Visual Comput (2005)
DOI 10.1007/s00371-005-0281-7

ORIGINAL ARTICLE

Hong Wu
Yizhou Yu

Photogrammetric reconstruction of free-form objects with curvilinear structures

Published online: 12 May 2005

H. Wu
Dimentionality, LLC

Y. Yu
University of Illinois at Urbana-Champaign

Siebel Center for Computer Science, 201 North Goodwin Avenue, Urbana, IL 61801, USA
yyz@cs.uiuc.edu

This work was done while Yizhou Yu was visiting Dimentionality, LLC.

Abstract The shapes of many natural or man-made objects have curve features. The images of such curves usually do not have sufficient distinctive features to apply conventional feature-based reconstruction algorithms. In this paper, we introduce a photogrammetric method for recovering free-form objects with curvilinear structures. Our method chooses to obtain the topology and geometry of a sparse 3D wireframe of the object first instead of directly recovering a surface or volume model. Surface patches covering the object are then constructed to interpolate the curves in this wireframe while satisfying certain heuristics such as minimal bending energy. The result is an object surface model with curvilinear structures from a sparse set of images. We can produce realistic texture-mapped renderings of the object model from arbitrary viewpoints. Reconstruction results on multiple real objects are presented to demonstrate the effectiveness of our approach.

Keywords Image-based modeling · Bundle adjustment · Curve reconstruction · Thin-plate splines · Constrained Delaunay triangulation

1 Introduction

One of the main thrusts of research in computer graphics and vision is to study how to reconstruct the shapes of 3D objects from images and represent them efficiently. Nowadays, the techniques for reconstructing objects, which can be easily described by points and/or lines, have become relatively mature in computer vision, and the theory for representing curves and surfaces has also been well-developed in computer graphics. Nevertheless, reconstructing free-form natural or man-made objects still poses a significant challenge in both fields. One important subset of free-form objects have visually prominent curvilinear structures such as contours and curve features on surfaces. Intuitively, the two surface patches on different sides of a curvilinear feature should have a relatively large dihedral angle. More precisely, curvilinear features should have large maximum principal curvature. Because of this property, they are very often part of the silhouette of an object, and are very important in creating the correct occlusions between foreground and background objects as well as between different parts of the same object. As a result, unlike smooth free-form objects, the shape of an object with curvilinear features can be described fairly well by these features only. Such objects are ubiquitous in the real world, including natural objects, such as leaves and flower petals, as well as man-made objects, such as architecture and furniture, automobiles and electronic devices. Therefore, a robust method for reconstructing such objects would provide a powerful tool for digitizing natural scenes and man-made objects.

3D image-based reconstruction methods can be classified as either automatic or photogrammetric. Automatic reconstruction methods include structure-from-motion and 3D photography. Structure from motion (SFM) tries to recover camera motion, camera calibration and the 3D positions of simple primitives, such as points and lines, simultaneously via the well-established methods in multiple-view geometry [7, 10]. The recovered points and lines are unstructured and require a postprocessing stage for constructing surface models. On the other hand, 3D photography takes a small set of images with precalibrated camera poses, and is able to output surface or volume models directly. However, both methods typically require sufficient variations (texture or shading) on the surfaces to solve correspondences and achieve accurate reconstruction.

However, detecting feature points or curvilinear structures on free-form objects is often an error-prone process which prevents us from applying the automatic algorithms. Photogrammetric reconstruction, which allows the user to interactively mark features and their correspondences, comes handy at this point. Photogrammetric methods along with texture mapping techniques [5, 8, 14, 22] can effectively recover polyhedral models and simple curved surfaces, such as surfaces of revolution. A few commercial software packages [3, 18, 25] are available for photogrammetric reconstruction or image-based modeling and editing. Certain algorithmic details of the packages have not been made public. When the real object is a free-form object, even photogrammetric methods need a significant amount of effort to reach reasonable accuracy.

Our research aims to make the process of modeling free-form objects more accurate, more convenient and more robust. The reconstructed models should also exploit compact and smooth graphical surface representations that can be conveniently used for photorealistic rendering. To achieve these goals, we introduce a photogrammetric method for recovering free-form objects with curvilinear structures. To make this method practical for objects without sufficient color or shading variations, we define the topology and recover a sparse 3D wireframe of the object first instead of directly recovering a surface or volume model as in 3D photography. Surface patches covering the object are then constructed to interpolate the curves in this wireframe while satisfying certain heuristics such as minimal bending energy. The result is that we can reconstruct an object model with curvilinear structures from a sparse set of images and can produce realistic renderings of the object model from arbitrary viewpoints.

1.1 Background and related work

Photographs and range images have been the two major data sources for 3D object reconstruction.

Acquiring high quality smooth object shapes based on range images has been a central endeavor within computer graphics. The initial data from a range scanner is a 3D point cloud which can be connected to generate a polygon mesh. Researchers have been trying to fit smooth surfaces to point clouds or meshes [6, 11, 13]. While these surface fitting techniques can generate high quality object models, obtaining the point clouds using range scanners is not always effective since range scanners cannot capture the 3D information of shiny or translucent objects very accurately. Furthermore, obtaining dense point clouds for objects with curvilinear structures is not always necessary, either, if a sparse wireframe can describe the shape fairly well. On the other hand, taking images using a camera tends to be more convenient, and is not subject to the same restrictions as range scanners.

In computer vision, while multiple-view geometry of points, lines, and planes have been extensively studied and well-understood, recent studies have gradually turned to use curves and surfaces as basic geometric primitives for modeling and reconstructing 3D shapes. The difficulty in the reconstruction of curves is that the point correspondences between curves are not directly available from the images because there are no distinct features on curves except the endpoints. An algorithm was proposed in [28] to automatically match individual curves between images using both photometric and geometric information. The techniques introduced in [20] aimed to recover the motion and structure for arbitrary curves from monocular sequences of images. Reconstruction of curves from multiple views based on an affine shape method was studied in [1, 2]. The reconstruction of algebraic curves from multiple views has also been proposed by [12].

There has also been much work in computer vision on reconstructing smooth surface models directly from silhouettes and/or curve constraints. Each silhouette generates a visual cone that is tangential to the object surface everywhere on the silhouette. The object surface can be reconstructed as the envelope of its tangent planes from a continuous sequence of silhouettes [4, 9]. The problem with silhouettes is that they are not static surface features and tend to change according to a moving viewpoint. Thus, the camera poses must be obtained independent of the silhouettes. In addition, concave regions on the surface cannot be accurately recovered. In [30], this approach is further extended and the whole object surface is covered with triangular splines deformed to be tangential to the visual cones. The strength of the extended approach lies in representing smooth free-form objects that do not have high-curvature feature curves. In the event that such salient curves are present, a larger number of images would be necessary to capture both the position and surface normal changes across them. In comparison, by explicitly representing these feature curves, our method can reconstruct shapes from less images, and can represent both convex and concave features equally well. In

[33], a method is developed to reconstruct 3D surfaces from a set of unorganized range curves which may intersect with each other. It requires dense range curves as opposed to sparse salient curves.

A single view modeling approach was taken by [37] to reconstruct free-form surfaces. It solves a variational optimization to obtain a single thin-plate spline surface with internal curve constraints to represent depth as well as tangent discontinuities. The proposed technique is both efficient and user-friendly. Nevertheless, representing both foreground and background using a single spline surface is inadequate for most 3D applications where the reconstructed objects should have high visual quality from a large range of viewing directions.

2 Overview

In this section, we first provide an overview of our photogrammetric system from the point of view of the user, then we describe our model representation, and outline our curve and surface reconstruction algorithms. The reconstruction pipeline is shown in Fig. 1.

