of all *Lgap* values between any two nonzero pixels in the image, which is a constant for a given image.

3.3.2 Path Reversal Rule. Considering that it is hard or even impossible to determine which pixel is the end point of the curve, each ant starts from a random nonzero pixel. Suppose the target curve is pixel sequence [1 2 3 4 5 6 7 8 9] and an ant starts from pixel number 4, constructs a path [4 5 6 7 8 9] and stops at pixel number 9 judged by the searching termination function mentioned above. Obviously, it is just part of the target curve. On the other hand, every pixel in the curve has two adjacent pixels except for the end points. From a random starting pixel, ant constructs a path in only one direction. In order to allow the ant to continue moving towards another direction after stopping searching in the first direction, path reversal rule is designed as follows: exchange the position of starting pixel and terminal pixel in the path, reverse the path between them at the same time. After applying this rule, the raw path [4 5 6 7 8 9] turns into [9 8 7 6 5 4]. In this way pixel number 9 becomes one of the two end points and pixel number 4 becomes a new starting node, from which ant could continue searching in another direction until stopped by searching termination function the second time. And it is likely that ant constructs a path [9 8 7 6 5 4 3 2 1] from [9 8 7 6 5 4].

3.4 Multi-curve Extraction

The most dominant curve in the binary image could be extracted by ACE algorithm. However, many images may contain two or even more curves in real applications. In this section, we develop our ACE algorithm to fit this situation.

ACO algorithms always focus on finding the global best solution in most cases. But for the problem of multi-curve extraction, different target curves correspond to different local best solutions.

Suppose two curves are represented as pixel sequences in Eq. (13) and Eq. (14). $Curve_a$ is the current global best solution obtained by the ACE algorithm.

$$Curve_a = [a_1 \quad a_2 \quad a_3 \quad \cdots \quad a_m] \quad (13)$$

$$Curve_b = [b_1 \quad b_2 \quad b_3 \quad \cdots \quad b_n]. \quad (14)$$

Each sequence constitutes a pixel set. When the intersection of the two pixel sets contains lots of pixels, $Curve_a$ and $Curve_b$ are considered to represent a same target curve. $Curve_b$ should be neglected due to its lower fitness. But in another situation, $Curve_a$ and $Curve_b$ indicate two quite different curves when their intersection is empty or only has a few pixels. Although $Curve_a$ is the most dominant curve, $Curve_b$ should be considered as another target curve if its fitness is higher than a given threshold.

Genetic algorithm is a population evolutionary method, in which the individuals with lower fitness will be eliminated gradually. Due to such a mechanism, $Curve_b$ is hard to be reserved by the GA-based curve extraction methods. Fortunately, it is not a problem in ACS-based algorithm. Ants randomly start from pixel b_i have a considerable probability to construct the target $Curve_b$ using the rules designed in ACE algorithm.

In order to judge whether two pixel sequences represent the same target curve, we define the similarity of two curves based on the size of their intersection.

Definition 6: *Similarity*— the similarity function of two curves.

$$Similarity(Curve_a, Curve_b) = \frac{|Curve_a \cap Curve_b|}{\min(|Curve_a|, |Curve_b|)}, \quad (15)$$

where '$|\ |$' denotes the size of the set. In the extreme case, according to the above definition, *similarity* would be 0 when the intersection is empty and it would be 1 when one set is a subset of another set.

For multi-curve extraction, we use *target_curves* to store all the target curves. In each iteration, all the constructed paths satisfy Eq. (16) will be selected as candidate solutions. For each such candidate solution $Curve_c$, calculate its similarity with all the target curves currently stored in *target_curves* according to Eq. (15). $Curve_c$ will be added into *target_curves* when Eq. (17) is satisfied for every

$Curve_t$ stored in *target_curves*. Otherwise, pick out $Curve_t$ whose similarity with $Curve_c$ is not satisfied. Use $Curve_c$ to substitute $Curve_t$ in *target_curves* if its fitness is higher than that of $Curve_t$. This situation means $Curve_c$ and $Curve_t$ represent a same target curve and $Curve_c$ further optimizes the former $Curve_t$.

$$Ftnf(Curve) \geq threshold\_fitness, \quad (16)$$

$$Similarity(Curve_c, Curve_t) \leq threshold\_similarity. \quad (17)$$

where *threshold_fitness* and *threshold_similarity* are two threshold parameters given in advance.

Besides, the global pheromone updating rule is modified in order to fit multi-curve extraction. Instead of only the globally best path will receive reinforcement, all paths stored in *target_curves* are allowed to release pheromone.

