Efficient Search of Faces from Complex Line Drawings

Jianzhuang Liu and Xiaoou Tang
Department of Information Engineering
The Chinese University of Hong Kong, Hong Kong, China
jzliu@ie.cuhk.edu.hk, xtang@ie.cuhk.edu.hk

Abstract

Single 2D line drawing is a straightforward method to illustrate 3D objects. The faces of an object depicted by a line drawing give very useful information for the reconstruction of its 3D geometry. Each of the two recently proposed methods for face identification from line drawings involves two combinatorial problems. When dealing with complex objects having more faces, the combinatorial explosion prevents these methods from practical uses. This paper proposes a new approach to tackling the face identification problem by a variable-length genetic algorithm (GA) with geometric constraints and a novel heuristic incorporated for local search. The hybrid GA solves the two combinatorial problems simultaneously. Experimental results show that our algorithm can find the faces of a line drawing having more than 30 faces much more efficiently.

1. Introduction

In computer vision, an important research area is to develop algorithms that can understand a single 2D line drawing representing an object and can reconstruct its 3D geometry. One application of this research is in CAD, where tools are highly desirable that can convert a design sketch into a 3D model directly. An object is made up by faces. If the face configuration of an object is known, the complexity of the 3D reconstruction will be reduced significantly. Roughly speaking, the conversion problem can be divided into two sub-problems: face identification and 3D geometry reconstruction. In this paper, we study the face identification problem only. For the 3D geometry reconstruction, the reader is referred to references [1], [2], [3], [4].

A 2D line drawing in this paper is defined as the projection of a wireframe object where all the edges and vertices of the object are visible and the drawing can be represented by a single edge-vertex graph. A drawing with hidden lines visible makes it possible to reconstruct its complete 3D model. Fig. 1 shows such a line drawing together with its faces. Much effort has been made in this face identification problem over the past two decades [5]. The recent work presented in [6] and [7] can handle a larger range of objects than previous methods. Both of them include two steps: finding a set of circuits that may be potential faces and searching for faces from this set. It needs to be emphasized that the two steps in each of the two methods correspond to two combinatorial problems. The number of circuits is generally exponential in the number of edges of a line drawing. In Section 4, it can be seen that the combinatorial explosion prevents the two methods from handling a line drawing having many faces (> 30) within feasible time.

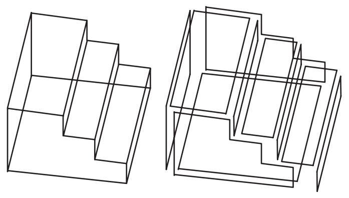


Figure 1. A line drawing with ten faces.

Genetic algorithms (GAs) are a class of probabilistic search algorithms that emulate natural evolutionary process of human beings. GAs, if well designed, can often outperform traditional optimization methods. Many successful applications of GAs to various problems have been published in the literature [8], [9]. In this paper, we design an efficient GA with variable-length chromosomes to solve the face identification problem. Geometric constraints on circuits of a line drawing and a heuristic called *minimal edge face phenomenon* are developed and incorporated into the operations of the GA. The proposed hybrid GA tackles simultaneously the two combinatorial problems involved in the previous methods [6], [7]. Our experiments show that the new algorithm reduces the computational complexity of face identification from exponential to linear with respect to the number of edges in line drawings.

2. Two related methods

Among a number of published approaches in face identification [5], the two in [6] and [7] can handle a larger range of objects (manifolds and non-manifolds as shown in Fig. 1 and in Section 4). These two methods are most related to the work in this paper, and will be briefly described in the following. Before that, we list several terms here that will be used in the rest of the paper.

- **Circuit.** A circuit is a closed trail in a graph where all its vertices except the end vertices are distinct.
- **Potential face.** A potential face is a circuit without edges intersecting.
- **Minimal potential face (MPF).** A MPF is a potential face without an edge connecting its nonadjacent vertices.
- **Degree.** The degree of vertex v, $d(v)$, is the number of edges adjacent to v.
- **Vertex rank.** The rank of vertex v, $R(v)$, denotes the number of faces with boundaries passing through v.
- **Edge rank.** The rank of edge e, $R(e)$, denotes the number of faces with boundaries passing through e.

