ELSEVIER

Pattern Recognition Letters 18 (1997) 1443–1451

Pattern Recognition Letters

A tabu search method for geometric primitive extraction [1]

Qifa Ke [*], Tianzi Jiang [2], Song De Ma

National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, P.O. Box 2728, Beijing 100080, P.R. China

Received 5 October 1996; revised 16 June 1997

Abstract

In this paper, we propose a novel method for extracting the geometric primitives from geometric data, which is essentially an optimization problem. Specifically, we use tabu search to solve geometric primitive extraction problem. To the best of our knowledge, it is the first attempt that tabu search is used in computer vision. Our tabu search (TS) has a number of advantages: (1) TS avoids entrapment in local minima and continues the search to give a near-optimal final solution; (2) TS is very general and conceptually much simpler than either simulated annealing (SA) or genetic algorithm (GA); (3) TS has no special space requirement and is very easy to implement (the entire procedure only occupies a few lines of code); (4) our TS-based method can successfully extract some geometric primitives which are specially difficult for the traditional methods such as Hough Transform (HT) and Robust Statistics (RS). TS is a flexible framework of a variety of strategies originating from artificial intelligence and is therefore open to further improvement. © 1997 Published by Elsevier Science B.V.

Keywords: Geometric primitive; Tabu search; Model-based vision

1. Introduction

Extracting predefined geometric primitives from geometric data is an important problem in the field of model-based vision because it is a prerequisite to solving other problems in model-based vision, such as pose determination, model building, and object recognition, and so on. Hough Transform (HT) (Levine, 1985) and Robust Statistics (RS) (Huber, 1981; Meer et al., 1991) are the most commonly used methods for geometric primitive extraction. Roth and Levine (1993) proved that extracting the best geometric primitive from a given set of geometric data is equivalent to finding the optimum value of a cost function. Once it is understood that primitive extraction is such an optimization problem, the use of any technique for tackling optimization problem suggests itself. The objective function of the global optimization problem for geometric primitive extraction has potentially many local minima. Conventional local search minimization techniques are time consuming and tend to converge to whichever local minimum they first encounter. These methods are

[*] Corresponding author. E-mail: ke@tulip.ia.ac.cn.

[1] This work was partially supported by the Chinese National Science Fundation, the Chinese National Key Project for Basic Researchers, and the Chinese National High Technology Program (863).

[2] E-mail: jiangtz@prlsun6.ia.ac.cn.

PII S0167-8655(97)00052-4

unable to continue the search after a local minimum is reached. The key requirement of any global optimization method is that it must be able to avoid entrapment in local minima and continues the search to give a near-optimal final solution whatever the initial conditions. It is well known that *simulated annealing* (SA) and *genetic algorithm* (GA) meet this requirement. Roth and Levine (1994) suggested to solve this problem using the genetic algorithm.

In this paper we use Glover's tabu search to solve this problem. In the best of our knowledge, it is the first attempt that tabu search is used in computer vision. Our tabu search (TS) method has a number of advantages: (1) TS avoids entrapment in local minima and continues the search to give a near-optimal final solution. (2) TS is very general and conceptually much simpler than either SA or GA. (3) TS is a flexible framework of a variety of strategies originating from artificial intelligence and is therefore open to further improvement. In addition, our TS-based method fulfills some tasks of geometric primitive extraction which are specially difficult for the HT and RS method.

The outline of this paper is as follows. Section 2 devotes to the background of the extracting geometric primitives and the motivation of this paper. Section 3 describes our tabu search for geometric primitive extraction in detail. Section 4 devotes to the implementation of the algorithm. Section 5 is the conclusion.

2. Primitive extraction and minimal subsets

In this section, we briefly review the definition and some facts of primitive extraction and minimal subsets. We refer the readers to (Roth and Levine, 1993) for the details.

