Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Pattern Recognition Letters 24 (2003) 1279–1293

Pattern Recognition Letters

www.elsevier.com/locate/patrec

3D shape recovery of complex objects from multiple silhouette images

Yen-Hsiang Fang, Hong-Long Chou, Zen Chen [*]

Department of Computer Science and Information Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan

Received 7 July 2001; received in revised form 1 November 2002

Abstract

A reconstruction method is proposed which represents the object with a line-based geometric model. The method does not need the point correspondence information in recovering the 3D object geometry. It is based on the concept of volume intersection, but it is substantially different from the existing octree-based reconstruction methods in the aspects of data structure, reconstruction process and representation uniqueness under a 2D rigid motion. For visualizing the 3D reconstructed object geometry a conversion from the line-based geometric model to a bounded triangular mesh model is developed. The experimental results show that the method is capable of capturing the different details of the object. And it works fast and requires a relatively low memory space.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: 3D shape recovery; Volume intersection; Line-based model; Dynamic line resolution; Triangular mesh model

1. Introduction

Reconstruction of 3D objects from images is an important task in computer vision. The 3D geometric model of the object finds applications in part manufacturing and design, target recognition, and virtual reality, etc. We are concerned with computer vision techniques for recovering the geometry of a complex object from multiple silhouette images. Here complex objects refer to objects containing parts of varying size, for instance, teapot with flat handle and rifle with tiny trigger.

There are various vision-based reconstruction methods including:

(1) The stereo vision methods (Fua, 1997; Huang and Netravali, 1994; Okutomi and Kanade, 1993).
(2) The structured light methods (Proesmans et al., 1998; Chia et al., 1996; Hu and Stockman, 1989).
(3) The volume intersection methods (Slabaugh et al., 2001; Niem, 1999; Snow et al., 2000; Moezzi et al., 1996; Fromherz and Bichsel, 1994;

[*] Corresponding author. Tel.: +886-35-712-121; fax: +886-35-5723-148.
E-mail address: zchen@csie.nctu.edu.tw (Z. Chen).

Table 1
Characteristics of the vision-based reconstruction methods

Method	Sensing mode	Data model	Reconstruction scheme
Stereo vision	Passive	Point set	Triangulation
Structured light	Active	Point network	Triangulation
Volume intersection	Passive	Voxel set	Volume intersection
Voxel coloring	Passive	Voxel set	Color consistency

Garcia and Brunet, 1998; Szeliski, 1993; Srivastava and Ahuja, 1990; Potmesil, 1987; Chien and Aggarwal, 1986).

(4) The voxel coloring methods (Kutulakos and Seitz, 2000; Szeliski and Golland, 1999; De Bonet and Viola, 1999; Seitz and Dyer, 1999; Prock and Dyer, 1998).

The characteristics of these methods are given in Table 1 and each method has its own merits and shortcomings. For a practical object reconstruction we are looking for a widely applicable method that is capable of producing a complete object geometric model that is reasonably accurate. The method we are going to propose has the following distinct features:

(1) It does not need the point correspondence information in recovering the 3D object geometry, as required by the stereo vision method.
(2) It builds a complete and dense geometric model of the object. Generally, the geometric model built by the stereo vision method is a sparse reconstruction due to the limited number of available feature point unless an active structured light is used in the vision method.
(3) It is based on the concept of volume intersection, but it is substantially different from the related octree-based reconstruction methods (Garcia and Brunet, 1998; Szeliski, 1993; Srivastava and Ahuja, 1990; Potmesil, 1987; Chien and Aggarwal, 1986) in the aspects of data structure, reconstruction process, and representation uniqueness under a 2D rigid motion. We elaborate on these differences below:
    (a) Difference in the data model/structure: In the octree-based method the basic unit of the geometric model is the octree cell and the octree cells are organized in a tree data structure with a branch factor of 8. Our method uses line segment as the basic unit and the line segments are organized in a 2D array of linked lists.
    (b) Difference in the reconstruction process: The reconstruction process of the octree-based method is recursive, while our method constructs the line-based model sequentially.
    (c) Representation uniqueness under a 2D rigid motion: In the octree-based method, a 2D translation or rotation of a root cell will generally end up with a dramatic change in the octree representation form. Thus, the octree-based representation is sensitive to the 2D rigid motion. However, in our line-based modeling the 2D object motion only leads to a similar motion of the 2D array representation of the object without changing its intrinsic data structure form. The insensitivity to the 2D object motion is useful to the applications such as object matching and spatial relation deduction.

The paper is organized as follows. Section 2 describes the line-based data model used to fit the 3D object geometry and the phase-one reconstruction process using the intersection operation. Section 3 presents the dynamic line resolution adjustment for the phase-two reconstruction based on the shape smoothness. For visualizing the 3D reconstructed object geometry a technique is given in Section 4 that converts the line-based geometric model to a bounded triangular mesh model. Section 5 gives the experimental results to illustrate the quality of the reconstruction results and the required processing time and memory space. Section 6 is the conclusions.

2. Reconstruction of the object line-based geometric model

The original (or starting) line-based data model used to fit the 3D object geometry is defined as a 2D array of line segments that have the same length and are perpendicular to a base plane at the regular grid points, as shown in Fig. 1. The uniform spacing between the grid points determines the spatial resolution of the line segments. Each line segment may contain more than one line section intersecting with the object. The reconstruction process consists of two phases. In the first phase the used line-based data model has a fixed line resolution. In the second phase the object line-based model obtained in the first phase is refined with a dynamic line resolution. The reconstruction of phase-one is described here and the phase-two reconstruction will be given in the next section.

The phase-one reconstruction algorithm:

1. Input: A set of silhouette images of the object obtained with a calibrated camera and a rotating turntable.
2. Output: The phase-one object line-based geometric model.
3. Method:
   (1) Generate a 2D array of line segments with a fixed line resolution and a uniform line length that based on the physical dimensions of the object to be reconstructed.
   (2) For each view
      (2.1) Project the 3D line segments to 2D image plane based on the camera calibration parameters.
      (2.2) For each projected line segment, calculate the 2D line sections that intersect with the object silhouette.
      (2.3) Back-project each found 2D line section to find the corresponding 3D line section onto the chosen line segment.
   (3) Find the intersection of the 3D line sections obtained from all views.

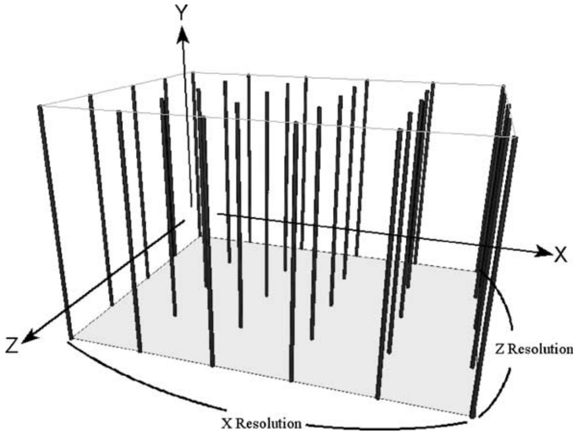


Fig. 1. The original line-based data model used to fit the 3D object geometry.