Shpitalni and Lipson's face identification method [6] is built upon an observation on face configuration and a basic theorem called the *face adjacency theorem*. The observation, serving as the *criterion* for the problem, is that given a line drawing, human beings tend to choose a face configuration in which there are as many edges as possible. The face adjacency theorem states that two adjacent planar faces may coexist in the same object if and only if their common edges are collinear. The method can be formulated as follows.

Definition 1 *Given n MPFs generated from a line drawing, the maximum ranks $R^+(e)$ and $R^+(v)$ of all the edges and vertices, and a binary matrix $B = [b_{ij}]_{n\times n}$, the face identification problem is to search for the subset x of the MPFs such that*

$$\text{minimize:} \quad \sum[R^+(e) - R(e)] + \sum[R^+(v) - R(v)] \tag{1}$$

$$\text{subject to:} \quad R(e) \le R^+(e),\ \forall e \tag{2}$$

$$R(v) \le R^+(v),\ \forall v \tag{3}$$

$$b_{ij} = 1,\ i \ne j,\ i,j \in x \tag{4}$$

where $R(e)$ and $R(v)$ are the respective actual edge and vertex ranks in x, and B is obtained according to the face adjacency theorem with $b_{ij} = 1$ (0) denoting that faces i and j can (cannot) coexist in the same object.

In this formulation, (1) implies the criterion and (4) reflects the geometric constraint imposed by the face adjacency theorem. Shpitalni and Lipson calculated the maximum ranks $R^+(e)$ and $R^+(v)$ from a line drawing through an iterative procedure (see [6] for details).

Liu and Lee [7] used the same criterion and face adjacency theorem to formulate the problem. They indicated that the maximum edge ranks of a line drawing can be calculated directly by

$$R^+(e) = \min\{d(v_1), d(v_2)\} - 1. \tag{5}$$

We use (5) to compute the maximum edge ranks of a line drawing in the GA implementation. The face identification in [7] is formulated as follows.

Definition 2 *Let $w(i)$ be the number of edges of a circuit i. Given a line drawing and the set of the MPFs, $SMPF$, generated from the drawing, the problem is to*

$$\text{maximize:} \quad f(x) = \sum_{i\in x} w(i),\ x \subset SMPF \tag{6}$$

$$\text{subject to:} \quad b_{ij} = 1,\ i \ne j,\ i,j \in x. \tag{7}$$

Liu and Lee [7] proved that the two formulations in Definitions 1 and 2 are equivalent, and developed a much faster algorithm to find the faces in a drawing based on Definition 2.

Each of the above-mentioned two methods involves two combinatorial problems. The method in [6] generates the MPFs by a circuit space approach, and uses the A* algorithm to search for the optimal solution on a tree constructed by the MPFs. The method in [7] employs a depth-first search algorithm to find the MPFs, and develops a maximum weight clique finding algorithm to solve the problem in Definition 2. Our experiments show that neither method works for line drawings with more than 30 faces within feasible time.

3. A hybrid GA for face identification

In this section, we develop a GA with variable-length chromosomes and a local search heuristic for the face identification. Incorporated into the local search algorithm are the maximum ranks, geometric constraints imposed by the face adjacency theorem, and a novel heuristic called minimal edge face phenomenon. The hybrid GA is designed to find solutions by direct search on a line drawing.

Standard GAs generally have these five basic components [9]:

- A genetic representation of solutions to a problem.
- A scheme to create an initial population of solutions.
- A fitness function to evaluate how good a solution is.
- Genetic operators used to generate offspring.
- Parameters of GAs.

