3-D Modeling from a Single View of a Symmetric Object

Tianfan Xue, *Student Member, IEEE*, Jianzhuang Liu, *Senior Member, IEEE*, and Xiaoou Tang, *Fellow, IEEE*

***Abstract*— 3-D technologies are considered as the next generation of multimedia applications. Currently, one of the challenges faced by 3-D applications is the shortage of 3-D resources. To solve this problem, many 3-D modeling methods are proposed to directly recover 3-D geometry from 2-D images. However, these methods on single view modeling either require intensive user interaction, or are restricted to a specific kind of object. In this paper, we propose a novel 3-D modeling approach to recover 3-D geometry from a single image of a symmetric object with minimal user interaction. Symmetry is one of the most common properties of natural or manmade objects. Given a single view of a symmetric object, the user marks some symmetric lines and depth discontinuity regions on the image. Our algorithm first finds a set of planes to approximately fit to the object, and then a rough 3-D point cloud is generated by an optimization procedure. The occluded part of the object is further recovered using symmetry information. Experimental results on various indoor and outdoor objects show that the proposed system can obtain 3-D models from single images with only a little user interaction.**

***Index Terms*— Interactive 3-D modeling, single view, symmetry.**

I. Introduction

IN THE PAST years, there has been rapid development in 3-D applications, including 3-D TV and movie, virtual reality, 3-D city, and 3-D object retrieval. However, compared with the developments of 3-D applications, 3-D resources are still limited. Traditionally, 3-D models are either captured by 3-D cameras or designed by designers using computer-aided design tools. The former requires careful calibration and scans objects under strict conditions, and the latter is manually intensive.

Manuscript received August 19, 2011; revised April 29, 2012; accepted May 3, 2012. Date of publication May 22, 2012; date of current version August 22, 2012. This work was supported in part by the Natural Science Foundation of China, under Grant 60975029 and Grant 61070148, by the Science, Industry, Trade, Information Technology Commission of Shenzhen Municipality, China, under Grant JC200903180635A, Grant JC201005270378A, and Grant ZYC201006130313A, and the Introduced Innovative R&D Team of Guangdong Province "Robot and Intelligent Information Technology." The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Carlo S. Regazzoni.

T. Xue and X. Tang are with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, and also with the Shenzhen Key Laboratory for Computer Vision and Pattern Recognition, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: xtf009@ie.cuhk.edu.hk; xtang@ie.cuhk.edu.hk).

J. Liu is with Media Laboratory, Huawei Technologies Company Ltd., Shenzhen 518129, China, and also with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: jzliu@ie.cuhk.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2012.2200494

Recently, there have been works focusing on recovering 3-D information directly from a single 2-D image. These methods fall into two categories. One is fully automatic modeling methods [1]–[5]. Blanz and Vetter proposed a morphable face model that models the shape and texture of a testing face as a linear combination of a set of training images [1]. This morphable model requires the training 3-D objects have very similar topologies (faces in [1]) and small texture variations. Huang and Cowan proposed to recover 3-D geometry from an indoor image, using a set of perspective geometric cues [2]. Lee *et al*. also tried to obtain the 3-D geometry from an indoor image based on some assumptions in indoor environments [3]. Leetta and Mundy developed a deformable vehicle model to recover the 3-D shape of a vehicle from a single view of it [4]. They use a multiresolution model to model the variations among different kinds of vehicles. Sigal *et al*. learned a parameterized mesh model from a database of 3-D human bodies and use this model to build an automatic shape and pose estimation system with a single image being the input [5]. Although these methods may have good performance on the objects they focus on, they utilize many object-specific assumptions and can hardly extend to other objects.

The other category is interactive 3-D modeling methods that build a fine 3-D model with the help of user interactions [6]–[16]. Debevec *et al*. combined both geometry-based and image-based techniques to model an architecture from a set of images and render this architecture in a new view [10]. In this paper, the user marks all the edges of the architecture in the images and provides a rough 3-D model, and then the algorithm generates a refined 3-D model from the images. Zhang *et al*. proposed a modeling system to get a free-form curved surface from a single image [12]. The modeling result mainly depends on the user's marks, which specify a set of geometric constraints about the recovered surface. In [16], the user first draws lines along the edges of the objects and a 3-D model is recovered from these edges using some image regularities. These interactive methods usually require intensive user interaction.

We focus on recovering 3-D geometry from single images of reflectionally symmetric objects in this paper.[1] Symmetry is ubiquitous in natural and manmade objects, and provides rich information for 3-D reconstruction. Recently, researchers have proposed some 3-D modeling algorithms

[1] In the rest of this paper, we simply use "symmetry" to denote "reflectional symmetry."

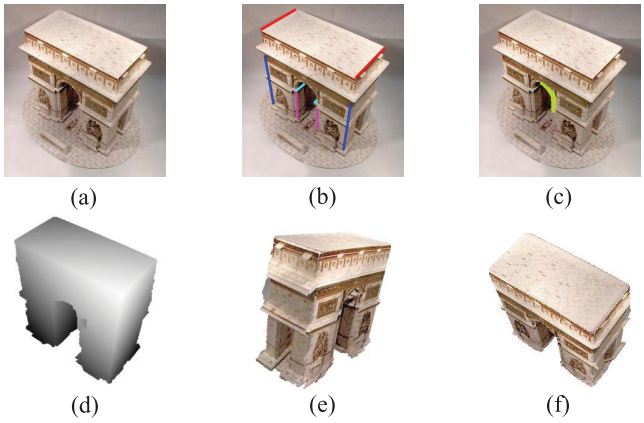


Fig. 1. (a) Image. (b) and (c) User interaction marking the symmetric lines and depth discontinuity region. (d) Recovered depth map. (e) and (f) 3-D reconstruction result of our algorithm shown in two different views with texture mapped.

based on symmetry. Hong *et al.* [17] used a "canonical" coordinate frame to automatically recover 3-D shape from a single image. This method can only deal with objects with repetitive patterns, such as a wall consisting of repetitive tiles [17]. Yang *et al.* studied the homography groups and proposed a framework to estimate the poses and structures of 2-D symmetric patterns [18]. François *et al.* [11] proposed to get the 3-D geometry of a symmetric object from an image overlaid by straight line drawings, which requires intensive user interaction and cannot deal with objects with curved edges. Jiang *et al.* [6] recovered the 3-D geometry of a symmetric object from faces marked by the user. Although a detailed 3-D model can be built, this method focuses on architectural objects only.