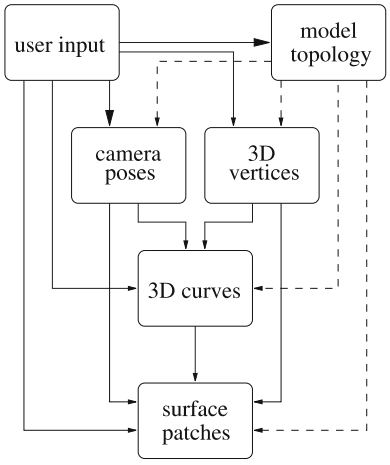

Fig. 1. Schematic of our photogrammetric reconstruction pipeline

2.1 The user's view

Constructing a geometric model of an object using our system is an incremental and straightforward process. Typically, the user selects a small number of photographs to begin with, and recovers the 3D geometry of the visible feature points and curves as well as the locations and orientations from which the photographs were taken. Eventually, 3D surface patches bounded by the recovered curves are estimated. These surface patches partially or completely cover the object surface. The user may refine

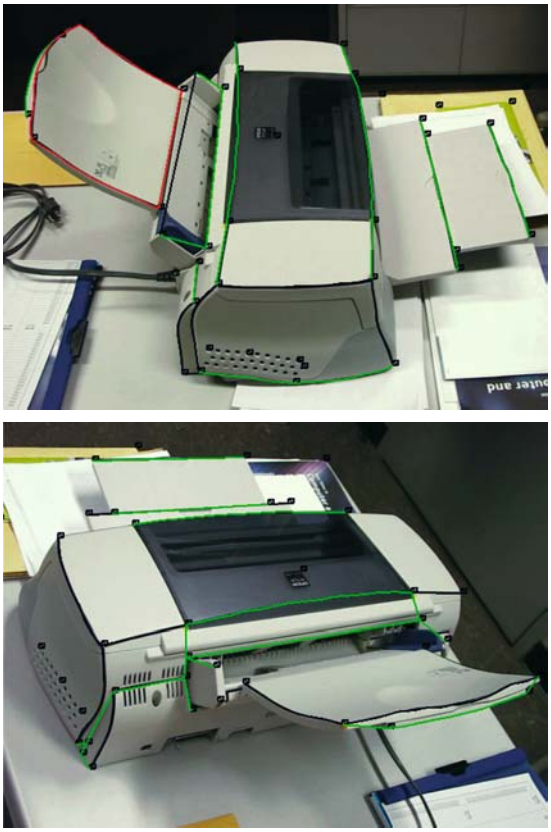

Fig. 2. During user interaction, three types of features—points, curves (*lines*) and regions—are marked. The points (*little squares*) and curves are originally drawn in black on the image planes. Their color changes to green once they are associated with correspondences. A region, shown in red, is marked by choosing a loop of curves

the model and include more images in the project until the model meets the desired level of detail.

There are two types of windows used in the reconstruction system: image viewers and model viewers. By default, there are two image viewers and one model viewer. The image viewers display two images of the same object at a time and can switch the displayed images when instructed. The user marks surface features, such as corners and curves, as well as their correspondences in the two windows (Fig. 2). Straight lines are considered as a special case of curves. The user marks point features in the images by point-and click; marks curve features by dragging the mouse cursor in the image plane with one of the buttons pressed. Features with and without associated correspondences are displayed in two distinct colors so that isolated features can be discovered easily. The user can also choose a sequence of curves to form the boundary of a region on the object surface. When the user concludes feature and region marking for the set of input images, the computer determines the 3D positions and shapes of the corners and curves that best fit the marked features in the images as well as the locations and orientations of the cameras. A 3D

surface patch that interpolates its boundary curves is also estimated for each marked image region.

The user can add new images to the initial set, and mark new features and correspondences to cover additional surface regions. The user can choose to perform an *incremental* reconstruction by computing the camera pose of a new image as well as the 3D information for the features associated with it. Alternatively, a full reconstruction can be launched to refine all the 3D points and curves as well as all the camera poses. An incremental reconstruction is less accurate and takes only a few seconds while a full reconstruction for reasonably complex models takes a few minutes. To let the user verify the accuracy of the recovered model and camera poses, the computer can reproject the model onto the original images (Fig. 5c,d). Typically, the projected model deviates from the user-marked features by less than a pixel.

Lastly, the user may generate novel views of the constructed object model by positioning a virtual camera at any desired location. Textures from the original images can be mapped onto the reconstructed model to improve its appearance.

2.2 Model representation and reconstruction

We represent the reconstructed 3D object model using boundary representations (B-reps) [16]. Such representations typically consist of three types of primitives: vertices, edges and faces. Edges can be either line segments or curve segments. Faces can be either planar polygons or curved surface patches that interpolate their respective boundary edges. For the same object, our system actually uses two boundary representations for different purposes: a compact and accurate representation with curves and curved surface patches for internal storage, and an approximate triangle mesh for model display and texture-mapping. The triangle mesh is obtained by discretizing the curves and surface patches into line segments and triangles, respectively.

Every boundary representation of an object implies two aspects: topological and geometric specifications. The topological specification involves the connectivity of the vertices and the adjacency of the faces while the geometric specification involves the actual 3D positions of the vertices and the 3D shapes of the curves and surface patches. The topological information can be obtained without knowing any specific geometric information. In our system, the topology of the reconstructed object evolves with user interactions. In the following, let us enumerate the types of user interaction and the corresponding topological changes they incur.

- *Marking a 2D point feature.* A 3D vertex is always created along with every new 2D point feature. The position of this 3D vertex is unknown at the beginning. Every 3D vertex maintains a list of its corresponding 2D points in the images. This list only has one member at first.
- *Marking the correspondence between two 2D points.* The two 3D vertices associated with the two point features are merged into one single vertex. The list of 2D points of the resulting vertex is the union of the original two lists.
- *Drawing a 2D curve.* In our system, an open 2D curve must connect two previously marked points while a closed curve must start and end at the same previously marked point. A 3D curve is also created along with every new 2D curve. The geometry of this 3D curve is unknown at this moment. However, the 3D curve automatically connects the two 3D vertices corresponding to the two endpoints of the 2D curve. Thus, a curved edge is created for the object. Every 3D curve maintains a list of its corresponding 2D curves in the images.
- *Marking the correspondence between two 2D curves.* The two 3D curves associated with the two 2D curves are merged into one single curve. The list of 2D curves of the resulting 3D curve is the union of the original two lists.
- *Marking a 2D region.* A 2D region is defined by a closed loop of 2D curves. When a 2D region is marked, a 3D surface patch is also created. The shape of this surface patch is unknown at this moment. The loop of 2D curves for the 2D region has a corresponding loop of 3D curves which define the boundary edges of the created 3D surface patch.

This topological evolution has two major advantages:

- *Correspondence propagation.* Once two vertices or curves merge, their corresponding 2D primitives are also merged into a single list. Thus, any two 2D primitives in the resulting list become corresponding to each other immediately without any user interaction.
- *Consistency check.* Marking correspondences is prone to errors. One important type of error is that two primitives belonging to the same image become corresponding to each other after correspondence propagation. This is not allowed because it implies that a 3D point can be projected to two different locations in the same image. This type of error can be easily detected through vertex or curve merging.

The geometric aspect of the object model is recovered automatically through 3D reconstruction algorithms which will be elaborated in the next few sections. A full reconstruction process consists of the following sequential steps (Fig. 1):

- The 3D positions of the vertices and all the camera poses are recovered once 2D point features and their correspondences have been marked;

– The 3D shapes of all the curves are obtained through a robust curve reconstruction algorithm (Fig. 5a,b);
– 3D thin-plate spline representations of the surface patches are obtained through a surface fitting algorithm (Fig. 5e);
– The curves and spline surface patches are further discretized to produce a triangle mesh for the object (Fig. 5f);
– Texture maps for the triangle mesh are generated from the original input images for synthetic rendering (Fig. 5g,h).

3 Camera pose and vertex recovery

In the first stage of geometric reconstruction, both camera poses and the 3D coordinates of the vertices are recovered simultaneously given user-marked point features and their correspondences. This is analogous to traditional structure-from-motion in computer vision. Therefore, we simply adapt classical computer vision techniques. Unlike structure-from-motion, we only have a sparse set of images while feature correspondences are provided by the user.

Camera poses involve both camera positions and orientations, which are also named external parameters. Besides these external parameters, a calibrated camera also has a set of known intrinsic properties, such as focal length, optical center, aspect ratio of the pixels, and the pattern of radial distortion. Camera calibration is a well-studied problem both in photogrammetry and computer vision; some successful methods include [32]. Although there are existing structure-from-motion techniques for uncalibrated cameras [10], we have found camera calibration to be a straightforward process and using calibrated cameras considerably simplifies the problem.