The calculations involved in the phase-one reconstruction are given below. For each view, the 3×4 projection matrix H is known. Using H, we can calculate the projected 2D line segment of each 3D line segment on the image plane. For a 3D line segment L_i with start point P_0 and end point P_1, the projection points p_0 and p_1 are given as:

$$p_0 = H \cdot P_0$$
$$p_1 = H \cdot P_1$$

By connecting p_0 and p_1, we can obtain the corresponding 2D line segment l_i.

We use the Bresenham algorithm (Bresenham, 1965) to represent l_i by the discrete integer coordinates stored in a linked list PL:

$$\mathrm{PL} = \{(u_0, v_0) \to (u_1, v_1) \to (u_2, v_2) \to \cdots \to (u_n, v_n)\}.$$

By tracing the coordinates of the line PL, we can obtain the 2D line sections that intersect with the binary object silhouette image, as shown in Fig. 2.

To obtain the corresponding 3D line sections intersecting the object, we need to back-project the 2D line sections to its original 3D line segment from which they are obtained. The back-projection diagram is illustrated in Fig. 3.

In Fig. 3 the 3D line segment L_i with the start point P_0 and the end point P_1 is projected to 2D line segment l_i from point p_0 to point p_1. Also, a general 3D point P on L_i is projected to the 2D point q on l_i. The lengths of L_i and l_i are D and d. The length from q to p_0 is l.

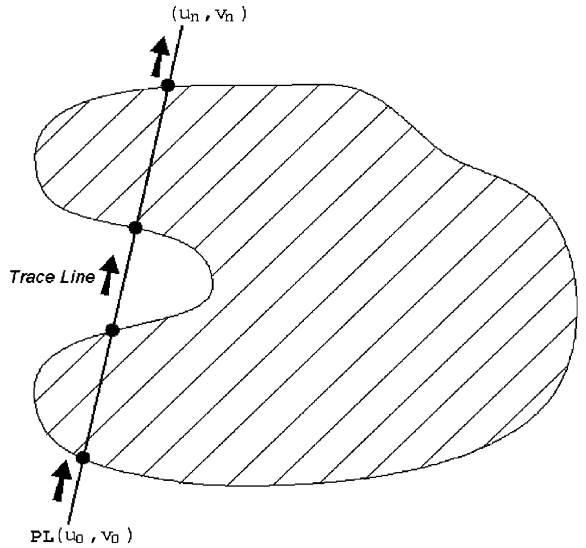


Fig. 2. Trace the line segment to find its line sections that intersect with the binary object silhouette image.

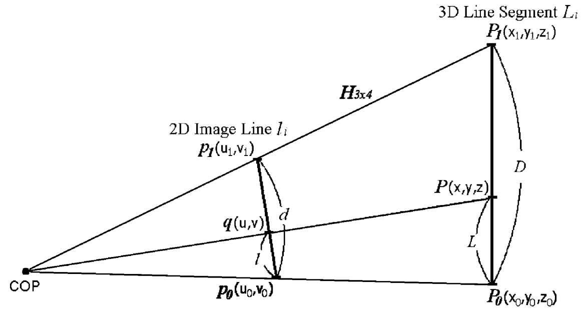


Fig. 3. Back-projection of a 2D point q on the image line to a 3D point P on the 3D line segment.

Given the projection matrix H and the coordinates of P_0, P_1, p_0, p_1, and q, we want to calculate the coordinate of the unknown 3D point P and the length L between P and P_0:

Let the line equations of L_i and l_i be given by

$$\begin{cases} x = x_0 + t(x_1 - x_0) \\ y = y_0 + t(y_1 - y_0) \\ z = z_0 + t(z_1 - z_0) \end{cases} \quad 0 \leqslant t = \frac{L}{D} \leqslant 1 \tag{1}$$

and

$$\begin{cases} u = u_0 + s(u_1 - u_0) \\ v = v_0 + s(v_1 - v_0) \end{cases} \quad 0 \leqslant s = \frac{l}{d} \leqslant 1 \tag{2}$$

where $(x, y, z)^{\mathrm{T}}$ is the coordinates of the point P, and t and L are unknown parameters.

From projection geometry, we have

$$w_j \begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} = H \begin{bmatrix} x_j \\ y_j \\ z_j \\ 1 \end{bmatrix}, \quad j = 0 \text{ and } 1 \tag{3}$$

and

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{4}$$

After some manipulations, we can obtain the following linear equations:

$$\begin{cases} u \cdot w + (w_0 u_0 - w_1 u_1) \cdot t = w_0 u_0 \\ v \cdot w + (w_0 v_0 - w_1 v_1) \cdot t = w_0 v_0 \\ 1 \cdot w + (w_0 - w_1) \cdot t = w_0 \end{cases} \tag{5}$$

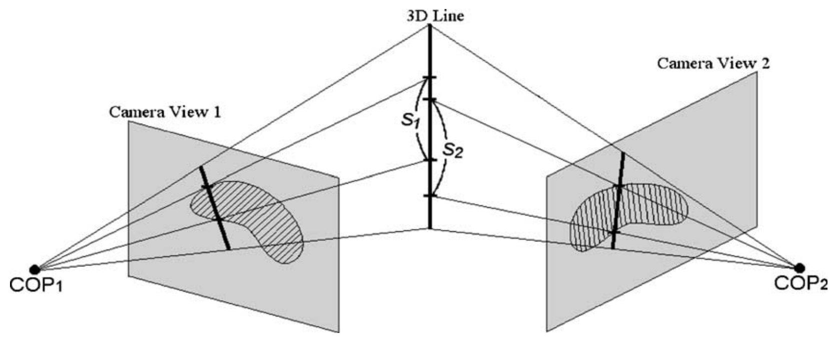


Fig. 4. Find the intersection of the line sections on a common line segment reconstructed from different views.

Then, we solve for t to obtain

$$t = \frac{w_0 s}{w_1 + s(w_0 - w_1)} = \frac{w_0 \cdot l}{w_1 \cdot d + (w_0 - w_1) \cdot l}.$$

Plugging the value of t into Eq. (1), we can calculate the coordinate of P and length L.

Using the above method, we can back-project the two endpoints of the 2D line section to the corresponding 3D line segment to locate the 3D line section intersecting the object silhouette in the view under consideration.

The final step of the reconstruction process is to intersect the 3D line sections obtained from all views, as depicted in Fig. 4.

3. Dynamic line resolution adjustment

The line segments in the phase-one reconstruction are uniformly distributed. If the line resolution is not high enough, it may miss some details of the object. For this reason, we have to check any two horizontally adjacent line segments for possible loss of the object details. If there is such a possibility, then a new line segment is inserted between the two original line segments (i.e., increase the line resolution locally) to capture the possible details of the object.

Two conditions for inserting a new line segment:

(1) No vertical overlap: The object we reconstruct is supposed to be an integral part, so it has no isolated components. Under this assumption, the line sections of any two horizontally adjacent line segments should have some degree of overlap in the vertical direction (i.e., y-direction). If no vertical overlap is found, the line may be near the object boundary or has a steep object slope. To obtain high shape accuracy, a new line segment is inserted to detect the finer object features. Fig. 5 shows an example of this case.