In this paper, we propose a novel 3-D reconstruction method from a single view of a symmetric object with a little user interaction. Fig. 1 shows an example. Given an input image of a symmetric object [Fig. 1(a)], the user first marks symmetric lines [Fig. 1(b)] and depth discontinuity regions [Fig. 1(c)]. Based on these marks, our system first finds the planes that approximate the surface of the object. Then it generates a 3-D point cloud using a Markov random field (MRF). The invisible part of the object is recovered with symmetry information, and noise in the 3-D point cloud is removed by a Gaussian filter. For the object in Fig. 1(a), the recovered depth map and 3-D model are shown in Fig. 1(d)–(f), respectively. This paper originates from our previous work [19]. In [19], we develop an automatic reconstruction algorithm using symmetry information. However, since that method is fully automatic, it works well only for objects with simple textures [e.g., it cannot handle the object in Fig. 1(a)]. In this paper, we propose a different algorithm based on user interactions, which can deal with much more complex objects than the automatic method. The algorithm proposed in this paper is different from [19] in the following aspects: 1) we design a more robust plane detection algorithm with user's marks, which works well even for objects with complex textures; 2) we construct a more complete MRF energy function with the information provided by the user to avoid the ambiguity (multiple explanations) faced by [19]; and 3) we use a more robust scheme to estimate vanishing points and remove reconstruction errors with Gaussian filtering. The proposed 3-D modeling system is easy to use and can handle a large class of symmetric objects. Experimental results show that this method works well on various indoor and outdoor objects, and is robust to user's marking errors.

II. Geometric Properties in 3-D Geometry from a Single View of a Symmetric Object

A. Camera Model

We use a simplified camera model in this paper. The camera has zero skew and does not have radial distortion, and the aspect ratio of the pixel equals 1. In this model, the projection matrix is

$$M = [K|0], \quad K = \begin{pmatrix} -f & 0 & u_0 \\ 0 & -f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

where f is the focal length of the camera, and (u_0, v_0) is the position of the principle point in the camera coordinate system.

In the rest of this paper, a bold upper-case letter (say, $\mathbf{X}$) denotes the homogeneous coordinate of a 3-D point, and its 2-D projection on the image plane is denoted by the corresponding bold lower-case letter $\mathbf{x}$. A plane $n_x x + n_y y + n_z z + d = 0$ is represented by $\boldsymbol{\pi} = (n_x, n_y, n_z, d)^\top = (\mathbf{n}^\top, d)^\top$, where $\mathbf{n} = (n_x, n_y, n_z)^\top$ is the normal of the plane. If not specified, homogeneous coordinates are used, and variables in Euclidean coordinates are represented by letters with a tilde above them, such as $\widetilde{\mathbf{X}}$.

B. Geometric Properties

In this section, we show some geometric properties used to obtain 3-D geometry from a single view of a symmetric object. At this moment, we suppose that the camera matrix is already known and symmetric point pairs are already detected. The computation of the matrix and symmetric point detection will be discussed in Section III-A. We call a pair of points (lines), a symmetric point pair (line pair) if they are symmetric with respect to the symmetric plane of the object.

We first define the epipole and epipolar lines in Definition 1. These two terms have often appeared in 3-D geometry-related literature in computer vision [20], [21]. For completeness and easy understanding of our work, we show their definitions again here.

Definition 1: Let $\boldsymbol{\pi} = (\mathbf{n}^\top, d)^\top$ be the symmetry plane of a symmetric object. The epipole is the vanishing point of the lines parallel to $\mathbf{n}$. The epipolar lines are the lines passing through the epipole.

Fig. 2 gives an example of the epipole and epipolar lines. In perspective geometry, the 2-D projections of a set of 3-D parallel lines converge to the same point, which is the vanishing point of these lines (or these directions). According to [20], the vanishing point of a line l is at $K\mathbf{n}_l$ in the image plane, where $\mathbf{n}_l$ is the direction of l. In our case, this vanishing point is the epipole and $\mathbf{n}_l = \mathbf{n}$. Thus, the epipole is at $K\mathbf{n}$. Next, we introduce the following property.

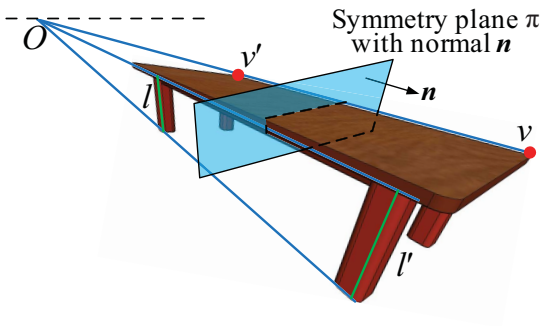


Fig. 2. Illustration of an epipole and epipolar lines. O is the epipole and the three blue lines passing through O are epipolar lines. The symmetric point v' of v lies on the epipolar line passing through v. Two lines l and l' are symmetric lines.

Property 1: Let X and X' be a pair of symmetric points with respect to the symmetry plane $\boldsymbol{\pi}$. Then their 2-D projections $\mathbf{x}$ and $\mathbf{x}'$ and the epipole lie on the same line.

Proof: Let the symmetry plane be $\boldsymbol{\pi} = (\mathbf{n}^\top, d)^\top$. Since $\mathbf{X}$ and $\mathbf{X}'$ are symmetric with respect to $\boldsymbol{\pi}$, we have

$$\alpha(\widetilde{\mathbf{X}} - \widetilde{\mathbf{X}}') = \boldsymbol{n} \tag{2}$$

where α is a nonzero scalar. Since $\mathbf{x} = K\widetilde{\mathbf{X}}$ and $\mathbf{x}' = K\widetilde{\mathbf{X}}'$, multiplying K on the both sides of (2), we have

$$\alpha\mathbf{x} - \alpha\mathbf{x}' - K\mathbf{n} = \mathbf{0}. \tag{3}$$

Since $K\mathbf{n}$ is the epipole, $\mathbf{x}$, $\mathbf{x}'$, and the epipole lie on the same line. ∎

The following Property 2 shows how to compute the 3-D positions of two points, if they are symmetric with respect to $\boldsymbol{\pi}$.