Given multiple input images with feature correspondences, we start the recovery process by looking for pairs of images with eight or more pairs of point correspondences. The point correspondences can be either user-specified or obtained through correspondence propagation. The relative pose between two cameras can be recovered from the linear algorithm presented in [15]. This algorithm requires that the points used are not coplanar. The major advantage of this algorithm is its linearity which is unlike nonlinear optimization that is likely to get stuck in local minima. Therefore, the user does not need to provide a good initialization through a user interface.

When the relative pose between two cameras has been computed, the system marks a connection between these two cameras. Once all the connections among the cameras have been created, we actually define a graph implicitly with the set of cameras as the nodes and the connections as the edges. The largest connected subgraph is chosen for reconstructing the geometry of the object. An arbitrary camera in this subgraph is chosen to be the base camera whose camera coordinate system also becomes the world coordinate system for the object. The absolute pose of any other camera in the subgraph can be obtained by concatenating the sequence of relative transformations along a path between that camera and the base camera. Once the camera poses have been obtained, the 3D positions of the vertices each of which has at least two associated 2D point features can be calculated by stereo triangulation.

The camera poses and vertex positions thus obtained are not extremely accurate. They serve as the initial solution for a subsequent nonlinear bundle adjustment [31]. Consider a point feature $\boldsymbol{x}$ in an image. Suppose it has an associated 3D vertex with position $\boldsymbol{X}$, the projection of $\boldsymbol{X}$ in the image should be made as close to $\boldsymbol{x}$ as possible. In bundle adjustment, this principle is applied to all marked image points while refining multiple camera poses and vertex positions simultaneously. We have achieved accurate reconstruction results with bundle adjustment.

4 Curve reconstruction

We reconstruct curves with the previously recovered camera poses and vertices. In the simplest situation, we have two corresponding image curves in two camera frames. For every pair of corresponding points on the image curves, a point on the 3D curve can be obtained by stereo triangulation. Therefore, the whole 3D curve can be reconstructed if the mapping between points on the two image curves can be obtained.

Let us first review the epipolar constraint before solving the mapping function. Suppose the relative rotation and translation between two camera frames are denoted as $\boldsymbol{R}$ and $\boldsymbol{T}$. The epipolar constraint between two corresponding points, $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ (in 2D homogeneous coordinates), in the respective two image planes can be formulated as

$$\boldsymbol{x_2}^T \hat{\boldsymbol{T}} \boldsymbol{R} \boldsymbol{x_1} = 0 \tag{1}$$

where $\hat{\boldsymbol{T}}$ is the skew symmetric matrix for $\boldsymbol{T}$ [19]. This epipolar constraint actually represents two distinct (epipolar) lines in the two image planes. If $\boldsymbol{x_1}$ is fixed and $\boldsymbol{x_2}$ is the variable in (1), it represents a line equation that the corresponding point of $\boldsymbol{x_1}$ in the second image should satisfy. Similarly, if we switch the role of $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ in (1), a line equation in the first image is defined. The distance between $\boldsymbol{x_2}$ and the epipolar line in the second image can be formulated as

$$D_2(\boldsymbol{x_1}, \boldsymbol{x_2}) = \frac{|\boldsymbol{x_2}^T \hat{\boldsymbol{T}} \boldsymbol{R} \boldsymbol{x_1}|}{||\hat{\boldsymbol{e}}_3 \hat{\boldsymbol{T}} \boldsymbol{R} \boldsymbol{x_1}||} \tag{2}$$

where $\boldsymbol{e_3} = [0,\ 0,\ 1]^T$, and

$$\hat{\boldsymbol{e}}_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

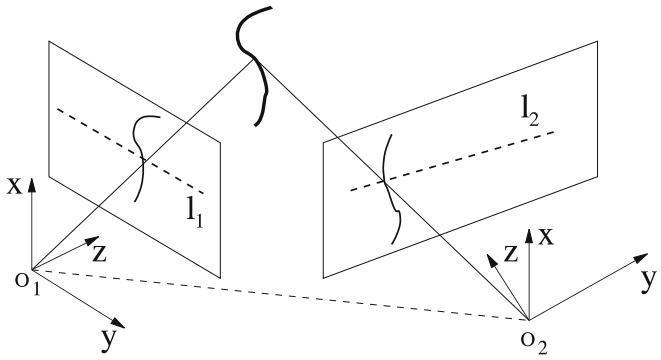

Fig. 3. The basic principle for obtaining point correspondences across image curves is based on the epipolar constraint. l_1 and l_2 are corresponding epipolar lines in two image planes. The intersections between the image curves and the epipolar lines correspond to each other

Similarly, the distance between $\boldsymbol{x_1}$ and the epipolar line in the first image can be formulated as

$$D_1(\boldsymbol{x_1}, \boldsymbol{x_2}) = \frac{|\boldsymbol{x_2}^T \hat{\boldsymbol{T}} \boldsymbol{R} \boldsymbol{x_1}|}{||\boldsymbol{x_2}^T \hat{\boldsymbol{T}} \boldsymbol{R} \hat{\boldsymbol{e}}_3^T||}. \tag{3}$$

Because of the epipolar constraint, solving the point mapping function between two image curves seems trivial at the first thought. For every point on the first curve, we can obtain an epipolar line in the second image. And the intersection between this line and the second curve is actually the corresponding point on the second curve (Fig. 3). However, this is true only when there is exactly one such intersection. In reality, uncertainties arise because of the shape of the curves and minor errors in the recovered camera poses (Fig. 4). There might be zero or multiple such intersections. In the worst case, the image curve is almost straight but parallel to the epipolar line to cause huge amount of uncertainty in the location of the intersection.

To obtain point correspondences between image curves robustly, we propose to compute one-to-one point mappings in an optimization framework. In general, reconstructions based on multiple views are more accurate than those based on two views because multiple views from various directions can help reduce the amount of uncertainty. Therefore, we discuss general multiple view curve reconstruction as follows. Note an image curve $\gamma(s)$ can be parameterized by a single variable $s \in [a, b]$. Consider the general case where there are m corresponding image curves, $\gamma_i(s_i)$, $0 \le i \le m-1$, each of which has a distinct parameter $s_i \in [a_i, b_i]$. Since we require that every curve connects two marked point features, the correspondences among the endpoints of these m curves are known. Without loss of generality, we choose γ_0 as the base curve and assume that $\gamma_0(a_0)$ corresponds to $\gamma_i(a_i)$, $1 \le i \le m-1$. Thus, obtaining point correspondences among these m curves is equivalent to solving $m-1$ mappings, $\sigma_i(s_0)$, $1 \le i \le m-1$, each of which is a continuous and monotonically increasing function that maps $[a_0, b_0]$ to $[a_i, b_i]$. In terms of closed image curves, as long as there are at least two point features on each of them and the respective point features correspond to one another, each closed curve can be broken into two or more open curves. The mapping functions for closed curves can then be represented as in open curve cases.