A GA searches some solution space by maintaining a population $P(t)$ of individuals (chromosomes) in each generation t. A solution is encoded into a chromosome. The main advantage of maintaining a population is that with many different solutions available, the search has greater chances to find global optima. The overall quality (fitness) of the solutions is improved generation after generation. In each generation, some individuals undergo stochastic transformations through genetic operators to create new individuals (offspring) $O(t)$. Two commonly-used operators are *crossover* and *mutation*. A new generation is formed by the selection of fitter individuals from $P(t)$ and $O(t)$. It is expected that the GA converges to the best solutions after a number of generations. The framework of the standard GA is rather general. A simple GA, however, does not usually give satisfactory results for a difficult optimization problem. In the following, we will design a GA to meet the nature of the face identification problem and to solve it efficiently.

3.1. Genetic representation and fitness function

The first step to design a GA is to encode a solution into a chromosome. A chromosome consists of a number of genes. For the face identification, we use a gene to represent a MPF. Unlike fixed-length chromosomes in most GAs, here the length of a chromosome is variable since the number of the faces of an object is unknown in advance. Fig. 2 shows a chromosome having m genes (MPFs) at some generation.

Given a line drawing with all the edges visible, human beings tend to choose a face configuration in which there are as many edges as possible. This is the criterion for our solution to the face identification problem (which is also the criterion in [6] and [7]). The MPFs kept in a chromosome give a possible solution to the problem. We define a fitness function used to evaluate a chromosome as follows.

Definition 3 *The fitness of a chromosome k is evaluated by*

$$f_k(x) = \sum_{i \in x} w(i) \tag{8}$$

where x is the set of compatible MPFs currently stored in the chromosome, and $w(i)$ denotes the number of edges of a MPF i.

Here maximizing $f_k(x)$ implies the criterion, and the term *compatible* imposes a constraint on the MPFs. Obviously, the more MPFs are added into a chromosome, the higher is its fitness. However, whether or not a new MPF can be added into the chromosome is determined by the geometric constraint, the face adjacency theorem. In other words, all the MPFs in a chromosome must be able to coexist in the same object. We call these MPFs compatible.

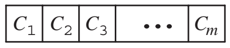


Figure 2. A chromosome having m genes.

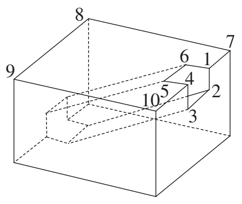


Figure 3. A block with a hole, where hidden edges are shown in dashed lines for easier observation.

3.2. Minimal edge face phenomenon

A face of an object corresponds to a circuit constructed by some edges in a line drawing. For an edge of a line drawing with all the vertices of degree > 1, we can always find a circuit passing through the edge and having fewest edges. Such a circuit is called a *minimal edge circuit*. Through extensive observation, we have found that it is very likely for a minimal edge circuit to stand for a face. Among all the edges of the line drawings given in this paper and in the two previous papers [6], [7], the percentage of a minimal edge circuit being a face is as high as about 95%. We call this the *minimal edge face phenomenon*.

Looking at the line drawing in Fig. 1, we can see that this phenomenon applies to all the edges. However, not all such circuits are faces of an object. An example can be seen from the drawing shown in Fig. 3 depicting a block with a hole passing through it. From edge $(3, 4)$, we can find a minimal edge circuit $(3, 2, 1, 6, 5, 4, 3)$ in the drawing but it is not a face. Moreover, not all faces are minimal edge circuits, such as the two faces each with eight edges in Fig. 1, and the face $(1, 7, 8, 9, 10, 4, 5, 6, 1)$ in Fig. 3.

Although there are exceptions, the minimal edge face phenomenon can apply to most cases. This heuristic leads to a very effective local search scheme described in the next section. Together with the GA, it efficiently solves the two combinatorial explosion problems involved in the previous methods [6], [7] simultaneously.

3.3. Local search scheme

Given a set of MPFs stored in a chromosome, the local search scheme tries to extend the chromosome by adding into it as many MPFs as possible. However, a new MPF, if added into the chromosome, has to be compatible with all the existing MPFs in it. Thus the following two conditions will be used in the local search algorithm:

• **Condition 1.** The new MPF does not cause the rank of any edge e in the line drawing to exceed the maximum edge rank $R^+(e)$.