A geometric primitive is a curve or surface which can be described by an equation with a number of free parameters. The input to a primitive extraction algorithm consists of N geometric data points in two or three dimensional Cartesian space, which are labeled $p_1, p_2, \ldots, p_N$, along with the equation defining the type of geometric primitive to be extracted. We assume that this defining equation is an implicit form $f(p, \boldsymbol{a}) = 0$. Here p is the datum points, and $\boldsymbol{a}$ defines the parameter vector for this particular primitive. This assumption is not restrictive because it has been shown that the parametric curves and surfaces used to define the parts in CAD databases can be converted to implicit form (Sederberg and Anderson, 1984). The output consists of the parameter vector $\boldsymbol{a}$ of the best primitive, along with the subset of the geometric data that belongs to this primitive. We define the residual r_i as the closest distance of ith point of the geometric data to the curve or surface. Given that residuals $r_1, r_2, \ldots, r_N$ have been calculated, then extracting a single instance of a geometric primitive is equivalent to finding the parameter vector $\boldsymbol{a}$ which minimizes the value of a cost function $h(r_1, r_2, \ldots, r_N)$. Denote $f(\boldsymbol{a}) = h(r_1, r_2, \ldots, r_N)$. Therefore, extracting the best geometric primitive from a given set of geometric data is equivalent to solving the global optimization problem:

$$\min_{\bar{\boldsymbol{a}} \in A} f(\bar{\boldsymbol{a}}), \tag{1}$$

where A is a set of feasible solutions. A *minimal subset* is the smallest number of points necessary to define a unique instance of a geometric primitive. It is possible to convert from a minimal subset of points to the parameter vector $\boldsymbol{a}$ for a wide variety of geometric primitives (Roth and Levine, 1993). Only values of the parameter vector defined by these minimal subsets are potential solutions to the extraction problem, thus the search space is reduced by using minimal subsets. Therefore, instead of searching the best parameter vector $\boldsymbol{a}$ in the parameter space, we will search the best minimal subset from the given geometric data.

3. Tabu search

3.1. Tabu search: a general statement

Here we briefly review some notations of tabu search and outline the basic steps of the tabu search procedure for solving optimization problems. We refer the readers to (Glover, 1993) for the details.

Tabu search has been proven effective for many optimization problems. It is a metaheuristic method that guides a local heuristic search procedure to explore the solution space beyond local optimality. It is different from the well-known hill-climbing local search techniques because the tabu search allows moves out of a current solution that makes the objective function worse in the hope that it eventually will achieve a better solution. It is also different from the simulated annealing (Kirkpatrick et al., 1983) and genetic algorithms (Lutton and Martinez, 1994; Goldberg, 1989) because the tabu search includes a memory mechanism. According to Glover's idea (Glover, 1993), in order to solve a problem using tabu search, the following components must be defined.

- *Configuration*: Configuration is a solution or an assignment of values to variables.
- *Move*: A move characterizes the process of generating a feasible solution to the problem that is related to the current solution (i.e. a move is a procedure by which a new solution is generated from the current one).
- *Neighbourhood*: A neighbourhood of the solution is the collection set of all possible moves out of a current configuration. Note that the actual definitions of the neighbourhood depend on the particular implementation and the nature of problem.
- *Tabu conditions*: In order to avoid a blind search, tabu search technique uses a prescribed problem-specific set of constraints, known as tabu conditions. They are certain conditions imposed on moves which make some of them forbidden. These forbidden moves are known as *tabu moves*. It is done by forming a list of certain size that records these forbidden moves. This is known as *tabu list*.
- *Aspiration conditions*: These are rules that override tabu restrictions, that is, if a certain move is forbidden by tabu restriction, then the aspiration criterion, when satisfied, can make this move allowable.

With the above basic components, the tabu search algorithm can be described as follows.

1. Start with a certain (current) configuration and evaluate the criterion function for that configuration.
2. Follow a neighbour of the current configuration, that is, a set of candidate moves. If the best of these moves is not tabu or if the best is tabu, but satisfies the aspiration criterion, then pick that move and consider it to be the new current configuration; otherwise pick the best move that is not tabu and consider it to be the new current configuration.
3. Repeat (i) and (ii) until some termination criteria are satisfied.

The best solution in the final loop is the solution obtained by the algorithm. Note that the move picked at a certain iteration is put in the tabu list so that it is not allowed to be reversed in the next iterations. The tabu list has a certain size, and when the length of the tabu reaches that size and a new move enters that list, then the first move on the tabu list is freed from being tabu and the process continues (i.e. the tabu list is circular).