Since these curves lie in m different image planes, the relative rotations and translations between the ith camera frame and the jth camera frame is respectively denoted as $\boldsymbol{R_{ij}}$ and $\boldsymbol{T_{ij}}$, $0 \le i, j \le m-1$. The epipolar constraint between corresponding points on the ith and the jth curves requires that

$$\gamma_j(\sigma_j(s_0))^T \hat{\boldsymbol{T}}_{\boldsymbol{ij}} \boldsymbol{R_{ij}} \gamma_i(\sigma_i(s_0)) = 0, s_0 \in [a_0, b_0]. \tag{4}$$

Thus, the desired mappings should be the solution of the following minimization problem,

$$\min_{\sigma_i, 1 \le i \le m-1} \sum_{ij, i<j} \int_{a_0}^{b_0} (\gamma_j(\sigma_j(s)))^T \hat{\boldsymbol{T}}_{\boldsymbol{ij}} \boldsymbol{R_{ij}} \gamma_i(\sigma_i(s)))^2 ds. \tag{5}$$

As in bundle adjustment, it is more desirable to minimize projection errors in the image planes directly. In an image plane, satisfying the epipolar constraint is equivalent to minimizing distances similar to those given in (2) and (3). Furthermore, to guarantee that $\sigma(s)$ is a monotonically increasing one-to-one mapping, $\sigma(s) \le \sigma(s')$ must be

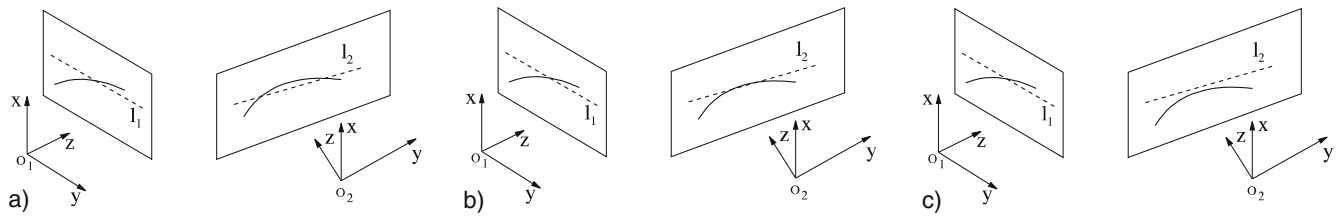

Fig. 4a–c. Uncertainties may arise when solving point correspondences across image curves. (**a**) There might be multiple intersections between the curve and the epipolar line. (**b**) The epipolar line might be tangential to the curve (in the image on the right). There is a huge amount of uncertainty in the location of the intersection if the curve is locally flat. (**c**) There might be no intersections between the curve and the epipolar line (in the image on the right) due to minor errors in camera calibration

held for arbitrary $s \in [a, b]$ and $s' \in [a, b]$ such that $s < s'$. To incorporate these considerations, the above minimization problem should be reformulated as

$$\min_{\sigma_i, 1 \le i \le m-1} \sum_{ij, i<j} \int_{a_0}^{b_0} \left(\frac{(\gamma_j(\sigma_j(s))^T \hat{\boldsymbol{T}}_{\boldsymbol{ij}} \boldsymbol{R}_{\boldsymbol{ij}} \gamma_i(\sigma_i(s)))^2}{||\hat{\boldsymbol{e}}_{\boldsymbol{3}} \hat{\boldsymbol{T}}_{\boldsymbol{ij}} \boldsymbol{R}_{\boldsymbol{ij}} \gamma_i(\sigma_i(s))||^2} + \frac{(\gamma_j(\sigma_j(s))^T \hat{\boldsymbol{T}}_{\boldsymbol{ij}} \boldsymbol{R}_{\boldsymbol{ij}} \gamma_i(\sigma_i(s)))^2}{||\gamma_j(\sigma_j(s))^T \hat{\boldsymbol{T}}_{\boldsymbol{ij}} \boldsymbol{R}_{\boldsymbol{ij}} \hat{\boldsymbol{e}}_{\boldsymbol{3}}^T||^2} \right) ds + \lambda \sum_i \int_{a_0}^{b_0} \int_{s}^{b_0} \max^2(\sigma_i(s) - \sigma_i(s'), 0) ds' \, ds \qquad (6)$$

where the first term addresses the epipolar constraints, the second term enforces that $\sigma_i(s)$ is a one-to-one mapping, and λ indicates the relative importance between these two terms. The second term vanishes when $\sigma_i(s)$ is actually a one-to-one mapping no matter how large λ is. Therefore, we have found that λ can be set to a large value such as 10^3 without biasing the final solution.

There are practical issues concerning the above minimization. First, before numerical optimization methods can be applied, the integrals should be replaced by summations since each user-marked image curve is actually a discrete set of pixels. A continuous image curve with subpixel accuracy is defined to be the piecewise linear curve interpolating this set of pixels. Given m corresponding image curves, $\gamma_i(s_i), 0 \le i \le m-1$, to achieve a high precision, we discretize their corresponding 3D curve using the number of pixels on the longest image curve which is always denoted as $\gamma_0(s_0)$. This scheme basically considers the longest image curve as the 2D parameterization of the 3D curve and there is a depth value associated with each pixel on the longest image curve. Each mapping $\sigma_i(s)$ is thus also a discrete function with the same number of entries as the number of pixels on $\gamma_0(s_0)$. Given a pixel on $\gamma_0(s_0)$, its corresponding points on other shorter image curves may have subpixel locations. Both the quasi-Newton and conjugate gradient [21] methods can then effectively minimize the discretized cost function. The number of discrete points on each curve is fixed throughout the optimization.

Second, a reasonably good initialization is required to obtain an accurate solution from a nonlinear formulation. In practice, we parameterize the image curves using their arc lengths. For the mapping functions we seek, the linear mapping between two parameter intervals is one of the possible initializations. But we actually initialize the mappings using dynamic programming which is particularly suitable for order-preserving one-dimensional mappings. We initialize each $\sigma_i(s)$ independently using only two curves (γ_0 and γ_i) and adopt the discrete version of the first term in (6) as the cost function for dynamic programming while enforcing one-to-one mapping as a hard constraint which means only order-preserving mappings are admissible. Specifically, we represent each curve γ_i as a discrete set of pixels, $\boldsymbol{p}_i^k, 0 \le k \le n_i$, where n_i is the number of pixels on the curve. Dynamic programming recursively computes the overall mapping cost. The cumulative cost between a pair of pixels on the two curves is defined as

$$C_{dp}(\boldsymbol{p}_0^k, \boldsymbol{p}_i^l) = D(\boldsymbol{p}_0^k, \boldsymbol{p}_i^l) + \min_{r \in S_{kl}} C_{dp}(\boldsymbol{p}_0^{k-1}, \boldsymbol{p}_i^r) \qquad (7)$$

where $D(\boldsymbol{p}_0^k, \boldsymbol{p}_i^l) = D_1(\boldsymbol{p}_0^k, \boldsymbol{p}_i^l) + D_2(\boldsymbol{p}_0^k, \boldsymbol{p}_i^l)$, and S_{kl} contains all admissible values of r under the condition that $\boldsymbol{p}_0^k$ matches $\boldsymbol{p}_i^l$.

Once we have obtained all the mapping functions, for every discrete value of $s_0 \in [a_0, b_0]$, there is a set of corresponding image points, $\gamma_i(\sigma_i(s_0)), 0 \le i \le m-1$. The point on the 3D curve corresponding to this list of 2D points can be obtained using bundle adjustment. At the end, all the 3D points recovered in this way form the reconstruction of the 3D curve. This reconstructed 3D curve is essentially unparameterized.

When smooth curves are desirable, we actually perform a novel and efficient bundle adjustment to directly fit a smooth 3D curve to a set of corresponding image curves. The smooth 3D curve can be either a spline curve or a subdivision curve. Both types of curves are controlled by a sparse set of control vertices. We only consider the sparse set of 3D control vertices $\boldsymbol{X}_l^c, l = 0, 1, \ldots, M$, as the unknowns during optimization to improve performance. A smooth curve can always be generated from this set of control vertices. A dense set of points sampling the generated curve are denoted as, $\boldsymbol{X}_k^s, k = 0, 1, \ldots, N$, where N equals the number of pixels on the longest image curve $\gamma_0(s_0)$ again. A sample point $\boldsymbol{X}_k^s$ can be projected into the m image planes to obtain m projected 2D points $\boldsymbol{x}_{kj}^p$, $j = 0, 1, \ldots m$. We can see that the locations of these projected 2D points are indirectly determined by the sparse set of 3D control vertices. Ideally, $\boldsymbol{x}_{kj}^p$ should lie on the image curve γ_j. In practice, there is likely to be a nonzero distance between the projected point and the image curve. We would like to minimize this type of distance by searching for the optimal 3D positions of the control vertices. In summary, we would like to solve the following minimization problem,

$$\min_{\boldsymbol{X}_l^c, 0 \le l \le M} \sum_{k=0}^{N} \sum_{j=0}^{m-1} \text{dist}(\boldsymbol{x}_{kj}^p, \gamma_j) \qquad (8)$$

where $\text{dist}(\boldsymbol{x}, \gamma)$ represents the minimum distance between a point and a curve. In practice, we adopted a type of interpolative subdivision curve [38] and have obtained accurate and efficient 3D curve reconstruction by solving the minimization problem in (8) using the conjugate gradient method. In our current implementation, the number of control vertices for each curve is fixed during optimiza-

tion. It is at least one order of magnitude smaller than the number of pixels on the image curves.