(2) A rapid change rate of vertical length: If the line sections of the two horizontally adjacent line segments are vertically overlapped, but the total lengths of these overlapping line sections have a large difference. Then it indicates that the object geometry may change dramatically between the two line segments. We will insert a new line segment to make the shape change smoothly.

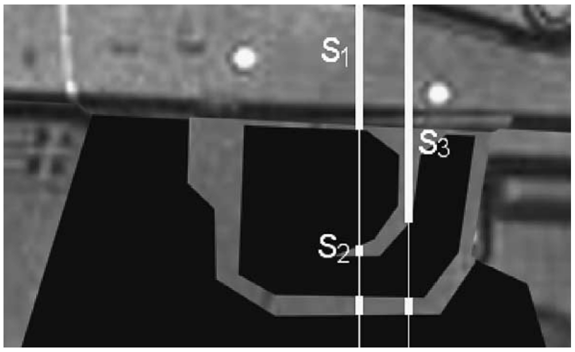


Fig. 5. The line sections S_2 and S_3 have no vertical overlap. A new line segment must be inserted in between.

To design the scheme for inserting the new line segments, we have to consider the 3D visualization of the reconstructed object at a later stage. As shall be seen, we need to divide the line segments into triplets for this object visualization. To do so, we proceed with the top view of the line-based model in which the line segments are projected to grid points, as shown in Fig. 6. We decompose each grid cell into two triangles along a diagonal line, then the line segments indicated by the three grid points of a triangle constitute a triplet we need. We should check each pair of line segments of the

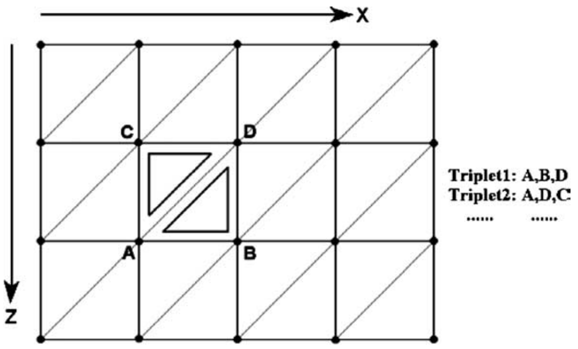


Fig. 6. Triplets of the line segments become grid points when seen from the top view of the line-based model.

triplets to see if they satisfy the two conditions for a new line insertion, as mentioned above.

To insert the new line segments we have to maintain the uniformity of the triangle-pair data structure depicted in Fig. 6, so we insert the new line segments as follows.

In Fig. 7, if L_i and L_j are two original line segments of a triplet that satisfy the condition of either having no vertical overlap or having a rapid length change rate, then a new line segment L_k needs to be inserted. Depending on the spatial relationship between L_i and L_j, there are three cases to be considered:

Case (a): L_i and L_j have a top-down spatial relationship.
As depicted in Fig. 7(a), the two grid cells on both sides of the triangle edge L_iL_j need to be equally subdivided into eight smaller cells. These cells form 16 new triangles.

Case (b): L_i and L_j have a left–right spatial relationship.
This is similar to case (a), as depicted in Fig. 7(b). Totally eight new line segments are inserted together with the line segment L_k.

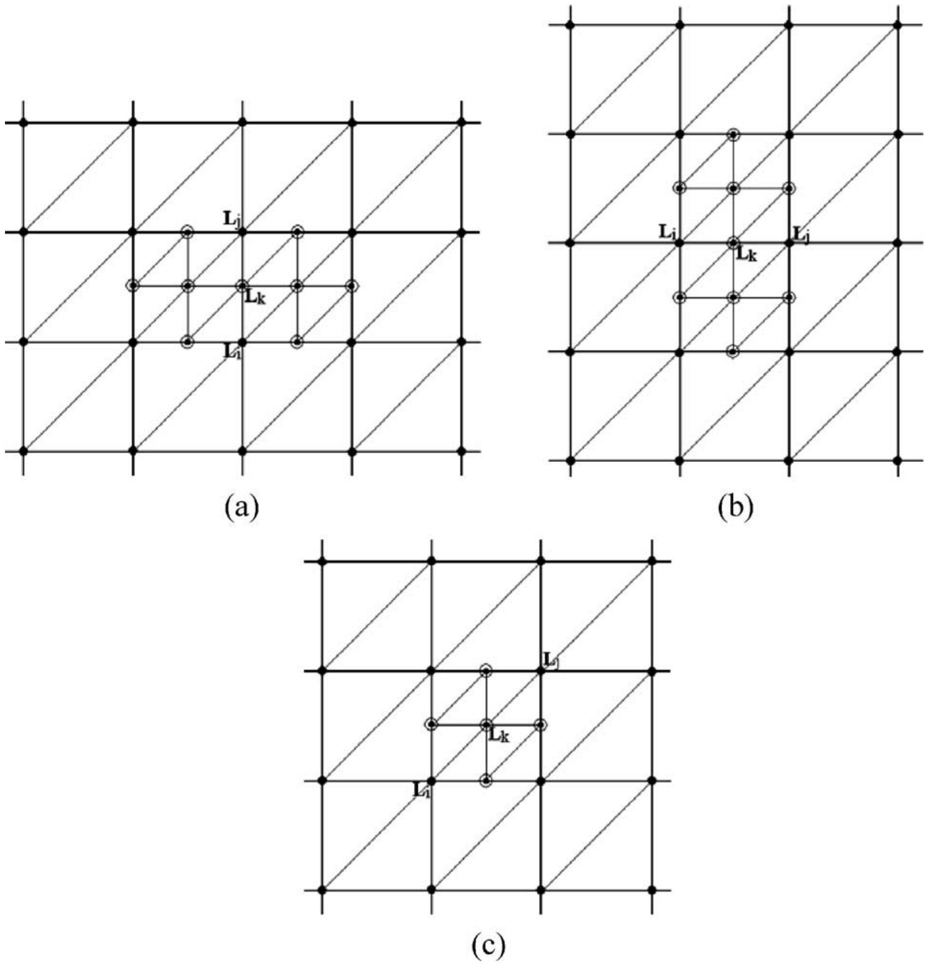


Fig. 7. (a)–(c) Three cases for inserting new line segments as seen from the top view of the line-based model. (Here $\odot$ indicates an inserted line segment.)

Case (c): L_i and L_j have a diagonal spatial relationship.
As depicted in Fig. 7(c), only the grid cell containing the diagonal edge L_iL_j is subdivided. Totally four new line segments are inserted together with L_k.

Using the above scheme, we can dynamically increase the local line resolution of the line-based model to reconstruct the details of the object, while maintaining the data structure uniformity.

The algorithm for dynamic line resolution adjustment is summarized as follows:

(1) Each pair of line segments in the original triplets is checked to see if a new line segment needs to be inserted.
(2) When either new line insertion condition is satisfied, the relevant grid cell(s) is subdivided and new line segments in triples are inserted, as described in the above three cases.
(3) The line segments of the newly inserted triplets are checked in pairs for the need of new line insertion.
(4) Steps (1)–(3) are repeated until a user-specified maximum subdivision level is reached or until no new line insertion is needed.

4. The bounded triangular mesh geometric model in the triangular mesh form

The object line-based geometric model obtained above is a collection of line sections that is obviously not bounded. An example is shown in Fig. 8. In order to visualize the 3D shape of the reconstructed object, this model needs to be converted to a bounded triangular mesh model.