3.2. Tabu search techniques for geometric primitive extraction

In this section, we develop a new algorithm for extracting geometric primitives. Each of the N geometric data points in the input has an associated index, which is a number from 1 to N. Then a minimal subset is identified by the indices of its member points. Let m be the size of minimal subset of the particular primitive we want to extract. Let $I = (I_1, \dots, I_m)$ denote the minimal subset of size m, where I_i corresponds to the indices of its member points. For the sake of convenience, we assume that $I_i < I_j$ for $i < j$, $i,j = 1, \dots, m$. For any minimal subset I, the objective function $f(I)$ is defined as follows:

$$f(I) = \sum_{i=1}^{m} s\left(r_{I_i}^2\right), \tag{2}$$

where s is step function; $s = 1$ if r_j is greater than or equal to the template width, and $s = 0$ otherwise. This objective function counts the number of points within a fixed distance of the geometric primitive. Moreover, it effectively matches a small template around this primitive to the geometric data.

Let I_{c}, I_{t} and I_{b} denote the current, trial and best configurations (minimal subsets) and f_{c}, f_{t} and f_{b} denote the corresponding current, trial and best objective function values, respectively. As described in the previous section, we operate with a configuration which is known as the current solution I_{c} and then through *moves*, as explained in Section 3.1, we generate trial solutions I_{t}. As the algorithm proceeds, we also save the best solution found so far which is denoted by I_{b}. Corresponding to these configurations, we also operate with the objective function values f_c, f_t and f_b, respectively.

For the geometric primitive extraction problem, our tabu search algorithm can be described as follows (modified from (Al-sultan, 1995)):

1. *Initialization*: Let I_c be an arbitrary solution, and f_c be the corresponding objective function value computed using Eq. (2). Let $I_b = I_c$ and $f_b = f_c$. Select values for the following parameters: MTLS (tabu list size), P (probability threshold), NTS (number of trial solutions) and let IMAX be the maximum number of iterations. Let $k = 1$, where k is the iteration step indice. We begin with empty tabu list, that is, set the counter TLL(tabu list length) = 0 and go to Step 2.
2. *Generating neighbourhood*: Using I_c, generate NTS trial solution $I_t^1, I_t^2, \ldots, I_t^{\mathrm{NTS}}$ (see Remark 1) and evaluate their corresponding objective function values $f_t^1, f_t^2, \ldots, f_t^{\mathrm{NTS}}$ and go to Step 3.
3. Order $f_t^1, f_t^2, \ldots, f_t^{\mathrm{NTS}}$ in an ascending order, and denote them by $f_t^{[1]}, f_t^{[2]}, \ldots, f_t^{[\mathrm{NTS}]}$. If $f_t^{[1]}$ is not tabu (see Remarks 2 and 3), or if it is tabu but $f_t^{[1]} < f_b$ (i.e., the aspiration condition is satisfied), then let $I_c = I_t^{[1]}$ and $f_c = f_t^{[1]}$, and go to Step 4; otherwise, let $I_c = I_t^{[L]}$ and $f_c = f_t^{[L]}$, where $f_t^{[L]}$ is the best objective function of $f_t^{[2]}, \ldots, f_t^{[\mathrm{NTS}]}$ that is not tabu and go to Step 4. If all $f_t^{[1]}, f_t^{[2]}, \ldots, f_t^{[\mathrm{NTS}]}$ are tabu go to Step 2.
4. Insert the current solution denoted by minimal subset I_c at the bottom of the tabu list and let TLL = TLL + 1 (if TLL = MTLS + 1, delete the first element in the tabu list and let TLL = TLL − 1). If $f_b > f_c$, let $I_b = I_c$ and $f_b = f_c$. If $k =$ IMAX, stop (I_b is the best solution found and f_b is the corresponding objective function value); otherwise, let $k = k + 1$ and go to Step 2.