5 Surface reconstruction

In the reconstruction system, every surface patch is defined by a closed loop of 2D boundary curves. The boundary curves need to be marked in the same image, and they enclose a 2D image region which we actually adopt as the parameterization for the target surface patch. Because of this parameterization, the surface patch is a depth function defined on the image plane in the local camera coordinate system. Therefore, recovering the surface patch has been reduced to estimating a depth value at every pixel inside the closed image region. The estimated surface patch can be represented in the world coordinate system by simply applying the transformation between the camera's local frame and the world frame.

There are two different choices for estimating the depth function in the local camera frame. If the original object surface has rich texture, but is not highly reflective or translucent (as the object in Fig. 7), the first option would try to estimate a dense depth field using a version of the stereo reconstruction algorithm [26] that is based on anisotropic diffusion of the depth values. It imposes a regularization term to guarantee depth smoothness and at the same time preserves depth discontinuities. Such an algorithm requires that there is at least another image of the same surface region. Since the depth on the boundary curves have already been recovered, these known depths serve as a boundary condition for the regularization term. The algorithm in [26] can be easily extended to incorporate more than two views of the surface.

On the other hand, if the original object surface has very sparse point features or no features at all, estimating a dense depth field becomes infeasible. In this case, we choose to simply fit a thin plate spline (TPS) surface to the boundary depth values as well as the depths at the sparse set of interior features if there are any. Since the thin plate spline model minimizes a type of bending energy, it is smooth and would not generate undesirable effects in featureless regions. We only use one single view for TPS fitting. In practice, our system chooses the image with the most frontal-facing view of the surface region. The reason that we only need one single view for TPS fitting is related to the type of objects we choose to focus on in this paper. As mentioned in Sect. 1, the feature curves are responsible for creating the correct occlusions between foreground and background objects as well as between different parts of the same object. Therefore, the visual shape of an object is captured very well by these curves. The surface patches in between these curves only need to be reconstructed to a less degree of accuracy. Necessary conditions for avoiding visual artifacts and inconsistencies are that the surface patches should interpolate their boundary curves and should be smooth without obviously extruding vertices because extruding vertices modify the occluding contours and silhouettes of the object and can be noticeable.

The thin plate spline model is commonly used for scattered data interpolation and flexible coordinate transformations [17, 23, 34]. It is the 2D generalization of the cubic spline. Let v_i denote the target function values at corresponding locations $\boldsymbol{x_i}$ in an image plane, with $i = 1, 2, \ldots, n$, and $\boldsymbol{x_i}$ in homogeneous coordinates, $(x_i, y_i, 1)$. In particular, we will set v_i equal to the depth value at $\boldsymbol{x_i}$ to obtain a smooth surface parameterized on the image plane. We assume that the locations $\boldsymbol{x_i}$ are all different and are not collinear. The TPS interpolant $f(x, y)$ minimizes the bending energy

$$I_f = \iint f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2 \, dxdy \tag{9}$$

and has the form:

$$f(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x} + \sum_{i=1}^{n} w_i U(||\boldsymbol{x_i} - \boldsymbol{x}||) \tag{10}$$

where $\boldsymbol{a}$ is a coefficient vector and w_i represents the weights of the basis function $U(r)$, which is defined as follows: $U(r) = r^2 \log r$ for $r \neq 0$; $U(0) = 0$. In order for $f(\boldsymbol{x})$ to have square integrable second derivatives, we require that

$$\sum_{i=1}^{n} w_i \boldsymbol{x_i} = 0 \, . \tag{11}$$

Together with the interpolation conditions, $f(\boldsymbol{x_i}) = v_i$, this yields a linear system for the TPS coefficients:

$$\left(\begin{array}{c|c} \boldsymbol{K} & \boldsymbol{P} \\ \hline \boldsymbol{P}^T & \boldsymbol{0} \end{array}\right)\left(\frac{\boldsymbol{w}}{\boldsymbol{a}}\right) = \left(\frac{\boldsymbol{v}}{\boldsymbol{0}}\right) \tag{12}$$

where $K_{ij} = U(||\boldsymbol{x_i} - \boldsymbol{x_j}||)$, the ith row of $\boldsymbol{P}$ is $\boldsymbol{x_i}^T$, $\boldsymbol{w}$ and $\boldsymbol{v}$ are column vectors formed from w_i and v_i, respectively, and $\boldsymbol{a}$ is the coefficient vector in (10). We will denote the $(n+3) \times (n+3)$ matrix of this system by $\boldsymbol{L}$. As discussed, e.g., in [23], $\boldsymbol{L}$ is nonsingular and we can find the solution by inverting $\boldsymbol{L}$. If we denote the upper left $n \times n$ block of $\boldsymbol{L}^{-1}$ by $\boldsymbol{A}$, then it can be shown that $I_f \propto \boldsymbol{v}^T\boldsymbol{A}\boldsymbol{v} = \boldsymbol{w}^T\boldsymbol{K}\boldsymbol{w}$.

When there is noise in the specified values v_i, one may wish to relax the exact interpolation requirement by means of regularization. This is accomplished by minimizing

$$E(f) = \sum_i (v_i - f(\boldsymbol{x_i}))^2 + \beta I_f. \tag{13}$$

The regularization parameter β, a positive scalar, controls the amount of smoothing; the limiting case of $\beta = 0$ reduces to exact interpolation. As demonstrated in [34], we can solve for the TPS coefficients in the regularized case

by replacing the matrix $\boldsymbol{K}$ by $\boldsymbol{K} + \beta \boldsymbol{I}$, where $\boldsymbol{I}$ is the $n \times n$ identity matrix.

6 Mesh construction and texture mapping

We actually obtain a triangle mesh for texture mapping by discretizing the estimated surface patches. To avoid T-junctions in the resulting mesh, we require that two adjacent surface patches sharing the same curve should be discretized such that the two sets of triangles from the two patches have the same set of vertices on the curve. We satisfy this requirement by discretizing the curves first. Given an error threshold, each curve is approximated by a polyline such that the maximum distance between the polyline and the original curve is below the threshold. Thus, the boundary of a surface patch becomes a closed polyline. Since each surface patch has a marked region as its parameterization in one of the input images, the 3D boundary polyline of a patch is reprojected onto that image to become a boundary polyline for the marked region. A constrained Delaunay triangulation (CDT) is then constructed to triangulate the image region while keeping its boundary polyline. This planar triangulation is elevated using the surface depth information to produce the final triangulation for the 3D surface patch.

We use texture-mapping to generate synthetic images of the reconstructed models. Since texture-mapping is not the focus of this paper, we simply apply previously developed image-based texture-mapping techniques [5, 29, 36]. The basic idea is to backproject some of the images onto each surface patch of a reconstructed model.

7 Time complexity analysis

Let us analyze the time complexity of the 3D reconstruction algorithms in this paper. There are multiple components for the curve reconstruction algorithms in Sect. 4. Let N_p be the total number of pixels on the image curves. The initialization step using dynamic programming takes $O(N_p)$ time since dynamic programming has linear time complexity. The nonlinear optimization following dynamic programming needs multiple iterations to converge. During each iteration, we need to calculate the gradient of the cost function. This gradient computation is linear in terms of N_p. Let n_i be the maximum number of iterations necessary for reconstructing a curve. The complexity of the nonlinear optimization is $O(n_i N_p)$. Nonlinear optimization using sparse control vertices tends to be more efficient because it has the same complexity for gradient computation, but has fewer variables and thus usually requires less iterations.

For the thin-plate spline (TPS) fitting in Sect. 5, we need to solve a linear system for each surface region. Let n_r be the number of surface regions, and n_c be the maximum number of constraints in a region. Solving a linear system needs at most $O(n_c^3)$ time. Therefore, the total complexity for TPS fitting is $O(n_r n_c^3)$. From the reconstructed curves, we can obtain depth values at every pixel on the boundary of a surface region. However, since we produce a triangle mesh at the end, we only need to set constraints at the vertices of the boundary polylines. At the interior of a region, we use the depth values at a very sparse set of feature points as well. Thus, n_c is typically very small, less than 50. Solving such small linear systems does not take much time.