• **Condition 2.** It can coexist in the line drawing with all the MPFs stored currently in the chromosome.

Now we discuss how to search for MPFs in order to fill a chromosome. We define a new term first, which will be used in Algorithm 1.

Definition 4 *Given a set x of MPFs currently stored in a chromosome, $RR(e) = R^+(e) - R(e)$ is called the remaining edge rank of an edge e, where $R(e)$ is obtained from the MPFs.*

When $x = \emptyset$, all the remaining edge ranks are equal to their corresponding maximum edge ranks. In this case, any MPF is allowed to appear in the chromosome. For an edge e, when $RR(e) = 0$, no new MPF can pass through this edge.

A new MPF is searched from a line drawing based on both the minimal edge face phenomenon and the remaining edge ranks. Algorithm 1 describes the pseudo-code of the local search algorithm.

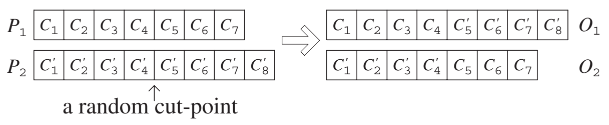
Algorithm 1. (Extending a chromosome)

[To add more minimal edge MPFs into the set x of MPFs currently stored in the chromosome by searching on the line drawing with the edges numbered from 1 to $|E|$, given the remaining edge ranks, $RR(e),\ e = 1, 2, ...|E|$.]

```
1. procedure $Extension(x)$
2. Generate a random permutation $(r_1, r_2, ..., r_{|E|})$ where
   $r_i \neq r_j$ for $i \neq j$, and $r_i, r_j, i, j \in \{1, 2, ..., |E|\}$
3. for $i = 1$ to $|E|$ do
4.    if $RR(r_i) > 0$ then
5.    begin
6.       Call $FindingMinimalEdgeMPF(r_i)$
7.       if there exists a new MPF and it can coexist
         with all the MPFs currently in $x$ then
8.          Add it to $x$ and update $RR$
9.    end
```

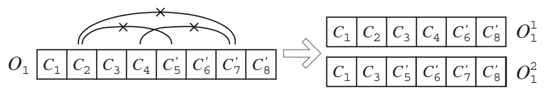
In this algorithm, the search for more MPFs is done by examining the edges one by one in a random way (Step 2) when the procedure is called. The purpose is to reduce the risk for the algorithm to get trapped in local maxima. Step 6 calls another procedure to search for one minimal edge MPF passing through edge r_i if the remaining rank $RR(r_i) > 0$. When there exists such a MPF, we have to test whether Condition 2 is satisfied (Step 7). If yes, the MPF is added into the chromosome and then the remaining edge ranks recorded in the array RR have to be updated (Step 8).

$FindingMinimalEdgeMPF()$ is a modified version of Moore's efficient breadth-first search algorithm for finding a shortest path in a graph [10]. Due to the limit of space, it is not given here. The interested reader may find the detail in [11].



P_1
C_1 C_2 C_3 C_4 C_5 C_6 C_7
P_2
C'_1 C'_2 C'_3 C'_4 C'_5 C'_6 C'_7 C'_8
a random cut-point
C_1 C_2 C_3 C_4 C'_5 C'_6 C'_7 C'_8
O_1
C'_1 C'_2 C'_3 C'_4 C_5 C_6 C_7
O_2

Figure 4. A single point crossover.



O_1
C_1 C_2 C_3 C_4 C'_5 C'_6 C'_7 C'_8
C_1 C_2 C_3 C_4 C'_6 C'_8
O_1^1
C_1 C_3 C'_5 C'_6 C'_7 C'_8
O_1^2

Figure 5. Two valid children from O_1.