Remark 1. Given a current solution I_c, one can generate a trial solution using several strategies. We use the following strategy. Given I_c and a probability threshold P, for $i = 1,2,\ldots,m$, draw a random number $R_i \sim u(0,1)$, where $u(0,1)$ is the uniform distribution on interval [0,1]. If $R_i < P$, then $I_t(i) = I_c(i)$; otherwise draw randomly an integer $\hat{l}$ from the set $\{l \mid l = 1,2,\ldots,m;\ l \neq I_c(j),\ j = 1,2,\ldots,m\}$ and let $I_t(i) = \hat{l}$.

Remark 2. We use the following method to determine whether a solution is tabu. Let $\boldsymbol{a}_t$ denote the parameter vector of the trial solution I_t (a minimal subset determines uniquely a geometric primitive, thus a unique corresponding parameter vector (Roth and Levine, 1993)). If there exists $I_i \in$ Tabulist which satisfies: $\|\boldsymbol{a}_t - \boldsymbol{a}_i\| < \varepsilon$ (where $\boldsymbol{a}_i$ and $\boldsymbol{a}_t$ are parameter vectors determined by I_i and I_t respectively), then $\boldsymbol{a}_t$ is tabu. Here $\|\cdot\|$ denotes the Euclidean distance and ε is a predefined small positive real number.

Remark 3. To extract the primitives of circle and ellipse from real images, we use another tabu restriction, that is, if the number of the edge pixels belonging to the current configuration is less than half of the pixel number of a corresponding complete circle or ellipse, then the minimal subset I_c and its corresponding parameter $\boldsymbol{a}_c$ are inserted into the tabu list and I_b, f_b are not updated in Step 4. It means that the occluded part of the circle or ellipse couldn't be more than half, i.e., at least half of the circle or ellipse should appear in the images.

4. Implementation of the algorithm

In this section, we present our experiment results on extracting various kinds of geometric primitives. We will also make some comparison of our method with traditional methods: robust statistics (RS) and Hough transform (HT). As mentioned in the previous section, the tabu search has three parameters MTLS, P and NTS. The tabu list enables the algorithm has short term memory. A large tabu list size allows more diversification while a small list size makes the algorithm more forgetful, i.e., allows intensification to happen. Determining the list size is a non-trivial problem. Glover (1990) suggests using a tabu list size in the range $[\frac{1}{3}n, 3n]$, where n is the size of problem. Recently, some researchers (Battiti et al., 1984; Xu et al., 1996) proposed variable tabu list size (tabu tenure). In our experiment, to make the procedure more simple and reliable, we do some test and then determine appropriate sizes for various kinds of images and geometric primitives. If the *real* image has a relatively high quality with a size about 320×320, then the parameters of (MTLS = 20, NTS = 10, $P = 0.6$) for circle and line extraction and (MTLS = 25, NTS = 15, $P = 0.8$) for ellipse extraction give the best results. Images with more noises or complex background require larger size of MTLS and NTS to find the best geometric primitives efficiently.

We first compare our algorithm with simulated annealing (SA) and genetic algorithm (GA) on synthetic data. Fig. 1 contains an ellipse and noise data. There are 50 points on the ellipse and 450 points scattering on the whole image. Fig. 2 plots the best object function value Fb versus the number of iterations. Here Fb is the number of points belonging to the extracted geometric primitives. All the three algorithms are able to extracted the ellipse but our algorithm is the best and fast, as shown in Fig. 2.

To extract primitives from real images, the first step is to obtain the geometric data. Geometric data could be obtained by active sensors or edge detection. In our experiment, geometric data are produced by zero-crossing edge detector (Ma, 1997).

To extract multiple primitives, we just repeat the algorithm presented in the above section. Following the first application of the algorithm, the data points belonging to the first geometric primitive are removed, and the next application of the algorithm has as input the remaining geometric data points. A more complex situation is the extraction of not only multiple primitives, but also different types of primitives. In this case, we apply the algorithm for lines, circles and ellipses simultaneously on the same geometric data. The best primitive is extracted and the algorithm repeated.