The conversion involves two steps, as depicted in Fig. 9. In the first step the non-bounded line-based model is converted to a solid prism model with bounded surfaces. In the second step the solid prism model is converted to a bounded triangular mesh model.

4.1. Conversion of the line-based model to the solid prism model

As described in Section 3, the line segments in the line-based model are organized in triplets. To construct a prism from the line sections of the line segments in a triplet, as shown in Fig. 10, we need first to specify which line sections to use. If each line segment contains only one line section, then there is only one possibility to connect the three line sections, which leads to a unique prism.

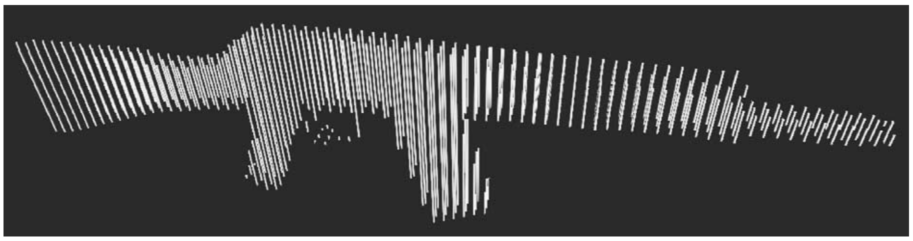


Fig. 8. A non-bounded line-based geometric model obtained from a reconstruction for a rifle.

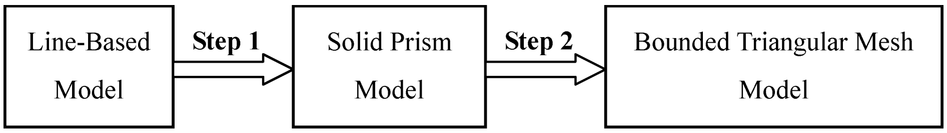


Fig. 9. The two-step model conversion.

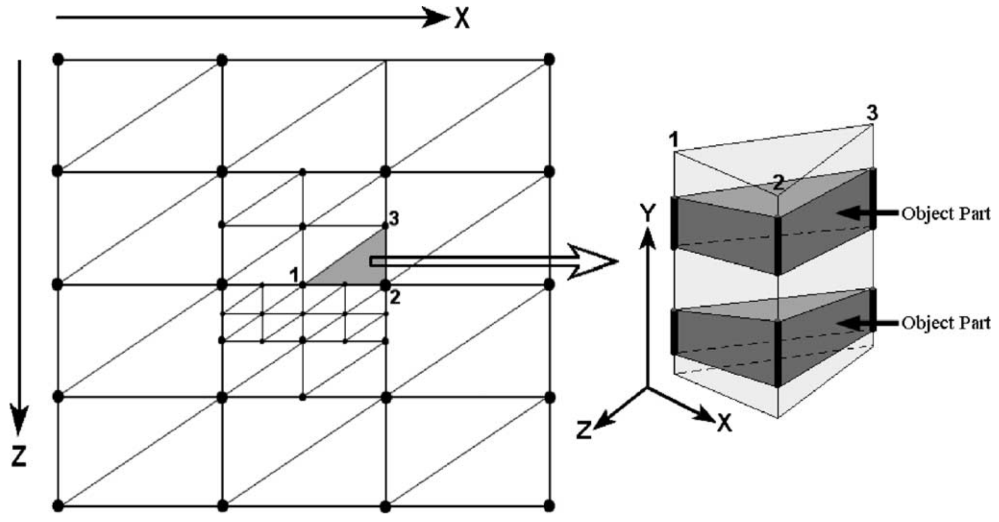


Fig. 10. The prisms at right are converted from the triplet of line segments 1, 2 and 3, shown in the top view of the line-based model at left.

However, when one or more line segments in the triplet are broken into line sections, then there is at least a portion of the line segment lying outside the object, called the open line section. The existence of open line section(s) means the object part under reconstruction contains a notch or a cut (see Figs. 11(c)–14(c)). Thus, all we need to do is to decide which open line sections define a cut. There are one to three open line sections in a cut. If two or three open line sections are involved in a cut, these sections are required to be vertically overlapped pairwise or through a transitive manner. Therefore, there are three possible ways to form a cut: (In the following in order to depict the mutual connections between line sections in constructing the cut or prisms in 2D space, we cut the constructed prisms in Figs. 11–14 along line segment 3 and, then, unfold the prism surfaces to make the three line segments lie on a 2D plane).

Case 1: A cut containing one open line section belonging to one line segment in the triplet. This is the case shown in Fig. 11. Without loss of generality, we assume line segment 2 has an open line section. To construct the cut and the prisms, we proceed as follows:

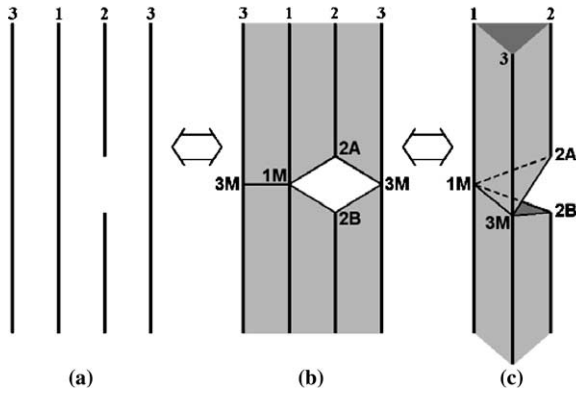


Fig. 11. A cut containing only one open line section of a line segment in the triplet.

1. Create one interior point 1M on line segment 1 that has an average height relative to the endpoints 2A and 2B, i.e. height of 1M = (height of 2A + height of 2B)/2, and then connect point 1M to points 2A and 2B.
2. Similarly, create another interior point 3M on line segment 3 having the same height as point 1M. Then connect point 3M to points 2A and 2B.
3. Finally, connect points 1M and 3M to get two triangles with vertex sets (1M,

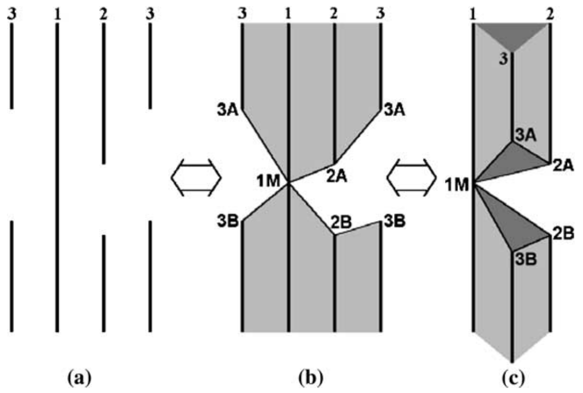


Fig. 12. A cut consists of two open line sections of two line segments in a triplet.