4. Experimental Results

In order to test the effectiveness of the proposed ACE algorithm, we pick 40 raw pavement images taken from camera together with 80 generated binary images as our testing samples. Each raw pavement image has 1024×1280 pixels. Binary tile images with 64×80 pixels each could be obtained using image preprocessing and segmentation algorithms adopted in (Wei et al. 2005). The generated images contain various curves of undefined shapes and each image has 80×80 pixels. The ACS parameters setting in our experiments is a typical combination as follows, whose detailed analysis could be obtained in (Dorigo and Gambardella 1997).

$\alpha = 1$, $\beta = 2$, $q_0 = 0.9$, $\varepsilon = 0.1$, $\rho = 0.1$, $m = 10$.

In the following images, white pixels are not identified, gray ones are recognized and black is background. The extraction results illustrated in figure 1 show the effectiveness of ACE algorithm.

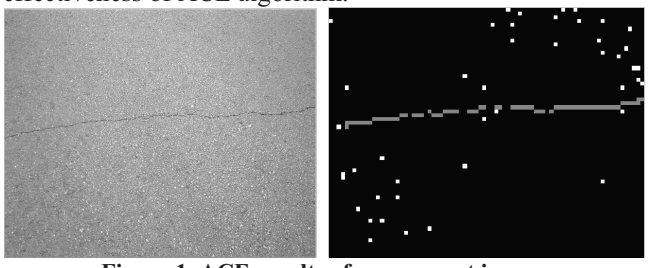
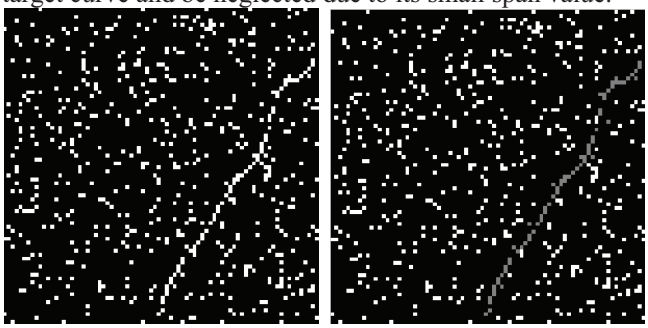


Figure 1. ACE results of a pavement image.

For some extreme highly-noised images, ACE algorithm also presents high performance. In the sample image illustrated in figure 2(a), 8% of pixels of backgrounds are random noisy pixels, which is much larger than the corresponding number 0.3% in the experimental images used in (Saitoh 1998). Although the noise proportion is so large that nearly reaches the limitation of human visual recognition, ACE algorithm can still extract the dominant curve in the image effectively.

For each image in the image samples, we run ACE algorithm three times considering its randomicity. The average extraction accuracy for each image is recorded to measure ACE's performance. Extraction accuracy (EA) refers to the proportion of correctly extracted pixel number to the total number of nonzero pixel. Table 1 summarizes ACE's performances on the image samples. 42 of 120 images get a perfect extraction result (EA=100%) and 118 of 120 images have an EA higher than 95%. The average EA for the 120 images is 98.8%. The reason for the failure sample is that the curve is separated into two parts by a big gap. The short part of the curve is not treated as another target curve and be neglected due to its small span value.



(a) Raw highly-noised image (b) ACE curve extraction result

Figure 2. ACE result of highly-noised image.

Table 1. ACE's performance on the image samples

EA=100%	EA>98%	EA>95%	Average EA
42/120=35%	99/120=83%	118/120=98%	98.8%

We compare the performance of ACE algorithm with previous NGA-based approach proposed in (Wei et al. 2005). Both methods are implemented in the same running environment. Representative binary images with different noise level are selected to make the comparison. Sample description and extraction results are given in table 2. We conduct *t* –test with 95% confidence to check whether the superiority of ACE method over NGA method is statistically significant. The results indicate that the proposed ACE algorithm extremely significantly outperforms the NGA-based approach in both views of extraction accuracy and computation time.

Generally speaking, the accuracy gap between NGA and ACE is not too big when the noise proportion is relatively low. And this gap becomes obvious as the noise proportion grows. That is because the ACE's state transition rule guarantees the ants focus on the nearby pixels when choosing a next pixel to visit, which is quite insensitive to the noisy pixels out of the curve. However, for every bit of NGA's chromosome, noisy pixels have the same chance to compete with the target pixel on the curve. Although in theory, the extraction accuracy of NGA-based approach may be enhanced by increasing the iteration number, it will cost much more time. On the other hand, the comparison of running time between ACE and NGA presented in table 2 shows that ACE algorithm owns much better computation efficiency than NGA-based approach.