Property 2: Let $M = [K|\mathbf{0}]$ be the projection matrix, $\boldsymbol{\pi} = (\mathbf{n}^\top, d)^\top$ be the symmetry plane, and $\mathbf{x}$ and $\mathbf{x}'$ ($\mathbf{x} \neq \mathbf{x}'$) be a pair of symmetric points in the 2-D image plane. If $K\mathbf{n} = \alpha\mathbf{x} - \alpha\mathbf{x}'$, then the corresponding 3-D points of $\mathbf{x}$ and $\mathbf{x}'$ in Euclidean coordinates are

$$\widetilde{\mathbf{X}} = \frac{\alpha d K^{-1}\mathbf{x}}{\frac{1}{2} + \alpha\mathbf{n}^\top K^{-1}\mathbf{x}'} \tag{4}$$

$$\widetilde{\mathbf{X}}' = \frac{\alpha d K^{-1}\mathbf{x}'}{\frac{1}{2} + \alpha\mathbf{n}^\top K^{-1}\mathbf{x}'}. \tag{5}$$

Proof: Let $\mathbf{X}^\top = (\mathbf{X}_{xyz}^\top, X_r)$ and $\mathbf{X}'^\top = (\mathbf{X}_{xyz}'^\top, X_r')$. Since $M\mathbf{X} = \mathbf{x}$, $M\mathbf{X}' = \mathbf{x}'$, and $M = [K|\mathbf{0}]$, we have

$$\mathbf{X}_{xyz} = K^{-1}\mathbf{x}, \quad \mathbf{X}_{xyz}' = K^{-1}\mathbf{x}'. \tag{6}$$

As $\mathbf{X}$ and $\mathbf{X}'$ are a pair of symmetric points with respect to the plane $\boldsymbol{\pi} = (\mathbf{n}^\top, d)^\top$, according to [22], there is a scalar h such that

$$\begin{cases} h\mathbf{X}_{xyz} = \mathbf{X}_{xyz}' - 2\mathbf{n}\mathbf{n}^\top\mathbf{X}_{xyz}' + 2dX_r'\mathbf{n} \\ hX_r = X_r'. \end{cases} \tag{7}$$

Suppose that $||\mathbf{n}|| = 1$ without loss of generality. Then, (6) and (7) lead to

$$K\mathbf{n} = \frac{h\mathbf{x}}{2dX_r' - 2\mathbf{n}^\top K^{-1}\mathbf{x}'} - \frac{\mathbf{x}'}{2dX_r' - 2\mathbf{n}^\top K^{-1}\mathbf{x}'}. \tag{8}$$

Since $K\mathbf{n} = \alpha\mathbf{x} - \alpha\mathbf{x}'$, we have

$$\alpha = \frac{h}{2dX_r' - 2\mathbf{n}^\top K^{-1}\mathbf{x}'} \tag{9}$$

$$-\alpha = -\frac{1}{2dX_r' - 2\mathbf{n}^\top K^{-1}\mathbf{x}'}. \tag{10}$$

Thus

$$X_r' = X_r = \frac{1 + 2\alpha\mathbf{n}^\top K^{-1}\mathbf{x}'}{2d\alpha}. \tag{11}$$

Finally, we obtain

$$\widetilde{\mathbf{X}} = \frac{1}{X_r}\mathbf{X}_{xyz} = \frac{\alpha d K^{-1}\mathbf{x}}{\frac{1}{2} + \alpha\mathbf{n}^\top K^{-1}\mathbf{x}'} \tag{12}$$

$$\widetilde{\mathbf{X}}' = \frac{1}{X_r'}\mathbf{X}_{xyz}' = \frac{\alpha d K^{-1}\mathbf{x}'}{\frac{1}{2} + \alpha\mathbf{n}^\top K^{-1}\mathbf{x}'}. \tag{13}$$

∎

Note that when the projection matrix $M = [K|\mathbf{0}]$ is given and the normal $\mathbf{n}$ of the symmetry plane is fixed, given a set of pairs of symmetric points, the 3-D positions of these points can be calculated with (4) and (5). Furthermore, from (4) and (5) we can see that, if the symmetry plane moves along its normal direction $\mathbf{n}$ (only d changes in this case), the 3-D coordinates of these points will only proportionally increase/decrease (i.e., the shape of the object changes only up to a scale).

III. Symmetry Based 3-D Modeling

Direct recovery of a fine 3-D model from a single image of an object is a challenging problem. Our strategy is first to find some planes to approximate the 3-D object (these planes are called underlying planes). Then the pixels of the object in the image are projected to the underlying planes, resulting in a 3-D point cloud. The occluded part of the object is also added using symmetry information. Finally, a 3-D mesh is generated from these 3-D points. The steps of our algorithm are shown in Fig. 3.

A. Segmentation and Calibration

Same as previous works, we assume that the object of interest has been segmented from the background, which can easily be done through GrabCut [23]. There are also plenty of images of symmetric objects without background in commercial image sharing web sites, such as Flickr and Google.

The calibration matrix is estimated from the vanishing points of three mutually orthogonal directions in a single image. Most previous vanishing point estimation methods [24] find vanishing points by voting edge directions. However, these methods may fail to detect some vanishing points, if there are no dominant edges corresponding to these vanishing points.

In this paper, we use a novel vanishing point estimation method with the help of user interaction. Suppose the three dominant orthogonal directions of the world coordinate frame are x, y, and z, and the vanishing points corresponding to them are $\mathbf{v_x}$, $\mathbf{v_y}$, and $\mathbf{v_z}$. The user draws two corners $(O, \vec{x}, \vec{y}, \vec{z})$ and $(O', \vec{x}', \vec{y}', \vec{z}')$ at two positions of the object on the 2-D image.

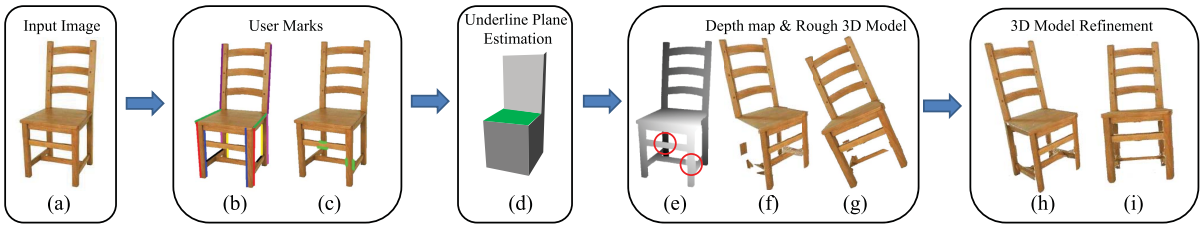


Fig. 3. Reconstruction procedure. (a) Input image. (b) Pairs of symmetric lines. (c) Discontinuity regions marked in green. (d) Underlying planes found to approximate the object. (e) Recovered depth map. (f) and (g) Coarse 3-D model shown in two views. (h) and (i) Refined 3-D model shown in two views.