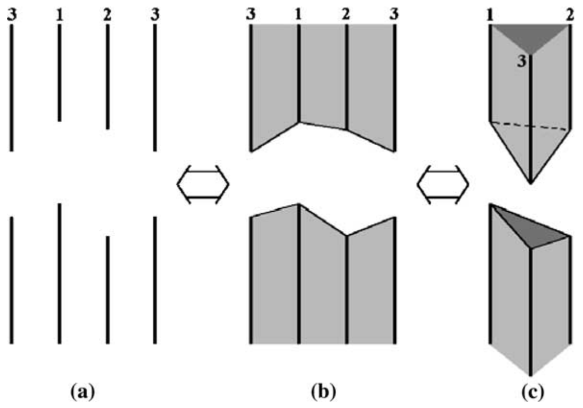


Fig. 13. A cut containing three open line sections belonging to three line segments, which are vertically overlapped pairwise.

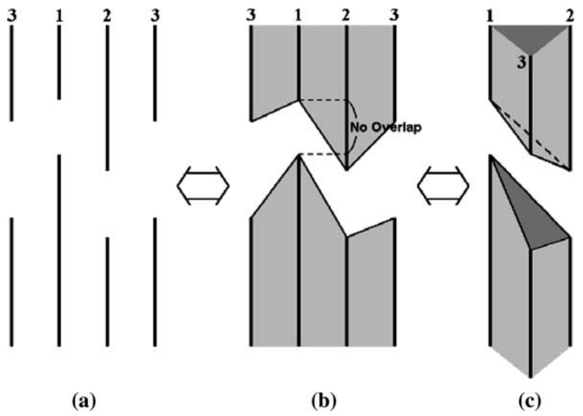


Fig. 14. A cut containing three open line sections belonging to three line segments, which are linked through the vertical overlapping relation in a transitive manner.

3M, 2A) and (1M, 3M, 2B), which form a cut as shown in Fig. 11(c).

Case 2: A cut containing two open line sections belonging to two line segments in the triplet, as shown in Fig. 12. We assume the two open line sections reside on line segments 2 and 3. To handle this case, the following steps are taken:

4. The corresponding endpoints of line sections of line segments 2 and 3 are connected to each other. Thus, points 2A and 3A are connected, so are endpoints 2B and 3B.
5. Create one interior point 1M having an average height with regard to points 2A, 2B, 3A and 3B.
6. Connect point 1M to points 2A and 3A and connect 1M to 2B and 3B. We get two triangles with vertex sets (1M, 2A, 3A) and (1M, 2B, 3B), which form a cut as depicted in Fig. 12(c).

Case 3: A cut containing three open line sections, one from each line segment in the triplet. There are two sub-cases shown in Figs. 13 and 14.

In Fig. 13 the three open line sections are vertically overlapped pairwise. Thus, it is intuitive to connect them together to form a cut.

In Fig. 14, the open line sections belonging to line segments 1 and 2 are not vertically overlapped. However, the open line sections belonging to line segments 2 and 3 are vertically overlapped, so are the open line sections belonging to line segments 3 and 1. That is, the open line sections belonging to line segments 1 and 2 are linked to the same cut in a transitive manner. Thus, we can connect the corresponding endpoints of the three open line sections to form a cut.

In this way, we can convert the non-bounded line-based model to a bounded prism model.

4.2. Conversion of the prism model to a bounded triangular mesh model

After the solid prism model is obtained, it is a simple matter to convert it into a bounded triangular mesh model. Each prism consists of the top and bottom triangular faces, and three quadrangles. If each quadrangle is cut along its diagonal to yield two triangles, then each prism can be converted to eight connected triangular polygons. By performing this conversion to each prism, we can construct the triangular mesh representation of the object. We can render the geometry of the object using any conventional graphic card. With the Z-buffer technique, the hidden surfaces can be removed and only the boundary surfaces are rendered.

5. Experimental results

We have implemented our line-based reconstruction method on a PC with an AMD Athlon 1.2 GHz CPU and 128 MB RAM. The 2D array dimension of the original line-based model for the phase-one reconstruction was chosen according to the physical size and details of the given object. The maximum subdivision level in the phase-two reconstruction is set to 2. For abbreviation, these model parameters are represented by $(M \times N, R)$, indicating the 2D array dimension is $M \times N$ and the maximum subdivision level in the dynamic line resolution scheme is R. If $R = 0$, the phase-two reconstruction is skipped and the model is reconstructed with a fixed line resolution.

We first applied our method to three different synthetic objects with an increasing geometric complexity: teapot, rifle, and flower. Thirty six views of each of these objects were taken with the object resting on a turntable that was rotated by 10° each time. In Fig. 15(a)–(c) three of the 36 views are shown. Notice the objects contain parts of varying detail. The qualities of the reconstructed objects with the listed model parameter settings are shown in Fig. 16(a)–(c). We can see the fine details of the objects were captured by our method, including teapot's flat handle, rifle's small trigger, and flower's thin petals, etc.

To evaluate the performance of the dynamic line resolution scheme, the reconstruction qualities of the dynamic and the fixed line resolution

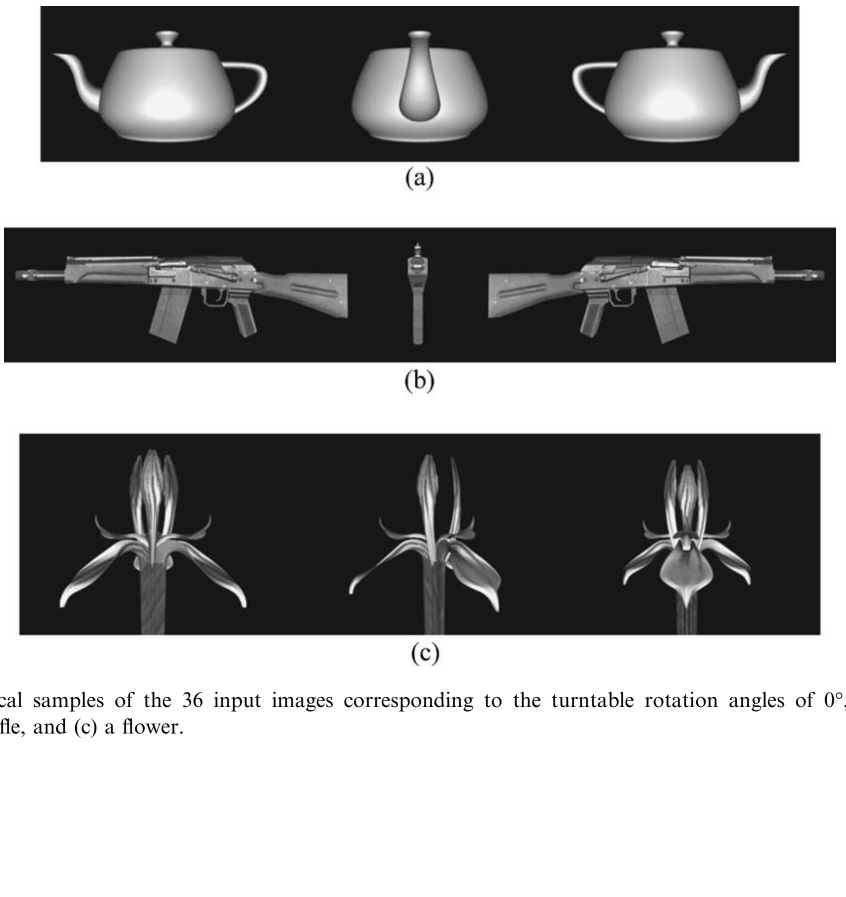
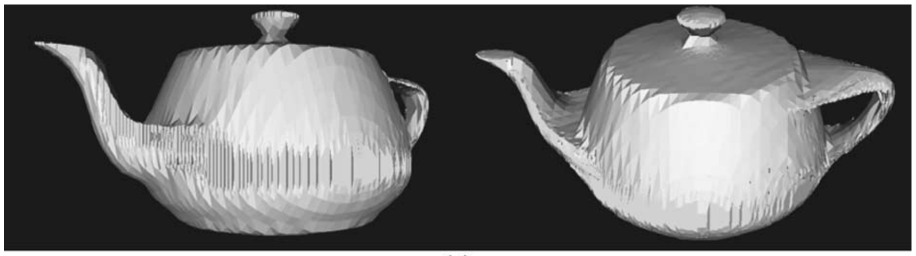
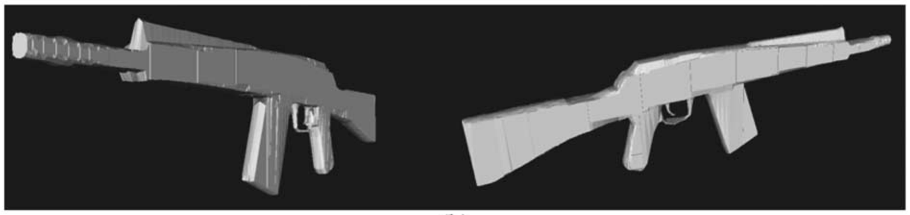