3.4. Mutation and crossover

Mutation and crossover are two commonly-used operators in GAs. It is worth noting that after an operation on a chromosome, the MPFs in the chromosome must still be able to coexist with each other in the line drawing.

The process of mutation makes the genes of a selected chromosome undergo random changes with a small mutation rate. The motivation behind it is to introduce a diversity of solutions into the population. In our application of the GA, mutation is performed by simply deleting some genes (MPFs) in a chromosome with a small rate.

Crossover is the main operator in the GA. Its goal is to mate good chromosomes to generate better offspring. With a crossover rate, it operates on two selected chromosomes by combining the genes of the two. We use a single point crossover to produce two children O_1 and O_2 from two parents P_1 and P_2, as shown in Fig. 4. However, the MPFs in O_1 or O_2 may not be able to coexist in the same line drawing. Let us take O_1 as an example. Suppose each of the three MPF pairs, C_2 and C'_5, C_2 and C'_7, and C_4 and C'_7, cannot coexist in the drawing. To maintain valid children, some MPFs must be deleted. Two valid children O_1^1 and O_1^2 from O_1 are shown in Fig. 5, which are shorter and have most compatible MPFs in O_1. Similarly, we may obtain two valid children O_2^1 and O_2^2 from O_2.

It is common in GAs that two parents produce two children in order to maintain the same population size. Here we keep two best children out of O_1^1, O_1^2, O_2^1 and O_2^2, which can be seen from the hybrid GA given in the next section.

3.5. The hybrid GA

Combining the standard GA with the local search scheme, we formulate the hybrid GA in Algorithm 2, where $MaxGeneration$ denotes the maximum generation the algorithm will reach, and $PopulationSize$ is an even integer denoting the size of the population $P(t)$. The local search procedure $Extension$ (Algorithm 1) is incorporated into the standard GA, always trying to add more faces into chromosomes to make them fitter.

Algorithm 2. (Hybrid GA)

```
1.  $t \leftarrow 0$
2.  Initialize $P(t)$ by calling $Extension(x)$ with $x = \emptyset$
    for each chromosome in $P(t)$
3.  Calculate the fitness value of each chromosome in
    $P(t)$ using (8)
4.  **while** $t \leq MaxGeneration$ **do**
5.  **begin**
6.      Rank the chromosomes in $P(t)$ linearly based on
        their fitness values
7.      **for** $i = 1$ **to** $PopulationSize/2$ **do**
8.      **begin**
9.          Select two chromosomes $j$ and $k$ from $P(t)$
            based on the ranking
10.         Perform mutation on $j$ and $k$ with a mutation
            rate, producing two chromosomes $l$ and $m$
11.         Perform single-point crossover on $l$ and $m$
            with a crossover rate, producing another two
            chromosomes $n$ and $o$
12.         Delete incompatible MPFs in $n$ and $o$,
            producing four valid children $n_1$, $n_2$, $o_1$ and $o_2$
13.         Call $Extension$ to extend $n_1$, $n_2$, $o_1$ and $o_2$,
            and keep the best extended two in $O(t)$
14.     **end**
15.     Form $P(t+1)$ by selecting $PopulationSize/2$
        best chromosomes in $P(t)$ and $PopulationSize/2$
        best chromosomes in $O(t)$
16.     $t \leftarrow t + 1$
17. **end**
```

4. Experimental results

A number of experiments have been conducted to demonstrate that our hybrid GA can identify faces from line drawings very efficiently and robustly. We also compare the efficiency between the GA and the algorithm in [7]. We do not compare with the algorithm in [6] because the algorithm in [7] is already much faster than it. In what follows, LLA and HGA are short for the algorithm in [7] and the hybrid GA, respectively.

All the algorithms are implemented using Visual C++, running on a 1 GHz Pentium III PC. In HGA, the population size, maximum generation, mutation rate and crossover rate, are set to be 50, 15, 0.05 and 0.9, respectively, for all the experiments.