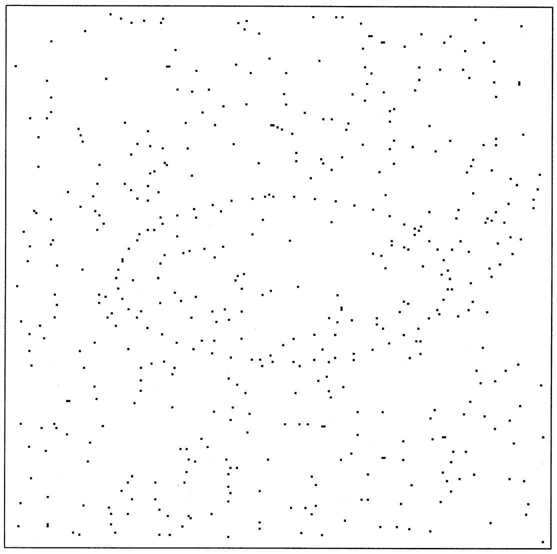
Fig. 1. The image of synthetic image, which contains 50 points on the ellipse and 450 points scattering all over the image.

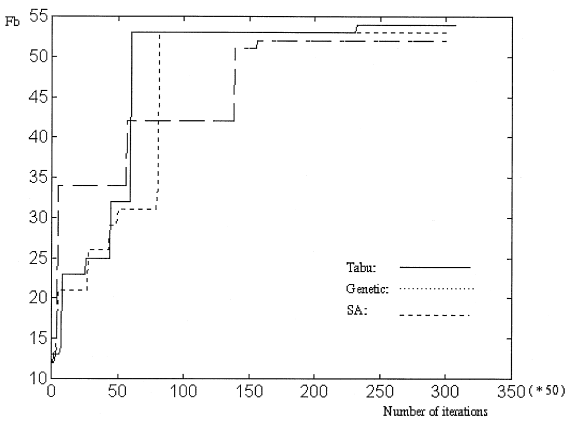

Fig. 2. The best object function versus the number of iterations.

Fig. 3 shows the extraction of multiple lines. Part (a) is the initial image, part (b) is the edge pixels of (a), part (c) shows the extracted lines while part (d) shows the extracted lines superimposed on the edge data. As can be seen from this figure, all of the correct lines have been found. This example is also significant in that it shows that our TS-based algorithm can reliably extract a primitive which contains far fewer than 50% of the geometric data while it has been consistently claimed that in order for the Robust Statistics methods to work the percentage of the geometric data belonging to a single primitive must be at least 50% of the total (Huber, 1981). This advantage of our algorithm can be seen more clearly in the following examples of Figs. 4–7.

Extracting circles and ellipses using HT is still an active area of research. Because the number of parameters of these primitives is higher than two (3 for circle and 5 for ellipse), it is space inefficient and therefore difficult for HT to perform the extraction of circle, especially the ellipse. Our approach has no such difficulty, as can be seen from Figs. 4–6. Fig. 4 shows the extraction of multiple circles from edge data. The largest circle is the *best* primitive and is extracted first, then the algorithm is repeated and the next circle extracted, and so on until all of the five circles are extracted. The fact that the circles are

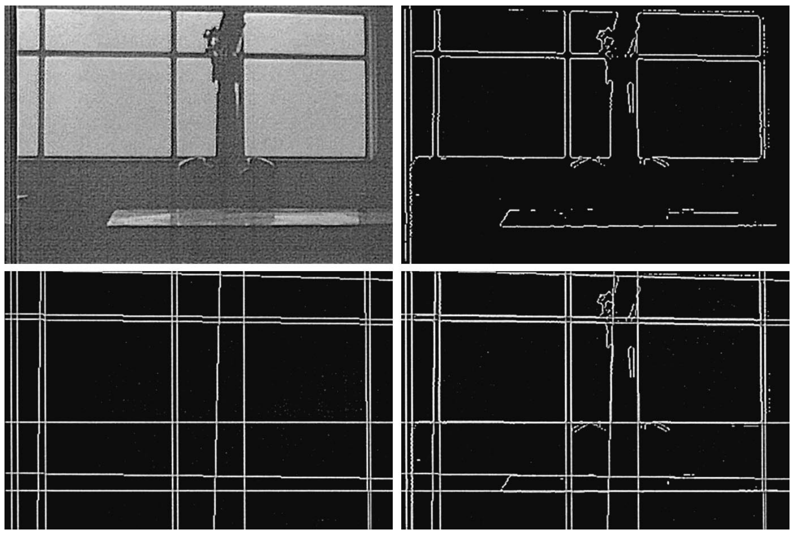
Fig. 3. Extracting straight lines. (a) Original indoor image. (b) Edge of (a). (c) The extracted lines. (d) Extracted lines superimposed on the edge image.