Fig. 15. Three typical samples of the 36 input images corresponding to the turntable rotation angles of 0°, 90°, and 180° for (a) a teapot, (b) a rifle, and (c) a flower.


(a)


(b)

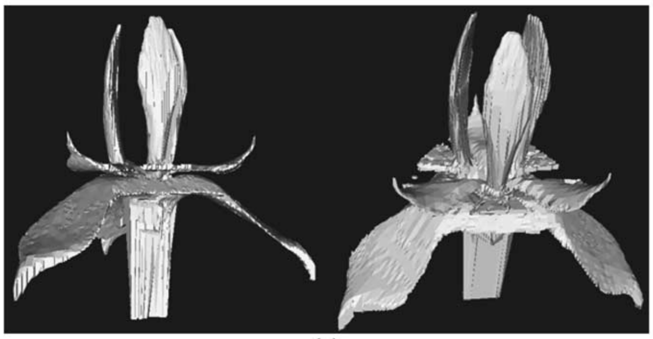

(c)

Fig. 16. Two views generated from the reconstructed triangular mesh models obtained for: (a) the teapot using the $(25 \times 16, 2)$ setting, (b) the rifle using the $(100 \times 8, 2)$ setting, and (c) the flower using the $(30 \times 30, 2)$ setting.

schemes are given in Fig. 17(a)–(c) and the processing time and memory space taken are tabulated in Table 2. From these comparisons, we can see the quality of reconstruction for the $(25 \times 16, 0)$ low fixed line resolution setting is not so good as that for the $(97 \times 61, 0)$ high fixed resolution setting, especially in the areas around the teapot handle and the lid knob. However, when changing the model parameter setting from $(25 \times 16, 0)$ to $(97 \times 61, 0)$, the processing time increases roughly by 10 times and the memory space increases by some 16 times. If using the dynamic line resolution scheme, the reconstruction quality for the $(25 \times 16, 2)$ setting is nearly as good as that for the $(97 \times 61, 0)$ setting, while the processing time taken is reduced to one third of that taken for the $(97 \times 61, 0)$ setting. This is because in the dynamic line resolution scheme the higher line resolution is only applied to the sparse object parts having high curvature or steep slope, while the lower line resolution to the major parts. Therefore, the dynamic line resolution scheme is a good compromise between the reconstruction quality and processing time, which is desirable in practical applications. (Note that we choose $(25 \times 16, 2)$ and $(97 \times 61, 0)$ for comparison. This is because the line spacing is the same in these two model parameter settings.)

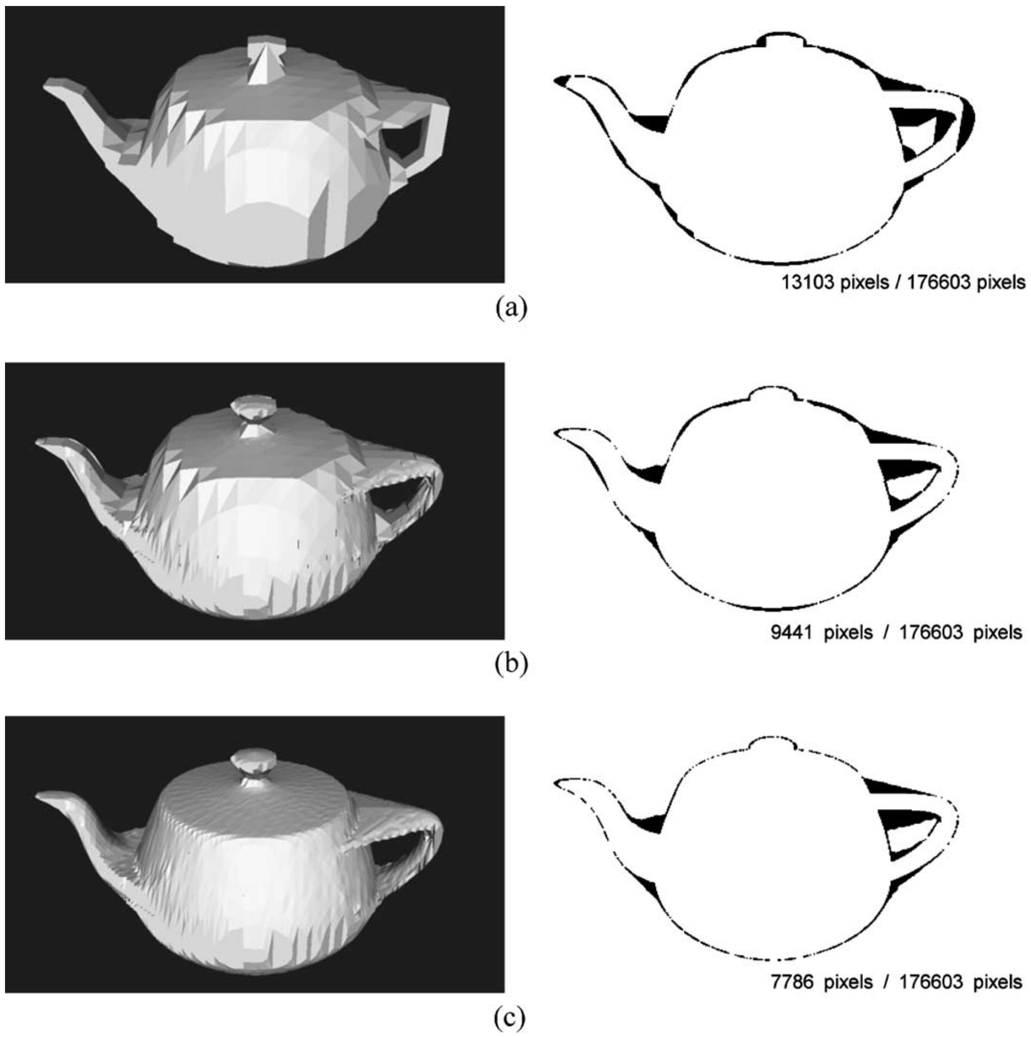

13103 pixels / 176603 pixels
(a)
9441 pixels / 176603 pixels
(b)
7786 pixels / 176603 pixels
(c)

Fig. 17. Comparison between reconstruction qualities of the teapots obtained with the three given model parameter settings: (a) $(25 \times 16, 0)$, (b) $(25 \times 16, 2)$, and (c) $(97 \times 61, 0)$. The left column shows a reconstructed view, and the right column shows the difference between the reconstructed view and the corresponding original view. The difference is marked by the black pixels whose number is given with respect to the total pixel number of the original view.