At the end, we need to run constrained Delaunay triangulation (CDT) to obtain the final meshes. Let n_v be the maximum number of vertices for a surface region. The complexity of CDT is $O(n_v \log n_v)$. The total complexity of mesh construction is $O(n_r n_v \log n_v)$. Again, mesh vertices are very sparse compared to the density of pixels. The number of vertices in our models is typically less than 1000. Running CDT at such a scale does not take much time, either. In addition to the original set of constraints for TPS fitting, CDT typically inserts new vertices into the mesh to guarantee mesh quality. The depth values at these new vertices are obtained from the TPS interpolation which thus becomes indispensable.

8 Reconstruction examples

We have reconstructed multiple objects using our interactive reconstruction system. The results are shown in Figs. 5–7. Four to twenty images were used for each of the models. The more views of an object we use, a more complete 3D model we can recover. Because of our emphasis on salient curves, a texture-mapped model can faithfully reproduce the original appearances of an object even from a very sparse set of images. This is demonstrated in Fig. 6. From the reconstructed curvilinear structures shown in Figs. 5–7, it is clear that these structures provide a compact shape description of the type of objects considered in this paper. The thin-plate spline surfaces estimated using these curves have high visual quality for texture-mapping. There are 303 vertices and 439 triangles for the model shown in Fig. 5; 335 vertices and 515 triangles for the model shown in Fig. 5; and 736 vertices and 1216 triangles for the model shown in Fig. 7. Synthetically rendered images of the reconstructed models can be generated from arbitrary viewpoints.

Given the user-marked point and curve features, the 3D reconstruction algorithms only took about 10–15 min

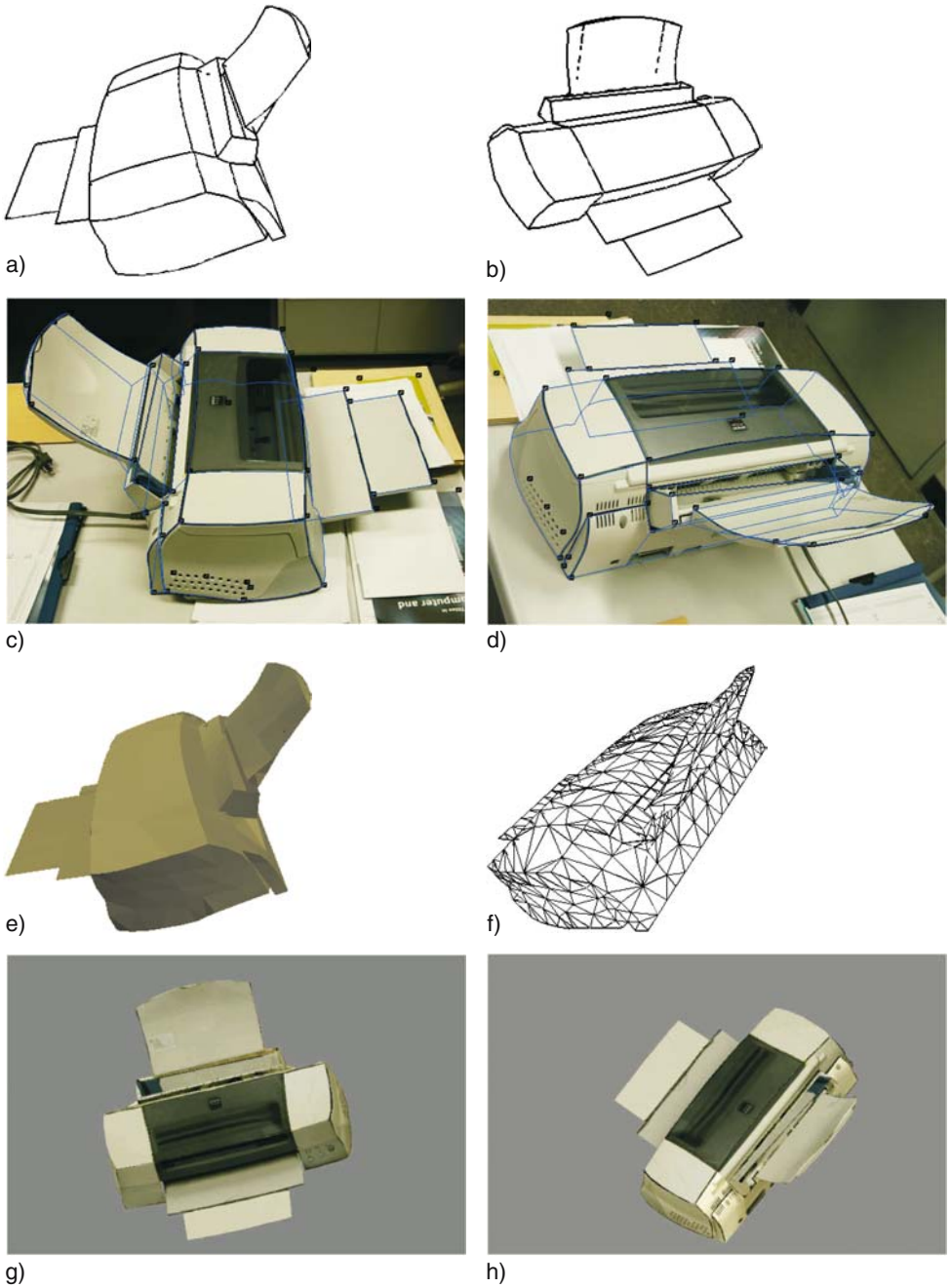

Fig. 5a–h. **a**,**b** Two views of the reconstructed 3D curvilinear structure of the printer shown in Fig. 2. **c**,**d** The reconstructed curvilinear structures can be projected back onto the input images to verify their accuracy. The user-marked curves are shown in black while the reprojected curves are shown in blue. **e** A triangle mesh is obtained by discretizing the reconstructed spline surface patches. **f** The wireframe of the triangle mesh shown in (**e**). **g**,**h** Two views of the texture-mapping result for the recovered printer model

to recover the aforementioned models on a standard Pentium 4 processor. Because the user may need to refine feature marking and selection, user interaction and automated 3D reconstruction need to be repeated alternatively a few times to obtain the final results. In summary, it took from one hour to a few hours to finish the whole process and produce the final version of each object. More than half of the time was spent on user interaction. The amount of user interaction is a limitation of our method. However, it is justified by the difficulty of automatic detection of high-curvature feature curves which are mostly geometric features instead of pixel intensity features. Automated feature detection is only possible when there are reasonable pixel intensity variations across the curves. For example, in Fig. 6, the whole object has a more or less uniform color and it is infeasible to detect some of the user-marked curves automatically if they do not happen to be intensity features. Nevertheless, humans can locate these curves using their prior knowledge of the object. Also in Fig. 7, the strong specular reflectance of the object surface produces many reflected textures which would significantly inter-