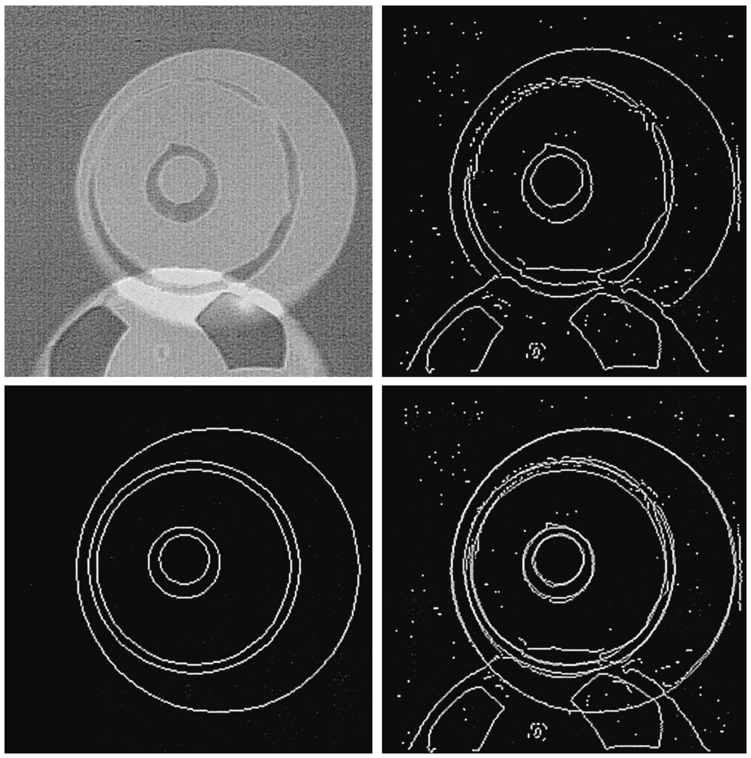
Fig. 4. Extracting straight lines. (a) Original image. (b) Edge of (a). (c) The extracted circles. (d) Extracted circles superimposed on the edge image.

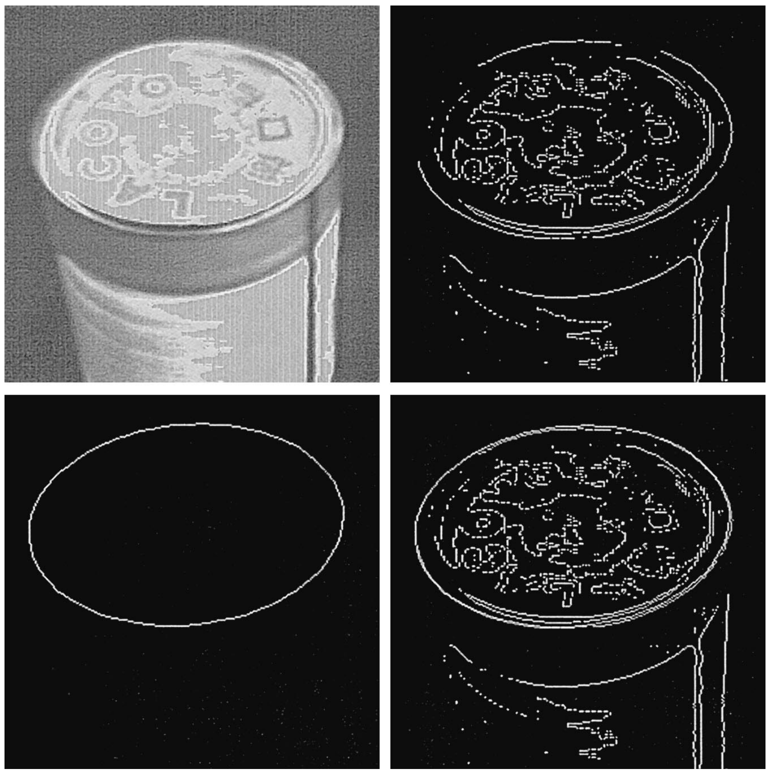

Fig. 5. Extracting ellipse. (a) Original can image. (b) Edge of (a). (c) The extracted ellipse. (d) Extracted ellipse superimposed on the edge image.