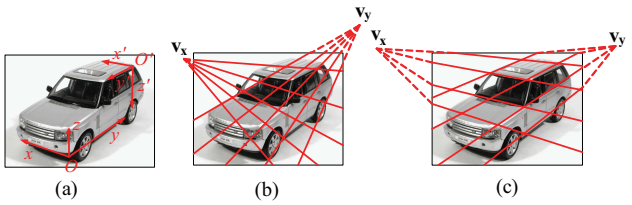


Fig. 4. Vanishing point detection. (a) Input image with corners $(O, \vec{x}, \vec{y}, \vec{z})$ and $(O', \vec{x}', \vec{y}', \vec{z}')$ marked by the user. (b) Perspective effect graph with initial vanishing points (rays corresponding to v_z are not shown for clarity). (c) Perspective effect graph with vanishing points after the adjustment (note that v_x and v_y are not in the proper location due to space limitations).

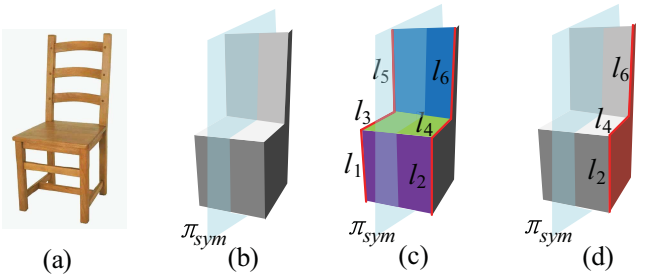


Fig. 5. Plane detection. (a) Input image. (b) Underlying planes. (c) Planes perpendicular to $\boldsymbol{\pi}$ marked by purple, green, and blue. (d) Plane not perpendicular to $\boldsymbol{\pi}$ marked by brown.

A corner consists of a center point O with three rays $\vec{x}$, $\vec{y}$, and $\vec{z}$ pointing to the vanishing points $\mathbf{v_x}$, $\mathbf{v_y}$, and $\mathbf{v_z}$, as shown in Fig. 4(a). Then the intersection of $\vec{x}$ and $\vec{x}'$ is $\mathbf{v_x}$, $\mathbf{v_y}$, and $\mathbf{v_z}$ are obtained similarly.

Since there may be some errors on the corners drawn by the user, these initial estimations of $\mathbf{v_x}$, $\mathbf{v_y}$, and $\mathbf{v_z}$ may not be accurate. The estimations are improved as follows: First, a perspective effect graph is drawn automatically on the image plane [see Fig. 4(b)], where each vanish point emits a set of rays. With the help of this perspective effect graph, the user then adjusts the position of each vanishing point to make these rays coincident with object edges in the image. Fig. 4(c) shows the vanishing points after the adjustment. Finally, the calibration matrix is estimated from them using the method in [20].

The position of the symmetry plane $\boldsymbol{\pi} = (\mathbf{n}^\top, d)^\top$ is calculated as follows. Since the 2-D epipole $\mathbf{v_x} = K\mathbf{n}$ (or $\mathbf{v_y} = K\mathbf{n}$), we have $\mathbf{n} = K^{-1}\mathbf{v_x}$ (or $\mathbf{n} = K^{-1}\mathbf{v_y}$). d is set to an arbitrary value, because as discussed in Section II-B, the choice of d does not affect the shape of the recovered 3-D object.

B. User's Marks

Two kinds of information are provided by the user: symmetric line pairs and depth discontinuity regions. The user marks some pairs of symmetric lines in the image, as shown in Fig. 3(b), where two lines of the same color form a symmetric line pair. The user marks at least two lines on each underlying plane, so that the plane can be detected later. For example, for the green plane in Fig. 3(d), the user marks two green lines shown in Fig. 3(b).

A depth discontinuity region is a place where the depth map is discontinuous. For example, there is depth discontinuity on two the legs of the chair [indicated by the red circles in Fig. 3(e)]. The user marks these discontinuity regions as shown in green in Fig. 3(c).

C. Underlying Plane Detection

First, we can compute the 3-D position of the symmetry plane $\boldsymbol{\pi}$, and the 3-D positions of symmetric lines, with the known calibration matrix [20]. To recover the 3-D geometry of an object, we approximate it by a set of planes (underlying planes), as shown in Fig. 5(b). For an object with a symmetry plane, it has two kinds of underlying planes. One is the planes perpendicular to $\boldsymbol{\pi}$, such as the purple, green, and blue planes in Fig. 5(c). Each plane of this kind passes through the two lines of a symmetric line pair. For example, in Fig. 5(c), the green plane passes through lines l_3 and l_4. The other is the planes not perpendicular to $\boldsymbol{\pi}$, such as the brown plane shown in Fig. 5(d). Each plane of this kind passes through at least two lines marked by the user. For example, in Fig. 5(d), the brown plane passes through lines l_2, l_4, and l_6.

Based on the above analysis, a plane detection algorithm is designed in Algorithm 1. Planes perpendicular to π_{sym} are detected in steps 1–5. For each symmetric line pair $(l_{i,1}, l_{i,2})$ marked by the user, we first calculate their 3-D positions using the method described in [11]. Then we find the plane that passes through $l_{i,1}$ and $l_{i,2}$. In steps 7–10, planes not perpendicular to π_{sym} are added. To find the 3-D plane passing through lines in L, for each line l in L, we first equally sample some points on L and add them to a point set P. Then we find the best fitting plane to these points using RANSAC in step 8, and once a fitting plane is found, those points near this plane are removed from P in step 9. Steps 8 and 9 are repeated until there are no more than four points in P.

Algorithm 1 Underlying plane detection

Input: A set of symmetric line pairs $L = \{(l_{i,1}, l_{i,2})\}$ marked by the user.

Initialization: The set of underlying planes $\Pi \leftarrow \phi$; a point set $P \leftarrow \phi$.

1) **for** $(l_{i,1}, l_{i,2}) \in L$
2) Calculate the 3-D positions of $l_{i,1}$ and $l_{i,2}$.
3) Add the plane passing through $l_{i,1}$ and $l_{i,2}$ to Π.
4) Equally sample 3-D points on lines $l_{i,1}$ and $l_{i,2}$, and add them to P.
5) **endfor**
6) Group planes in Π and points in P using mean-shift.
7) **while** there are more than four points in P
8) Find the plane π that best fits the points in P using RANSAC [25].
9) Add π and its symmetric plane π' with respect to π_{sym} to Π. Remove the points near π and π' from P.
10) **endwhile**

Return: The set of underlying planes Π.