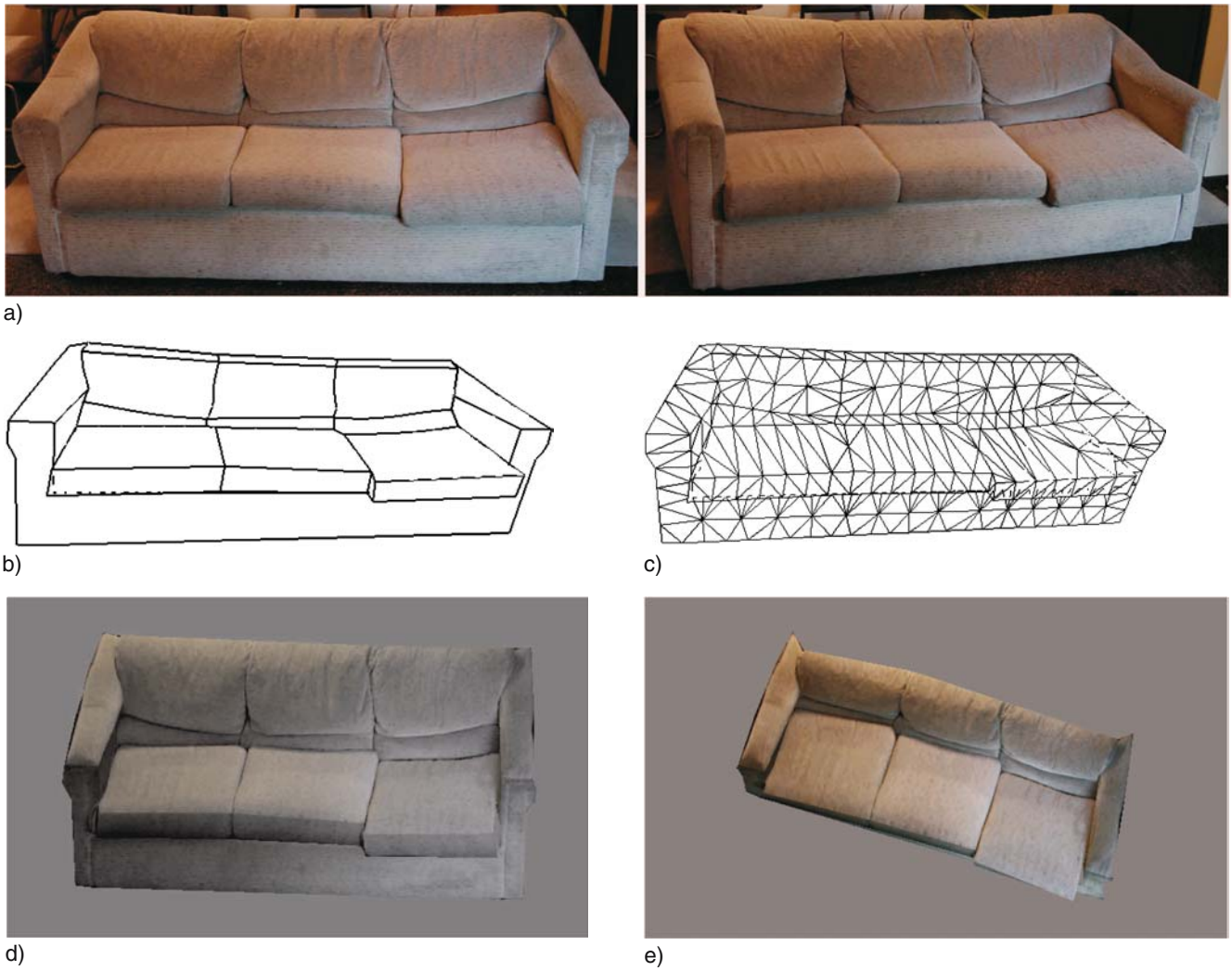

Fig. 6a–e. a Two of the four input images used for a couch. **b**item The reconstructed 3D curvilinear structure of the couch. **c** The wireframe of the discretized triangle mesh for the couch. **d**,**e** Two views of the texture-mapping result

fere with automatic surface curve detection. Therefore, we mean "salient curves" from a human perspective instead of from the machines'. When a free-form object does not seem to have recognizable salient curves from a human observer, our approach becomes inappropriate for its reconstruction.

As shown in Fig. 5c–d, the user can verify the accuracy of the recovered vertices and curves by reprojecting them back onto the original images. Usually, the projected vertices and curves deviate from the user-marked features by one pixel or less. Actually, the user does not have to be extremely careful in feature marking to achieve this accuracy. Typically, one only needs to mark a sparse set of key points on a curve and a spline interpolating these key points would be sufficient. In summary, such an accuracy is achieved through multiple measures in image acquisition, automatic 3D reconstruction and user interaction:

- The baseline between every pair of images should be relatively large. As in stereopsis, a large baseline makes the reconstruction less sensitive to errors in feature location.
- There should be at least one baseline not parallel to each surface curve. Otherwise, the reconstruction algorithm in Sect. 4 would not produce acceptable results.
- We use bundle adjustment in both camera pose estimation and curve reconstruction to make the final reconstruction less sensitive to errors in individual feature marking.
- The reprojected feature locations provide feedback to the user who can move a marked feature to a more accurate position once a marking error has been discovered. Thus, a user marking error behaves like an outlier in the reconstruction process and can be interactively eliminated.

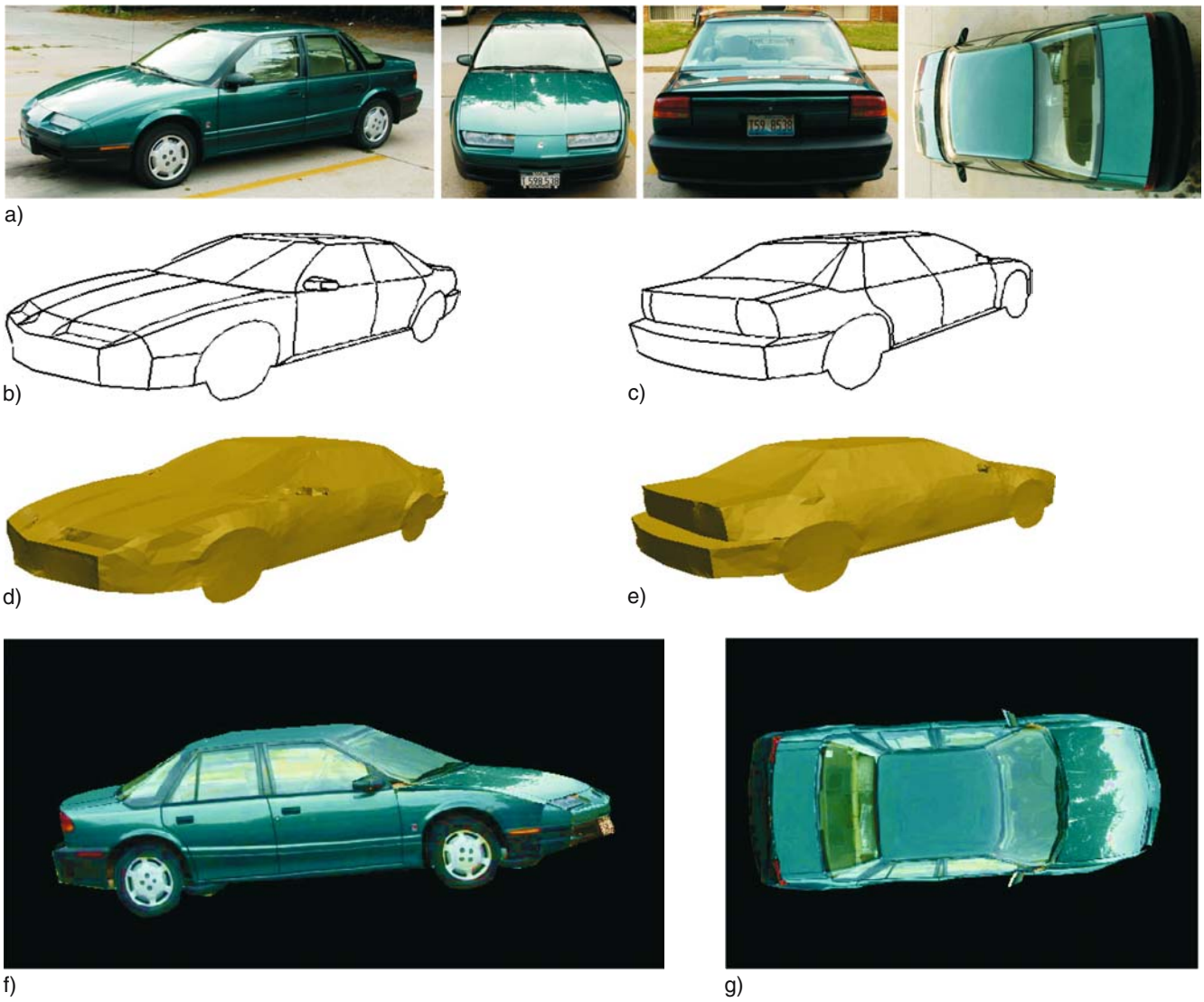

Fig. 7a–g. a Four of the input images used for an automobile. **b**,**c** Two views of the reconstructed 3D curvilinear structure of the automobile. **d**,**e** Two views of a high-resolution triangle mesh for the automobile. **f**,**g** Two views of the texture-mapping result. The image on the right shows an aerial view from the top

In Figs. 5–7, there were no images covering the bottom planes of the objects. Therefore, the bottom planes were missing in the reconstructed models. However, the user can always interactively fill these missing regions using simple primitives such as planar polygons in a postprocessing step. Because texture-mapping is able to provide additional details, certain small-scale 3D structures can be simplified during reconstruction. For example, the bottom part of the paper feeder of the printer has been simplified to a frustum. Note that lines are a special case of curves. A 3D line segment can be obtained immediately once its two endpoints have been recovered. We use line segments whenever appropriate because of the convenience they provide.