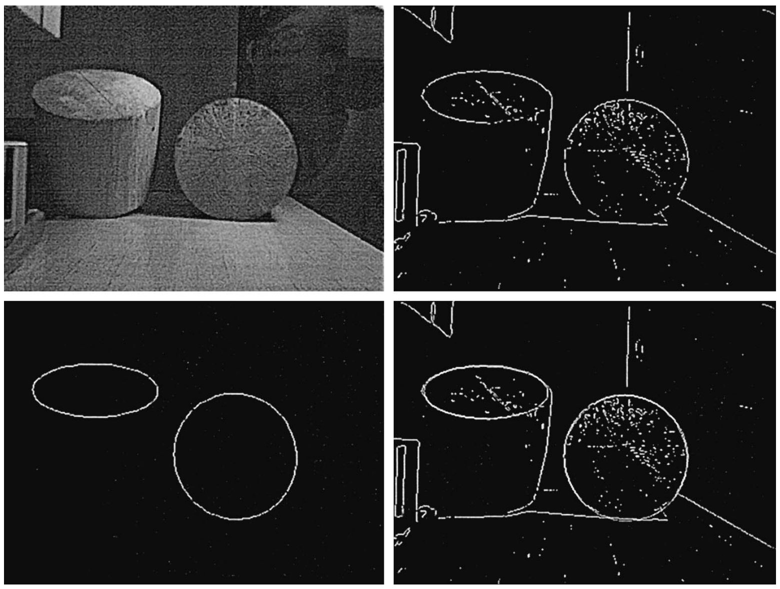

Fig. 6. Extracting circle and ellipse. (a) Original image. (b) Edge of (a). (c) The extracted ellipse. (d) Extracted ellipse and circle superimposed on the edge image.

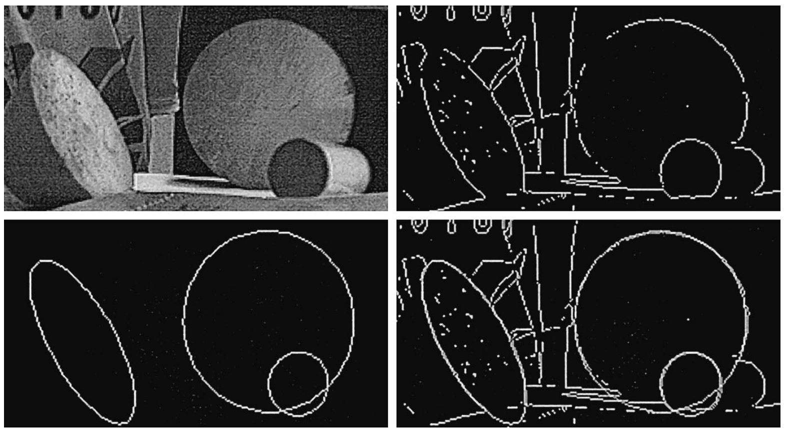

Fig. 7. Extracting circle and ellipses. (a) Original can image. (b) Edge of (a). (c) The extracted ellipse. (d) Extracted ellipse superimposed on the edge image.

very close together makes this a particularly difficult example for the RS method (Meer et al., 1991) and the HT method (Yuen et al., 1989) (because of the coarse quantization necessary for HT to deal with higher dimensional spaces), while our TS-based algorithms can fulfill the extracting task quickly and correctly, as can be seen from this figure. Fig. 5 shows the extraction of an ellipse from a complex background and Fig. 6 shows the extraction of different kinds of primitives (circle and ellipse).