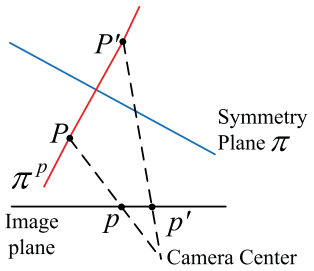


Fig. 6. Geometry to define the data term.

D. 3-D Point Cloud Generation

In the last section, we state how to obtain the underlying 3-D plane set Π and the 3-D positions of a few lines marked by the user. To derive the depth of each object pixel in the image, we need to find the symmetric pixel of each object pixel, which is not a trival problem. In this paper, we do it in reverse. We assign each foreground (object) pixel a label indicating which plane the pixel belongs to (the background is ignored). Then we use symmetry to validate whether this assignment is correct.

Finding the best labels for the pixels is formulated as a minimization problem with an MRF. The energy function to be minimized is defined as

$$E = \sum_{p} E_d(\boldsymbol{\pi}^p) + \sum_{(p,q)\in N_4} E_{ls}(\boldsymbol{\pi}^p, \boldsymbol{\pi}^q) + \sum_{(p,q)\in N_s} E_s(\boldsymbol{\pi}^p, \boldsymbol{\pi}^q) \tag{14}$$

where $\boldsymbol{\pi}^p \in \Pi$ is the plane pixel p is assigned to, N_4 is the four-neighborhood system, N_s is another pixel set defined later, and the three terms, E_d (data term), E_{ls} (local smoothness term), and E_s (symmetry term), will be described as follows.

1) Data Term: The data term is to determine which plane a pixel belongs to. Suppose p is assigned to plane $\boldsymbol{\pi}^p$. We first back-project p to the 3-D plane $\boldsymbol{\pi}^p$ and get its 3-D symmetric point P' with respect to $\boldsymbol{\pi}$. Then P' is projected to the image plane, resulting in p' (see Fig. 6). If p is on a line $l_{i,1}$ marked by the user (case 1), then p' must lie on the symmetric line $l_{i,2}$ of $l_{i,1}$. The following data term is used to ensure this constraint:

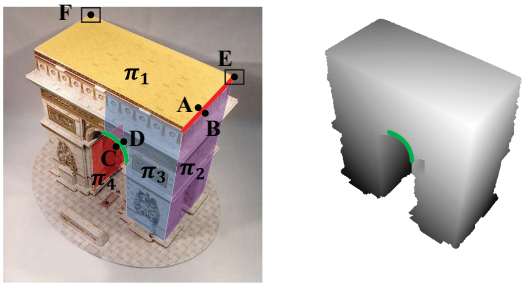
$$E_d(\boldsymbol{\pi}^p) = g(D(p', l_{i,2})) \tag{15}$$

where $D(p', l_{i,2})$ is the distance between point p' and line $l_{i,2}$, and g is a kernel function defined as $g(x) = \min\{x^2/\theta^2, 1\}$ with θ being a parameter. If p is not on any lines marked by the user (case 2), the similarity between p and p' reflects the correctness of p assigned to $\boldsymbol{\pi}^p$ (here, we assume that a pair of symmetric points have similar colors). Thus in case 2, the data term is defined as

$$E_d(\boldsymbol{\pi}^p) = \begin{cases} g(||I_p - I_{p'}||), & \text{if } p' \in I_F \\ 5, & \text{otherwise} \end{cases} \tag{16}$$

where I_F is the set of all object pixels, I_p is the 3-D vector of the RGB values of p, and g is the same kernel function as in (15) with a different parameter. A larger penalty five is assigned to the data term if p' is not on the object.

Here, we simply use the pixel intensity instead of patch-based features to build the data term. This is because that although two symmetric points have similar local topologies in 3-D space, the 2-D local regions around them may be



(a) (b)

Fig. 7. Illustration of the depth discontinuity. (a) Input image with user's marks. (b) Recovered depth map.

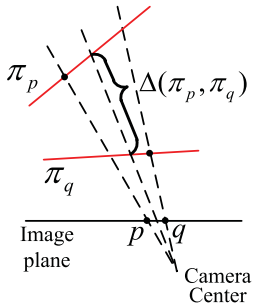


Fig. 8. Geometry to define the smoothness term.

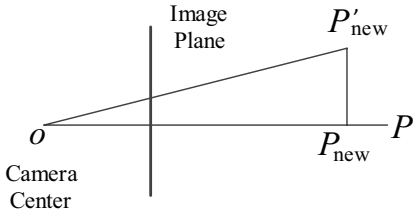


Fig. 9. Illustration of the point smoothing.

Fig. 10. Some experimental results. (a) Input images. (b) User's marks of symmetric line pairs. (c) User's marks of depth discontinuity regions in green. (d) Labeling results. (e) Recovered depth maps obtained by the algorithm in this paper. (f) and (g) Recovered 3-D models shown in two views with texture mapped. (h) Recovered depth maps obtained by the algorithm in [19].

Fig. 11. Another set of experimental results. (a) Input images. (b) User's marks of symmetric line pairs and depth discontinuity regions. (c) Labeling results and recovered depth maps obtained by the new algorithm. (d) Depth maps by [19] where the circles indicate the depth errors.

very different when they are projected to the 2-D plane. For example, in Fig. 7(a), the local region around point E is dissimilar to the local region around its symmetric point F. Therefore, we do not use the distance of local region descriptors or the cross correlation-based distance to build the data term.

2) Local Smoothness Term: The local smoothness constraint enforces neighboring 2-D pixels of the object are also close in the 3-D space. For example, in Fig. 7(a), point A and point B are close in the image. Since the plane $\boldsymbol{\pi}_1$ and plane $\boldsymbol{\pi}_2$ are connected in the 3-D space at the line marked by red, A and B are also close in the 3-D space. However, this constraint is not always satisfied. For example, in Fig. 7(a), point C and point D are close in the image but not in the 3-D space, because the plane $\boldsymbol{\pi}_3$ (which D belongs to) and the plane $\boldsymbol{\pi}_4$ (which C belongs to) are not connected in the 3-D space. There is a depth discontinuity at the region marked by green [see the depth map in Fig. 7(b)]. To obtain a good result, depth discontinuity regions are marked necessarily by the user.