9 Conclusions and future work

In this paper, we have introduced a photogrammetric method for recovering free-form objects with curvilinear structures. The result is an object surface model that can be discretized into a triangle mesh. Realistic renderings of the object model can be generated through texture-mapping.

There are possible improvements and extensions to our reconstruction system. Additional shape constraints should be exploited to further reduce the number of input images and the amount of user interaction. For example, many man-made objects have reflective, rotational

or translational symmetries. The enforcement of such symmetries can further improve the quality of the reconstructed models. The rendering part of our system can also be improved. Instead of texture-mapping, one should be able to recover surface photometric properties from photographs using the techniques in [24, 27, 35] to obtain a lighting independent representation of each object.

References

1. Berthilsson R, Astrom K (1997) Reconstruction of 3d-curves from 2d-images using affine shape methods for curves. In: IEEE Conference on Computer Vision and Pattern Recognition
2. Berthilsson R, Astrom K, Heyden A (1999) Reconstruction of curves in R^3, using factorization and bundle adjustment. In: International Conference on Computer Vision
3. Canoma. www.canoma.com
4. Cipolla R, Blake A (1992) Surface shape from the deformation of the apparent contour. Int J Comput Vis 9(2):83–112
5. Debevec PE, Taylor CJ, Malik J (1996) Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In: SIGGRAPH '96, pp 11–20
6. Eck M, Hoppe H (1996) Automatic reconstruction of b-spline surfaces of arbitrary topological type. In: Computer Graphics (SIGGRAPH Proceedings), pp 325–329
7. Faugeras O (1993) Three-dimensional computer vision. MIT Press, Cambridge, MA
8. Faugeras O, Laveau S, Robert L, Csurka G, Zeller C (1995) 3-d reconstruction of urban scenes from sequences of images. In: Gruen A, Kuebler O, Agouris P (eds) Automatic extraction of man-made objects from aerial and space images. Birkhauser
9. Giblin P, Weiss R (1986) Reconstruction of surfaces from profiles. In: International Conference on Computer Vision, pp 136–144
10. Hartley R, Zisserman A (2000) Multiple view geometry in computer vision. Cambridge University Press
11. Hoppe H, DeRose T, Duchamp T, Halstead M, Jin H, McDonald J, Schweitzer J, Stuetzle W (1994) Piecewise smooth surface reconstruction. In: Computer Graphics (SIGGRAPH Proceedings), pp 295–302
12. Kaminski J, Fryers M, Shashua A, Teicher M (2001) Multiple view geometry of non-planar algebraic curves. In: International Conference on Computer Vision
13. Krishnamurthy V, Levoy M (1996) Fitting smooth surfaces to dense polygon meshes. In: Computer Graphics (SIGGRAPH Proceedings), pp 313–324
14. Liebowitz D, Criminisi A, Zisserman A (1999) Creating architectural models from images. In: Proc of Eurographics
15. Longuet-Higgins HC (1981) A computer algorithm for reconstructing a scene from two projections. Nature 293:133–135
16. Mantyla M (1988) Introduction to solid modeling. WH Freeman
17. Meinguet J (1979) Multivariate interpolation at arbitrary points made simple. J Appl Math Phys 5:439–468
18. Mok3. www.mok3.com.
19. Murray RM, Li Z, Sastry SS (1994) A mathematical introduction to robotic manipulation. CRC Press
20. Papadopoulo T, Faugeras O (1996) Computing structure and motion of general 3d curves from monocular sequences of perspective images. In: European Conference on Computer Vision, pp 696–708
21. Polak E (1997) Optimization–algorithms and consistent approximations. Springer, Berlin Heidelberg New York
22. Poulin P, Ouimet M, Frasson MC (1998) Interactively modeling with photogrammetry. In: Eurographics workshop on rendering
23. Powell MJD (1995) A thin plate spline method for mapping curves into curves in two dimensions. In: Computational Techniques and Applications
24. Ramamoorthi R, Hanrahan P (2001) A signal-processing framework for inverse rendering. In: Proceedings of SIGGRAPH, pp 117–128
25. Realviz – image processing software and solutions for content creation. www.realviz.com/products/im/index.php.
26. Robert L, Deriche R (1996) Dense depth map reconstruction: a minimization and regularization approach which preserves discontinuities. In: European Conference on Computer Vision
27. Sato Y, Wheeler MD, Ikeuchi K (1997) Object shape and reflectance modeling from observation. In: Computer Graphics Proceedings, Annual Conference Series, pp 379–388
28. Schmid C, Zisserman A (1998) The geometry and matching of curves in multiple views. In: European Conference on Computer Vision
29. Soucy M, Godin G, Rioux M (1996) A texture-mapping approach for the compression of colored 3d triangulations. Visual Comput 12:503–514
30. Sullivan S, Ponce J (1998) Automatic model construction and pose estimation from photographs using triangular splines. IEEE Trans Pattern Anal Mach Intell 20(10):1091–388
31. Triggs B, McLauchlan P, Hartley R, Fitzgibbon A (2000) Bundle adjustment – a modern synthesis. In: Vision algorithms: theory and practice. Springer, Berlin Heidelberg New York, pp 298–375
32. Tsai R (1987) A versatile camera calibration technique for high accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. IEEE J Robot Automat 3(4):323–344
33. Tubic D, Hübert P, Laurendeau D (2003) 3d surface modeling from range curves. In: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2003)
34. Wahba G (1990) Spline models for observational data. SIAM
35. Yu Y, Debevec P, Malik J, Hawkins T (1999). Inverse global illumination: Recovering reflectance models of real scenes from photographs. In: Proceedings of SIGGRAPH, pp 215–224
36. Yu Y, Ferencz A, Malik J (2001) Extracting objects from range and radiance images. IEEE Trans Visual Comput Graph 7(4)
37. Zhang L, Dugas-Phocion G, Samson J-S, Seitz SM (2001) Single view modeling of free-form scenes. In: Proc Computer Vision and Pattern Recognition
38. Zorin D, Schröder P, Sweldens W. Interpolating subdivision for meshes with arbitrary topology. In: Computer Graphics (SIGGRAPH Proceedings), pp 189–192

HONG WU received her MS degree in computer science from the University of Cincinnati in 1996 and BS degree in computer science and engineering from Zhejiang University, China, in 1992. She has been working in the computer software industry since 1996 and has held an engineering or managerial position at General Magic, Oracle, Extricity Software and WebEx Communications. She is currently with Dimentionality LLC. Her work responsibilities ranged from software development, technical alliances to product management. She was the senior group product manager in the CTO office of WebEx Communications where she was responsible for new multimedia product initiatives.

YIZHOU YU is currently an assistant professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. He received his PhD degree in Computer Science from University of California at Berkeley in 2000, and his MS in Applied Mathematics and BS in Computer Science from Zhejiang University, China, in 1994 and 1992, respectively. He has done research in computer graphics and vision including fluid and hair simulation, image-based modeling and rendering, texture synthesis, visibility and mesh processing, radiosity and global illumination, and has authored or co-authored more than 30 research papers. He is a recipient of 2002 National Science Foundation Career Award and 1998 Microsoft Graduate Fellowship. His current interests include computer animation, image-based graphics, image registration, texture analysis and synthesis, and other computer graphics and vision problems.