The first set of line drawings for testing HGA come from all the objects given in [6] and [7]. HGA finds the same faces in each object as the other two algorithms, and takes about 0.25 second each. It is not necessary to compare the computational times between HGA and LLA on these objects with less than 30 faces because LLA is also fast enough to handle them.

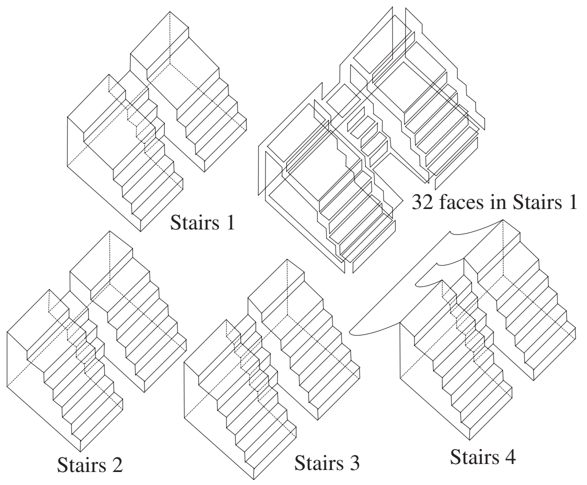


Figure 6. Four stairs models.

	Stairs 1	Stairs 2	Stairs 3	Stairs 4
Faces	32	36	38	39
Edges	90	102	108	114
MPFs	2561	8089	16428	48126
Memory in LLA	$>6.5\times10^6$	$>6.5\times10^7$	$>2.6\times10^8$	$>2.3\times10^9$
Memory in HGA	$<2.5\times10^5$	$<2.5\times10^5$	$<2.5\times10^5$	$<2.5\times10^5$
Time taken by LLA	9s	95s	1632s	?
Time taken by HGA	0.35s	0.43s	0.48s	0.50s

Table 1. Results for the four models in Fig. 6.

Next we show another set of objects each with more than 30 faces. In Fig. 6, four stairs models with increasing faces are shown. The 32 faces in Stairs 1 found by HGA are also shown. Table 1 summarizes the results for the four stairs. It is obvious that the number of MPFs grows exponentially with the number of edges, which causes both memory and computational time taken by LLA to increase exponentially. Let us consider the memory requirement first. In LLA (also in the algorithm in [6]), most memory consumption is due to the generation of the matrix $B = [b_{ij}]_{n \times n}$ with n being the number of MPFs (see Definitions 1 and 2). When there are 48126 MPFs in Stairs 4, LLA needs at least $48126 \times 48126 \simeq 2.3 \times 10^9$ basic memory units. In HGA, however, the memory requirement does not depend on the number of MPFs, but on the sizes of the two largest arrays to store chromosomes in $P(t)$ and $O(t)$, which together need a memory of less than

$$2 \times \text{Population} \times \text{Max\_Length\_Of\_A\_Chromosome} \times \text{Max\_Length\_Of\_A\_Circuit}.$$

In our experiments, the maximum lengths of a chromosome and a circuit are less than 50. Therefore, $P(t)$ and $O(t)$ to-

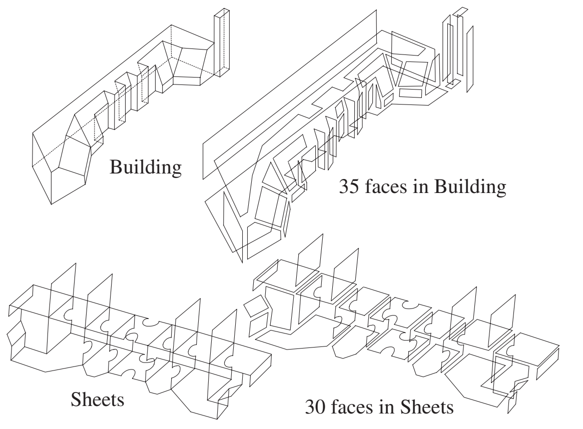


Figure 7. Two objects and their faces found.

gether take less than 2.5×10^5 basic memory units.