With the user's marks, the local smoothness term is defined as follows: if two neighboring pixels[2] p and q have the same label, then $E_{ls}(\boldsymbol{\pi}^p, \boldsymbol{\pi}^q) = 0$. Otherwise, $E_{ls}(\boldsymbol{\pi}^p, \boldsymbol{\pi}^q)$ is set to be the following value to punish the spacial difference between neighboring pixels:

$$E_{ls}(\boldsymbol{\pi}^p, \boldsymbol{\pi}^q) = \begin{cases} 0, & \text{if } p \in I_D \text{ or } q \in I_D \\ \alpha_{ls}\, g(\Delta(\boldsymbol{\pi}^p, \boldsymbol{\pi}^q)), & \text{otherwise} \end{cases} \tag{17}$$

where $\alpha_{ls} = 300$ is a weight for the local smoothness term, I_D is the set of pixels in the depth discontinuity regions, g is the

[2]Four-neighborhood system is used.

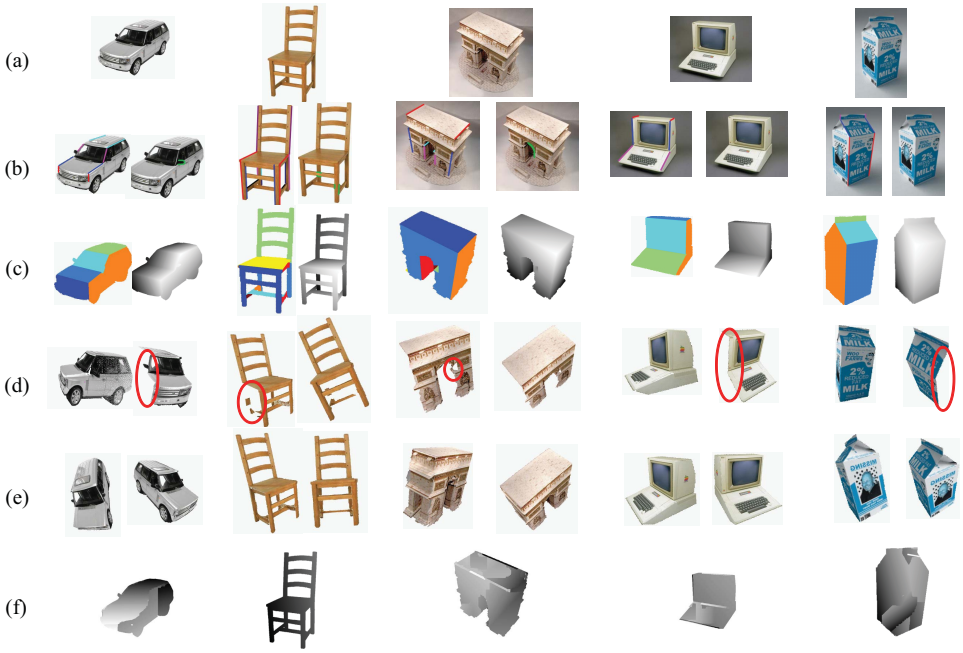


Fig. 12. (a) Input images. (b) User's marks of symmetric line pairs and depth discontinuity regions. (c) Labeling results and recovered depth maps obtained by the algorithm in this paper. (d) Coarse 3-D models (before refinement) shown in two views. (e) Refined 3-D models shown in two views. (f) Recovered depth maps obtained by the algorithm in [19].

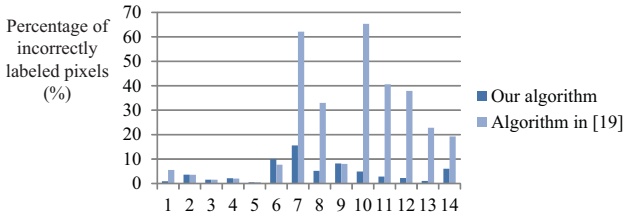


Fig. 13. Labeling accuracy comparison. Each two neighboring bars show the accuracy of the labeling results obtained by our algorithm and the algorithm in [19]. Numbers 1–6 denote the six objects in Fig. 10, numbers 7–9 denote the three objects in Fig. 11, and numbers 10–14 denote the five objects in Fig. 12.

same kernel function as in (15), and $\Delta(\boldsymbol{\pi}^p, \boldsymbol{\pi}^q)$ measures the spacial difference between $\boldsymbol{\pi}^p$ and $\boldsymbol{\pi}^q$ along the viewing ray passing through the middle point between p and q, as shown in Fig. 8. This local smoothness term ensures that the labels of neighboring pixels change only at the place where two planes are intersected, or at the depth discontinuity regions marked by the user.

3) Symmetry Term: If the symmetric pixel of a pixel p is q, then the symmetric pixel of q is also p. To enforce this constraint, the symmetry term is used to punish inconsistent pairs, defined as

$$E_s(\boldsymbol{\pi}^p, \boldsymbol{\pi}^q) = \begin{cases} \alpha_s, & \text{if } p' = q \text{ and } q' \neq p \\ \alpha_s, & \text{if } p' \neq q \text{ and } q' = p \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

where $\alpha_s = 10$ is a punishment constant, p' and q' are found by the method described in Section III-D1 (also see Fig. 6). N_s in (14) contains all the pixel pairs that have the potential to be symmetric, i.e., $N_s = \{(p, q) \mid p' = q \text{ or } q' = p \text{ for at least one plane}\}$.

The minimization of E in (14) is solved through expansion graph cuts Algorithm [26]. Although, it cannot be rigorously proved that the energy function satisfies the triangular formulation required by the algorithm, our experiments show that it obtains a better result than the swap graph cuts algorithm. After labeling all the pixels of the object, the depth map is computed by projecting each pixel to its assigned 3-D plane.

E. 3-D Model Refinement

After pixel assignments, a 3-D point cloud C is generated. Since the image is taken from a single view, some parts of the object are occluded and missing from C. Those missing parts are filled as follows. Let the set of 3-D points on one side of $\boldsymbol{\pi}$ be C_1, and the set of 3-D points on the other side be C_2, where $C_1 \cup C_2 = C$. For each point P in C_1, we find its symmetric point P' with respect to $\boldsymbol{\pi}$. If no point in C_2 falls into the 3-D ball centered at P' with radius r, then add P' to C_2. The missing points in C_1 are recovered in the same way.