Fig. 7 shows the extraction of three ellipses (here circle is considered as a special kind of ellipse). This example is significant in that: (1) the background is complex and the ellipses are occluded; (2) the small ellipse in the bottom-right corner is successfully extracted, which is very difficult for HT to extract. The last step of HT is peak detection and it is difficult in detecting a very small real peak. Case becomes more serious when the original images are contaminated by noise.

From above experiments, we see that our method performs the task of geometric primitive extraction efficiently and correctly. Our algorithm is coded in C language and tested on Sparc-10 station. The time for extracting a primitive from Figs. 3–7 is on the order of 1 second (about 0.5 second for a line (Fig. 4) and 1–2 seconds for a circle or ellipse (Figs. 4–7)). More complex images require more time to extracted the geometric primitive since more number of cost function evaluations and more time for evaluating the cost function itself are needed. Our TS-based algorithm has not specially requirement of space while HT has a heavy burden of space requirement.

5. Conclusion

Extracting predefined geometric primitives from geometric data is an important problem in the field of model-based vision because it is a prerequisite to solving other problems in model-based vision, such as pose determination, model building, and object recognition. In this paper we use Glover's tabu search to solve this problem. In the best of our knowledge, it is the first attempt that tabu search is used in computer vision. Our tabu search (TS) has a number of advantages. (1) TS avoids entrapment in local minima and continues the search to give a near-optimal final solution. (2) TS is very general and conceptually much simpler than either SA or GA. (3) TS is very easy to implement and the entire procedure only occupies a few lines of code. (4) Our method fulfills some tasks of geometric extraction which are specially difficult for the HT and RS methods. (5) Compared with HT, TS can be applied to a much wider variety of primitives and can be easily parallelized (Glover, 1993). The main advantage of HT is that it extracts all the primitives at once while our TS-based method must be repeatedly applied to the geometric data to extract all of the primitives. However, the TS is still practical if the number of the primitives is limited, such assumption is quite reasonable in model-based vision. In a word, TS is a flexible framework of a variety of strategies originating from artificial intelligence and is therefore open to further improvement. Now we are considering the application of Tabu search to other problems in computer vision.

Acknowledgements

We appreciate Mr Jinxiang Chai for performing the experiment of Fig. 2.

References

Al-sultan, K.S., 1995. A tabu search approach to the clustering problem. Pattern Recognition 28 (9), 1443–1451.

Battiti, R., et al., 1984. The reactive tabu search. ORSA J. Comput. 6, 126–140.

Glover, F., 1990. Artificial intelligence, heuristic frameworks and tabu search. Manag. Decis. Econom. 11, 365–375.

Glover, F., 1993. Tabu search. In: Reeves, C.R. (Ed.), Modern Heuristic Techniques for Combinatorial Problems. Wiley, New York.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Huber, P., 1981. Robust Statistics. Wiley, New York.

Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P., 1983. Optimization by stimulated annealing. Science 220, 621–680.

Levine, M.D., 1985. Vision in Man and Machine. McGraw-Hill, New York.

Lutton, E., Martinez, P., 1994. A genetic algorithm for the detection of 2D geometric primitive in images. In: Proc. ICPR'94, Vol. 1, pp. 526–528.

Ma, S.D., et al., 1997. Multiscale derivative computation. Image Vision Comput., to appear.

Meer, P., et al., 1991. Robust regression methods in computer vision: A review. Internat. J. Comput. Vision 6, 59–70.

Roth, G., Levine, M.D., 1993. Extracting geometric primitives. CVGIP: Image Understanding 58, 1–22.

Roth, G., Levine, M.D., 1994. Geometric primitive extraction using a genetic algorithm. IEEE Trans. Pattern Anal. Machine Intell. 16 (9), 901–905.

Sederberg, T.W., Anderson, D.C., 1984. Implicit representation of parametric curves and surfaces. Computer Vision, Graphics, Image Process. 28, 72–84.

Xu, J., Chiu, S., Glover, F., 1996. Fine-tuning a tabu search algorithm with statistical tests. Technical Report, University of Colorado at Boulder.

Yuen, H., et al., 1989. Detecting partically occluded ellipses using the Hough transform Image Vision Comput. 7.