The last two rows in Table 1 give the times (in seconds) taken by the two algorithms. It is obvious that the time consumed by LLA grows exponentially. We do not give the time for LLA to deal with Stairs 4 because it did not finish its job after running for one day. On the contrary, we are very happy with HGA. It is much more efficient and its computational time increases approximately linearly with the number of edges of the stairs models.

Two more objects in Fig. 7 are used to test HGA and LLA. HGA takes 0.35 second and 0.45 second to deal with them, respectively, while LLA has to spend 96 seconds and 137 seconds, which again demonstrates the significant better performance of HGA over LLA.

GAs are a stochastic global optimization technique. It is not guaranteed that a GA will find optimal solutions to a problem. Actually, GAs return only nearly optimal solutions in most applications to combinatorial problems [9]. The likelihood of some GA finding optimal solutions depends on factors like how hard a problem is, how well the GA is designed, and the parameters chosen in the GA. For HGA, given the set of the parameters (population $= 50$, maximum generation $= 15$, mutation rate $= 0.05$, crossover rate $= 0.9$), it can find the optimal solutions in a very high probability. For all the line drawings in the experiments (including those in [6], [7]), we ran HGA on each object more than 500 times with random initializations, and did not find that HGA failed once. This fact indicates that HGA is not only efficient but also robust.

5. Conclusions

A hybrid GA for finding faces from single 2D line drawings has been presented. The faces identified from a line drawing provide important information for the reconstruction of its 3D geometry. Our strategy to conquer the combinatorial explosion in face identification is to combine the standard GA with a novel local search scheme. The former is well known for its good global search ability; the latter has high likelihood to find faces on a line drawing based on the maximum edge ranks, face adjacency theorem, and minimal edge face phenomenon. From the experiments, it can be seen that HGA finds the same faces as the previous two algorithms [6], [7] do, but exhibits significantly better performance for objects with faces > 30, both in computational time and memory requirement. The experiments also show that HGA is robust in finding optimal solutions.

Acknowledgments

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong SAR. (Project no. AoE/E-01/99).

References

[1] T. Marill, “Emulating the Human Interpretation of Line-Drawings as Three-Dimensional Objects,” *Int’l J. Computer Vision*, vol. 6, no. 2, pp. 147–161, 1991.

[2] Y.G. Leclerc and M.A. Fischler, “An Optimization-Based Approach to the Interpretation of Single Line Drawings as 3D Wire Frames,” *Int’l J. Computer Vision*, vol.9, no.2, pp. 113–136, 1992.

[3] H. Lipson and M. Shpitalni, “Optimization-Based Reconstruction of a 3D Object from a Single Freehand Line Drawing,” *Computer-Aided Design*, vol. 28, no. 8, pp. 651–663, 1996.

[4] K. Sugihara, “A Necessary and Sufficient Condition for a Picture to Represent a Polyhedral Scene,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 5, pp. 578–586, 1984.

[5] J. Liu, Y.T. Lee and W.-K. Cham, “Identifying Faces in a 2D Line Drawing Representing a Manifold Object,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, pp. 1579–1593, 2002.

[6] M. Shpitalni and H. Lipson, “Identification of Faces in a 2D Line Drawing Projection of a Wireframe Object,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, pp. 1000–1012, 1996.

[7] J. Liu and Y.T. Lee, “A Graph-Based Method for Face Identification from a Single 2D Line Drawing,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1106–1119, 2001.

[8] L. Chambers, *Practical Handbook of Genetic Algorithms*. Boca Raton: CRC Press, 1995.

[9] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Optimization*. New York: John Wiley & Sons, 2000.

[10] G. Chartrand and O.R. Oellermann, *Applied and Algorithmic Graph Theory*. New York: McGraw-Hill, 1993.

[11] J. Liu, “Efficient Search of Faces from Complex Line Drawings,” Technical Report, The Multimedia Lab, Dept. of IE, The Chinese University of Hong Kong, 2003.