The 3-D point cloud C contains discontinuity parts on the boundary of each plane. This is because in the previous steps, we use piecewise planar faces to approximate the curved faces of the object. To smooth the surface, the following Gaussian filter is applied to the 3-D point set:

$$P'_{\text{new}} = \sum_{P' \in N(P)} \exp\left(-\frac{||P - P'||^2}{\sigma^2}\right) P' \tag{19}$$

$$P_{\text{new}} = \frac{\overrightarrow{OP'_{\text{new}}} \cdot \overrightarrow{OP}}{||OP||^2} P \tag{20}$$

where P and P' are 3-D Euclidean coordinates of two points in the set, P_{new} is the 3-D position of P after filtering, $N(P)$ is a 3-D neighborhood of P, and $\overrightarrow{OP}$ and $\overrightarrow{OP'_{\text{new}}}$ are two vectors from the camera center to P and P'_{new}, respectively, as show

Fig. 14. Depth map accuracy evaluation. (a) Testing images. (b) Ground truth depth maps from the 3-D model. (c) Recovered depth maps obtained by our algorithm. The root mean squared error (RMSE) for each depth map is also shown.

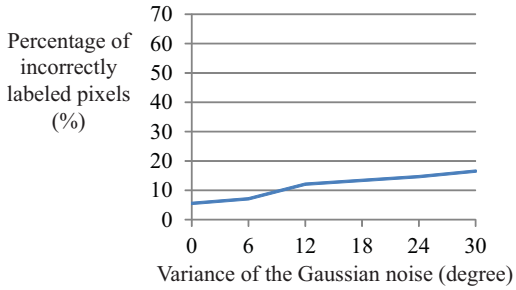


Fig. 15. Labeling accuracy of our algorithm when the vanishing points are incorrectly marked.

in Fig. 9. For each point P, we first calculate a weighted average position P'_{new} with the points around P. To ensure that the new position P_{new} has the same projection as P on the image plane, we use the projection of P'_{new} on the line $\overrightarrow{OP}$ as the new position.

After missing part recovery and surface smoothing, the final 3-D mesh model is generated from the 3-D point cloud using a simple scheme in [27].

IV. Experiments

In this section, we show extensive experiments to verify the performance of our algorithm. All the testing images are reflectional-symmetric objects, and are collected from Google Images.

The first experiment is carried out on six objects from [19]. These objects have less texture and simpler topology, and both the algorithms in this paper and in [19] perform well on these objects, as shown in Fig. 10. For each input object in Fig. 10(a), the user first marks the symmetric line pairs and depth discontinuity regions on the images. Each symmetric line pair is marked by two lines of the same color [see Fig. 10(b)]. Depth discontinuity regions are marked by green strokes [see Fig. 10(c)]. The second and third objects do not have depth discontinuity regions. The labeling results are shown in Fig. 10(d), and the reconstruction results are given in Fig. 10(e)–(g). Fig. 10(e) shows the depth of each pixel after pixel labeling, and Fig. 10(f) and (g) shows two views of each recovered 3-D model. For comparison, we also show the recovered depth maps obtained by the algorithm in [19] in Fig. 10(h).

The second experiment is conducted on another three objects from [19]. These objects have the most complex topology in [19]. The recovered depth maps of the algorithm in this paper and the algorithm in [19] are shown in Fig. 11(c) and (d), respectively. Note that there are errors in the depth maps obtained by the algorithm in [19] (see the red circles), which are caused by the ambiguity discussed in [19]. These errors disappear in the depth maps recovered by the new algorithm because the user interaction helps to avoid this ambiguity.

Fig. 12 shows the third experiment on five more complex objects. The recovered depth maps are given in Fig. 12(c) and two views of each recovered 3-D model in Fig. 12(d). These objects have more complex textures than those in Figs. 10 and 11. The algorithm in [19] cannot generate a reasonable result, which can be seen by comparing Fig. 12(c) and (f). However, with the user interaction, the new algorithm can deal with them very well. To demonstrate the performance of the refinement, we also show two views of the 3-D models before the refinement in Fig. 12(d), in which some parts of the objects are missing, as indicated by the red ellipses. These missing parts are recovered after the refinement [see Fig. 12(e)].

To objectively evaluate the performance of our algorithm, we test it using two quantitative criteria. The first one is labeling accuracy. For each testing image, we manually label the plane each pixel belongs to and use these labeling results as the ground truth to evaluate labeling accuracy. The labeling accuracy is estimated by the percentage of incorrectly labeled pixels in all object pixels, which is shown in Fig. 13, where the accuracy of the algorithm in [19] is also given for comparison. This quantitative evaluation is consistent with the visual result. For objects with simple topology and texture (objects 1–6), both our algorithm and the algorithm in [19] have low labeling errors. For more complex objects (objects 7–14), there are significant errors obtained by the algorithm in [19], while ours still performs well.

The second criterion is the depth map accuracy. Since there are no depth maps for common photos, we instead use a 3-D model generated by some designer (from Google 3-D

Warehouse). Five testing images are generated by projecting the 3-D model onto the 2-D plane in different views, as shown in Fig. 14(a). Then for each testing image, the user marks the symmetric lines and depth discontinuity regions. The ground truth depth maps and the recovered depth maps are shown in Fig. 14(b) and (c), respectively. The RMSE for each depth map is shown in the lowest row in Fig. 14. It can be seen that the estimated depth maps are very similar to the ground truth, which demonstrate the good performance of our algorithm.

To find out how robust our algorithm is to incorrect user marks, we test it when the vanishing points are incorrectly marked. For each input testing image, the user first draws two corners (see Section III-A) and carefully adjusts the $\mathbf{v_x}$, $\mathbf{v_y}$, and $\mathbf{v_z}$ directions using the perspective graph. The vanishing points obtained from this method are considered as the ground truth. Then we simulate the marking errors by adding Gaussian noise with zero mean and different variances to the directions, from which incorrect vanishing points are calculated. Finally, the 3-D geometry is recovered using these vanishing points. The reconstruction accuracy with different marking errors is shown in Fig. 15. From it, we can see that our algorithm is not significantly affected, even with large marking errors.

Our algorithm is implemented in MATLAB combined with C++. On average, it takes 13 seconds to recover the 3-D geometry of one object on a PC with 2.4 GHz Intel Core 2 CPU. The user only needs less than 1 min to do the interaction on each image. This paper focuses on piece-wise planar or near piece-wise planar objects. Part of our future work is to deal with objects with curved surfaces, such as footballs and wheels.

V. Conclusion

In this paper, we proposed a novel 3-D modeling approach to recover 3-D geometry from a single image of a symmetric object. In our system, the user first marks symmetric lines and depth discontinuity regions on the image. Our algorithm then finds a set of 3-D planes to approximate the surface of the object. After that, the planes each pixel belongs to are detected by an optimization procedure with an MRF, and a rough 3-D model was generated based on these planes. Finally, the 3-D model was refined using symmetry information and Gaussian filtering. Experimental results on various objects showed that, with only little user interaction, our algorithm can successfully recover the 3-D model of a symmetric object from a single image. In the future, we plan to extend the current framework to curved object [28] and objects of other kinds of symmetry, such as rotational symmetry and translational symmetry.