Table 2
Processing time and memory space for the synthetic object reconstruction experiments

	Processing time (s)			Memory space (Sections)		
	Low resolution	Dynamic resolution	High resolution	Low resolution	Dynamic resolution	High resolution
Teapot	$(25 \times 16, 0)$ 0.4	$(25 \times 16, 2)$ 2	$(97 \times 61, 0)$ 6	214	1078	3423
Rifle	$(100 \times 8, 0)$ 1	$(100 \times 8, 2)$ 4	$(397 \times 29, 0)$ 11	690	3019	9796
Flower	$(30 \times 30, 0)$ 2	$(30 \times 30, 2)$ 10	$(117 \times 117, 0)$ 21	477	4638	7709

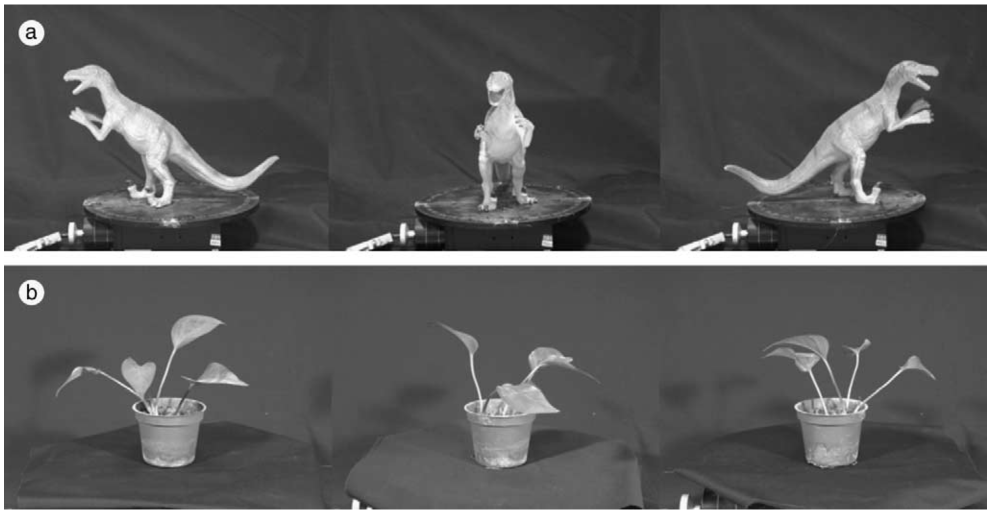


Fig. 18. Three of the 36 input images corresponding to the turntable rotation angles of 0°, 90°, and 180° for (a) a dinosaur and (b) a pot plant.

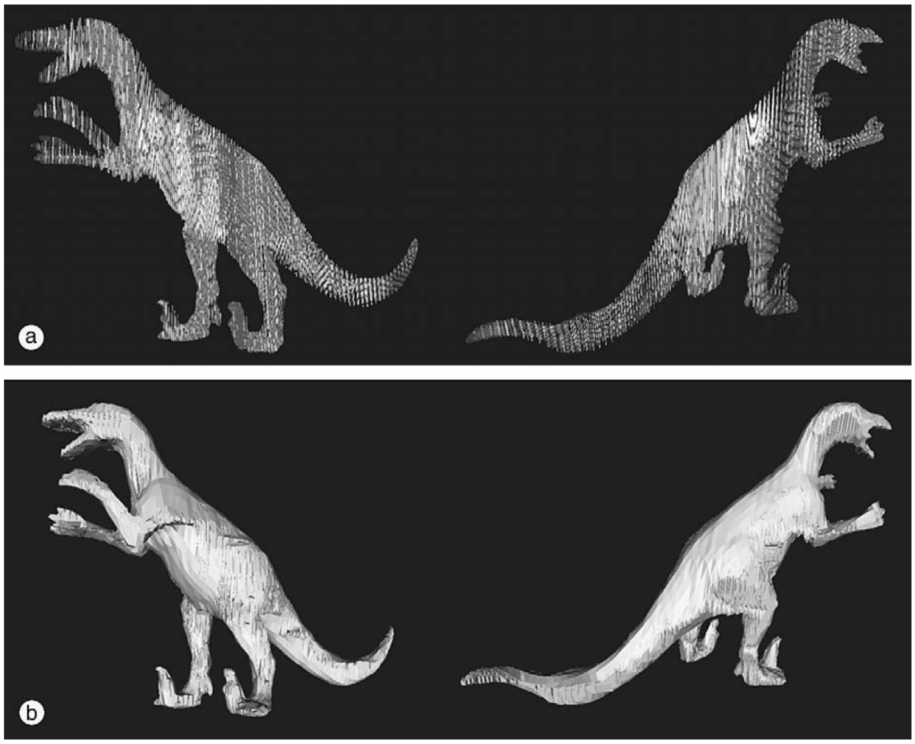


Fig. 19. Reconstruction results for the dinosaur using the $(81 \times 25, 2)$ model parameter setting. (a) Two views generated from the reconstructed line-based model. (b) Two views generated from the reconstructed triangular mesh model. In (a) the gray lines (lying mostly at the outer surface of the object) indicate the line segments inserted when using the dynamic line resolution scheme; the white lines (lying mostly inside the object) are the original line segments with a fixed resolution.