Table 2. Comparison of extraction results between ACE and NGA

Sample Description			Iteration Number		EA (%)		Running Time (s)	
ID	Noise Proportion	Nonzero Pixel	NGA	ACE	NGA	ACE	NGA	ACE
1	0.5%	90	500	20	95.6	100	36.7	6.8
2	0.6%	99	500	20	100	100	36.8	7.0
3	0.9%	117	1000	30	95.7	99.1	78.4	8.8
4	1.0%	129	1000	30	96.9	100	79.8	11.2
5	1.4%	142	1500	40	94.4	99.3	124.1	14.4
6	1.8%	161	1500	40	91.9	100	128.7	19.8
7	2.0%	194	2000	50	89.2	96.9	185.9	30.7
8	2.7%	233	3000	60	90.1	98.3	269.3	35.1
9	3.2%	275	4000	80	88.0	99.3	399.7	62.2
10	4.0%	330	5000	100	84.2	99.1	516.5	99.3

Moreover, images containing multiple curves are used to test the effectiveness of ACE algorithm in multi-curve extraction tasks. Parameter *threshold_fitness* is used to discriminate whether a pixel sequence denotes a target curve. It is insensitive because the fitness of a target curve and a noisy sequence usually has a large gap. Consider that almost all target curves in our experiments own a fitness higher than 2.0, we set *threshold_fitness=2.0*. Besides, we set *threshold_similarity=0.7* and keep other parameters setting the same with those in the former experiments. Extraction results of some representative samples are presented in figure 3, which demonstrate that the proposed algorithm is effective in multi-curve extraction.

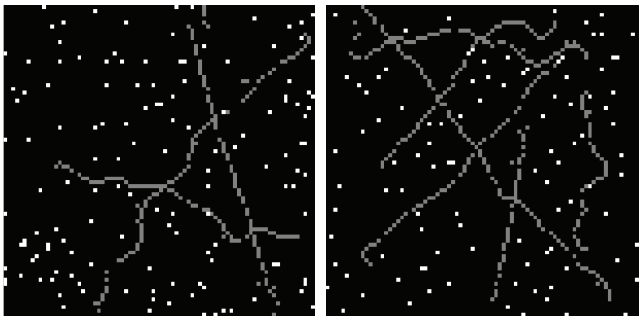


Figure 3. Extraction results of multi-curve images.

5. Conclusion

This work studies the problem of nonparametric curve extraction, which focuses on dealing with the curves whose shape and functions are not given in advance. We propose an ACS-based curve extraction algorithm in this paper and the curves extracted in this way could place no emphasis on parametric ones or not, while the Hough Transform method can only detect parametric curves. Instead of coding the curves in GA-based approaches, solutions are flexibly constructed step by step in our method. Besides the traditional operators in ACS, searching termination function and path reversal rule are specially designed to fit curve extraction task. They are in effect in this research and the proposed ACS-based method is much more effective and efficient than previous GA-based method. The solution construction mechanism in the method is insensitive to noisy pixels and thus guarantees the method good performance even if the processed image is extreme highly-noised. What is more, the method could preserve multiple meaningful locally best solutions, which fits the multi-curve extraction tasks.

6. References

Brown, C.M. 1983. Inherent bias and noise in the Hough Transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5: 493-505.

Chandan, S. et al. 2008. Hough transform based fast skew detection and accurate skew correction methods. *Pattern Recognition*, vol. 41: 3528-3546.

Dorigo, M., and Gambardella, L.M. 1997. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, vol. 1: 53-66.

Dorigo, M.; Maniezzo, V.; and Colorni, A. 1996. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 26: 29-41.

Dorigo, M., and Stützle, T. 2004. *Ant colony optimization*. MIT Press, MA.

Duda, R., and Hart, P. 1972. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, 11-15.

Fernandes, L.A.F., et al. 2008. Real-time line detection through an improved Hough transform voting scheme. *Pattern Recognition*. vol. 41: 299-314.

Hough, P.V.C., 1962. Methods and means for recognizing complex patterns. *US patent 3069654*.

Saitoh, F. 1998. Curve extraction using genetic algorithm based on closeness and continuity in perceptive grouping factors. *IAPR Workshop on MVA 98*: 488-493.

Sklansky, J. 1978. On the Hough technique for curve detection. *IEEE Transactions on Compute*, vol.27: 923-926.

Wei, W., et al. 2005. The feature extraction of nonparametric curves based on niche genetic algorithms and multi-population competition. *Pattern Recognition Letters*, vol. 26: 1483-1497.