References

[1] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *Proc. ACM SIGGRAPH Conf. Comput. Graph.*, 1999, pp. 1–8.

[2] J. Huang and B. Cowan, "Simple 3D reconstruction of single indoor image with perspective cues," in *Proc. Comput. Robot Vis.*, 2009, pp. 1–8.

[3] D. C. Lee, M. Hebert, and T. Kanade, "Geometric reasoning for single image structure recovery," in *Proc. IEEE Int. Conf. Comput. Vis.*, Jun. 2009, pp. 2136–2143.

[4] J. M. Leotta and L. J. Mundy, "Predicting high resolution image edges with a generic, adaptive, 3-D vehicle model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1311–1318.

[5] L. Sigal, A. Balan, and M. J. Black, "Combined discriminative and generative articulated pose and non-rigid shape estimation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1–8.

[6] N. Jiang, P. Tan, and L. Cheong, "Symmetric architecture modeling with a single image," in *Proc. ACM SIGGRAPH Conf. Exhibit. Asia*, 2009, pp. 1–8.

[7] F. A. van den Heuvel, "Line-photogrammetric mathematical model for the reconstruction of polyhedral objects," in *Proc. Int. Soc. Opt. Photon.*, 1999, pp. 1–12.

[8] A. Grün, "Semi-automated approaches to site recording and modeling," *Int. Arch. Photogrammetry Remote Sensing*, vol. 33, no. 5, pp. 309–318, 2000.

[9] S. El-Hakim, J. Beraldin, and F. Blais, "Critical factors and configurations for practical 3D image-based modeling," in *Proc. 3D Meas. Tech.*, 2003, pp. 1–9.

[10] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach," in *Proc. ACM SIGGRAPH Conf. Comput. Graph.*, 1996, pp. 1–9.

[11] A. François, G. Medioni, and R. Waupotitsch, "Mirror symmetry $\Rightarrow$ 2-view stereo geometry," *Image Vis. Comput.*, vol. 21, no. 2, pp. 137–143, 2003.

[12] L. Zhang, G. Dugas-Phocion, J. Samson, and S. Seitzt, "Single view modeling of free-form scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Apr. 2001, pp. 990–997.

[13] M. Prasad, A. Zisserman, and A. Fitzgibbon, "Fast and controllable 3D modelling from silhouettes," in *Proc. Annu. Conf. Eur. Assoc. Graph.*, 2005, pp. 1–4.

[14] D. Liebowitz, A. Criminisi, and A. Zisserman, "Creating architectural models from images," *Comput. Graph. Forum*, vol. 18, no. 3, pp. 39–50, 1999.

[15] D. Jelinek and C. Taylor, "Reconstruction of linearly parameterized models from single images with a camera of unknown focal length," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 7, pp. 767–773, Jul. 2001.

[16] Z. Li, J. Liu, and X. Tang, "A closed-form solution to 3D reconstruction of piecewise planar objects from single images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–6.

[17] W. Hong, A. Yang, K. Huang, and Y. Ma, "On symmetry and multiple-view geometry: Structure, pose, and calibration from a single image," *Int. J. Comput. Vis.*, vol. 60, no. 3, pp. 241–265, 2004.

[18] A. Y. Yang, K. Huang, S. Rao, W. Hong, and Y. Ma, "Symmetry-based 3-D reconstruction from perspective images," *Comput. Vis. Image Understand.*, vol. 99, no. 2, pp. 210–240, 2005.

[19] T. Xue, J. Liu, and X. Tang, "Symmetric piecewise planar object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 2577–2584.

[20] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[21] C. A. Rothwell, D. A. Forsyth, A. Zisserman, and J. L. Mundy, "Extracting projective structure from single perspective views of 3D point sets," in *Proc. IEEE Int. Conf. Comput. Vis.*, May 1993, pp. 573–582.

[22] P. J. Schneider and D. H. Eberly, *Geometric Tools for Computer Graphics*. San Mateo, CA: Morgan Kaufmann, 2002.

[23] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, 2004.

[24] H. Kong, J. Audibert, and J. Ponce, "Vanishing point detection for road detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 93–103.

[25] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[26] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.

[27] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *Int. J. Comput. Vis.*, vol. 59, no. 3, pp. 207–232, 2004.

[28] Y. Wang, Y. Chen, J. Liu, and X. Tang, "3D reconstruction of curved objects from single 2D line drawings," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1834–1841.

Tianfan Xue (S'11) received the B.E. degree in computer science and technology from Tsinghua Universtiy, Beijing, China, in 2009, and M.Phil. degree in computer vision from The Chinese University of Hong Kong, Hong Kong, in 2011.

He is currently a Research Assistant with the Department of Information Engineering, Chinese University of Hong Kong. His current research interests include computer vision and machine learning.

Jianzhuang Liu (M'02–SM'02) received the B.E. degree in telecommunications from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 1983, the M.E. degree in image processing from the Beijng University of Posts and Telecommunications, Beijing, China, in 1987, and the Ph.D. degree in computer vision from The Chinese University of Hong Kong, Hong Kong, in 1997.

He was a Faculty Member with Xidian University, Xi'an, China, from 1987 to 1994. From 1998 to 2000, he was a Research Fellow with Nanyang Technological University, Singapore. From 2000 to 2012, he was a Post-Doctoral Fellow, an Assistant Professor, and an Adjunct Associate Professor with The Chinese University of Hong Kong. He joined the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, as a Professor in 2011. He is currently a Chief Scientist with Huawei Technologies Co. Ltd., Shenzhen. He has published more than 100 papers, most of which are in prestigious journals and conferences in computer science. His current research interests include computer vision, image processing, machine learning, multimedia, and graphics.

Xiaoou Tang (S'03–M'06–SM'02–F'09) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 1990, the M.S. degree from the University of Rochester, Rochester, NY, in 1991, and the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge, in 1996.

He is a Professor with the Department of Information Engineering and an Associate Dean (Research) with the Faculty of Engineering, The Chinese University of Hong Kong, Hong Kong. He was the Group Manager with the Visual Computing Group, Microsoft Research Asia, Beijing, China, from 2005 to 2008. His current research interests include computer vision, pattern recognition, and video processing.

Dr. Tang was a Program Chair of the IEEE International Conference on Computer Vision in 2009. He is an Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE and the *International Journal of Computer Vision*. He was a recipient of the Best Paper Award from the IEEE Conference on Computer Vision and Pattern Recognition in 2009.