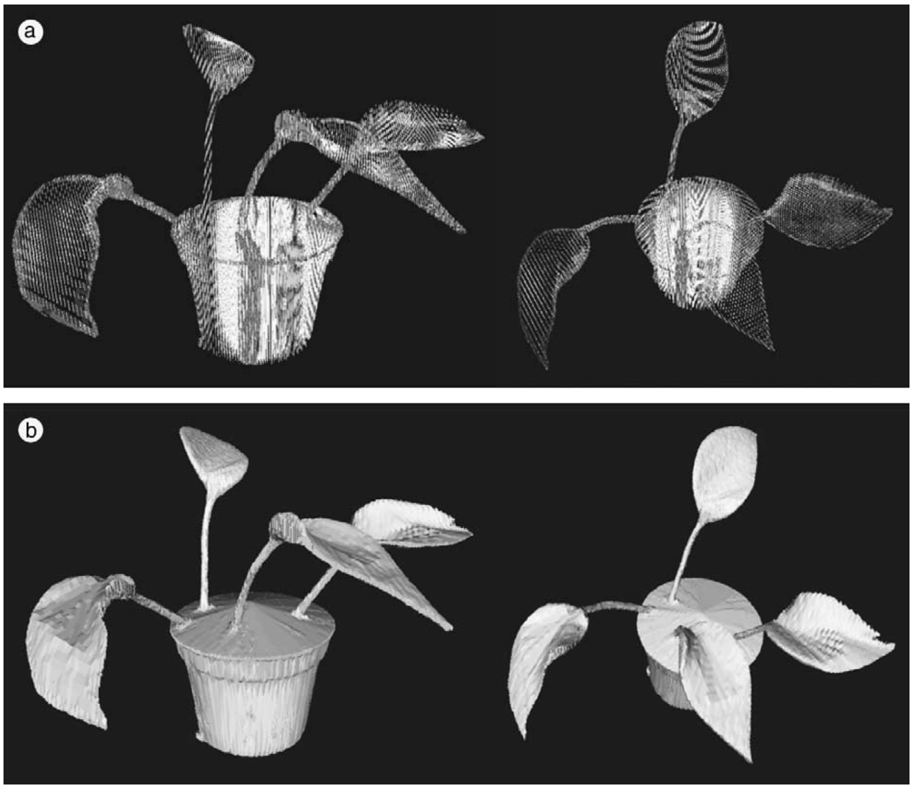


Fig. 20. Reconstruction results for the pot plant using the $(116 \times 106, 2)$ model parameter setting. (a) Two views generated from the reconstructed line-based model. (b) Two views generated from the reconstructed triangular mesh model.

Table 3
Processing time and memory space for the real object reconstruction experiments

	Processing time (s)		Memory space (Sections)	
	Dynamic resolution	Fixed resolution	Dynamic resolution	Fixed resolution
Dinosaur	$(81 \times 25, 2)$ 13	$(321 \times 97, 0)$ 53	6113	14177
Pot plant	$(116 \times 106, 2)$ 78	$(461 \times 421, 0)$ 547	13412	65736

We also apply our method to reconstruct the real objects. Figs. 18–20 and Table 3 show the reconstruction results and the time and space complexities for two real objects: dinosaur and pot plant. Here, again the turntable was rotated every 10° so that 36 images were taken for each object.

Generally speaking, the recovered shape of the object is in a fairly good agreement with the real shape except at the object spots where they are indented or too thin to be detected. These spots may be corrected using some post processing technique.

6. Conclusions

We have presented a rapid line-based method for constructing the 3D shape for complex objects. The method consists of two reconstruction phases; phase one is with a fixed line resolution and phase two with a dynamic line resolution. A conversion from the line-based geometric model to the bounded triangular mesh model is given to render the 3D reconstructed object shape. It has been shown in the experiments that it is capable of capturing most of the fine details of the object. The statistics show that the method works fast and requires a relatively low memory space compared to the octree-based reconstruction method. If needed, we can refine the reconstruction result by adding additional views in the reconstruction process. The geometric model thus obtained is adequate for use in object recognition and pose determination. And it is also suitable for browsing if the texture mapping is added. Currently we are planning to do the post-processing refinement of the geometric model. We shall use the geometric model obtained by the current method to enhance the stereo matching between the feature points visible on the object surfaces. Once the corresponding point pairs are found, we can compute their new 3D coordinates by the triangulation technique. Afterwards the surface patches are modified accordingly.

References

Bresenham, J.E., 1965. Algorithm for computer control of a digital plotter. IBM Syst. J., 25–30.

Chien, C.H., Aggarwal, J.K., 1986. Volume/surface octrees for the representation of three-dimensional objects. In: Proceedings of IPPR Conference on Computer Vision Graphics and Image Processing, Vol. 36, pp. 100–113.

Chia, T.L., Chen, Z., Yueh, C.J., 1996. Curved surface reconstruction using a simple structured light method. In: Proceedings of International Conference on Pattern Recognition, Vol. 1, pp. 844–848.

De Bonet, J.S., Viola, P., 1999. Roxels: responsibility weighted 3D volume reconstruction. In: Proceedings of International Conference on Computer Vision, Vol. 1, pp. 418–425.

Fromherz, T., Bichsel, M., 1994. Shape from contours as initial step in shape from multiple cues. In: ISPRS Commission III Symposium on Spatial Information from Digital Photogrammetry and Computer Vision, pp. 240–256.

Fua, P., 1997. From multiple stereo views to multiple 3D surfaces. Int. J. Comput. Vision 24 (1), 19–35.

Garcia, B., Brunet, P., 1998. 3D reconstruction with projective octrees and epipolar geometry. In: Proceedings of International Conference on Computer Vision, pp. 1067–1072.

Hu, G., Stockman, G., 1989. 3-D surface solution using structured light and constraint propagation. IEEE Trans. Pattern Anal. Machine Intell. 11 (4), 390–402.

Huang, T.S., Netravali, A.N., 1994. Motion and structure from feature correspondences: a review. Proc. IEEE 82 (2), 252–268.

Kutulakos, K., Seitz, S., 2000. A theory of shape by space carving. Int. J. Comput. Vision 38 (3), 199–218.

Moezzi, S., Katkere, A., Kuramura, D.Y., Jain, R., 1996. Reality modeling and visualization from multiple video sequences. IEEE Comput. Graph. Appl. 16 (6), 58–63.

Niem, W., 1999. Automatic reconstruction of 3D objects using a mobile camera. Image Vision Comput. 17 (2), 125–134.

Okutomi, M., Kanade, T., 1993. A multiple-baseline stereo. IEEE Trans. Pattern Anal. Machine Intell. 15 (4), 353–363.

Potmesil, M., 1987. Generating octrees models of 3D objects from their silhouettes in a sequence of images. In: Proceedings of IPPR Conference on Computer Vision Graphics and Image Processing, Vol. 40, pp. 1–29.

Prock, A., Dyer, C., 1998. Towards real-time voxel coloring. In: Proceedings of the DARPA Image Understanding Workshop, pp. 315–321.

Proesmans, M., Van Gool, L., Defoort, F., 1998. Reading between the lines—a method for extracting dynamic 3D with texture. In: Proceedings of International Conference on Computer Vision, pp. 1081–1086.

Seitz, S., Dyer, C., 1999. Photorealistic scene reconstruction by voxel coloring. Int. J. Comput. Vision 35 (2), 151–173.

Slabaugh, G., Culbertson, B., Malzbender, T., Schafer, R., 2001. A survey of methods for volumetric scene reconstruction from photographs. In: International Workshop on Volume Graphics.

Snow, D., Viola, P., Zabih, R., 2000. Exact voxel occupancy with graph cuts. In: Proceedings of International Conference on Computer Vision and Pattern Recognition, Vol. 3, pp. 345–352.

Srivastava, S.K., Ahuja, N., 1990. Octree generation from object silhouettes in perspective views. In: Proceedings of IPPR Conference on Computer Vision Graphics and Image Processing, Vol. 49, pp. 68–84.

Szeliski, R., 1993. Rapid octree construction from image sequences. In: Proceedings of IPPR Conference on Computer Vision Graphics and Image Processing: Image Understanding, Vol. 58 (1), pp. 23–32.

Szeliski, R., Golland, P., 1999. Stereo matching with transparency and matting. Int. J. Comput. Vision 32 (1), 45–61.