Recovering The Shape of Polyhedra Using Line-Drawing Analysis and Complex Reflectance Models

Ilan Shimshoni and Jean Ponce
Department of Computer Science and Beckman Institute
University of Illinois, Urbana, IL 61801

Appeared in CVPR'94.

Abstract

Following Sugihara [23], we represent the geometric constraints imposed by the line-drawing of a polyhedron as a set of linear equalities and inequalities. Unlike him, we explicitly take into account the uncertainty in vertex position. This allows us to circumvent the superstrictness of the constraints without deleting any of them. For a given error bound, deciding whether a line-drawing is the correct projection of a polyhedron is reduced to linear programming, and 3D shape recovery is reduced to optimization under linear constraints. Our method can be used for recovering the shape of any polyhedral object whose reflectance can be modelled accurately. We have implemented it for the Lambertian model and the Lambertian model with interreflections. We present results obtained using real images.

1 Introduction

We address the problem of deciding whether a labelled line-drawing [2, 5, 7, 9, 18, 25] represents a correct picture of some polyhedron, and, in this case, reconstructing a polyhedron that projects onto the line-drawing [4, 10, 12, 13, 15, 23, 24] using shading information. We propose an approach that combines Sugihara's algebraic characterization of line-drawings of polyhedra [21, 23, 24] with gradient space constraints [4, 10, 12, 13, 15], shape from shading techniques [6, 17], and radiosity methods [3]. Its novelty is that it explicitly takes into account uncertainty in vertex position. Its main advantages are its efficiency in using linear programming to reject incorrect scene interpretations and its ability to perform shape recovery using real images of objects with realistic complex reflectance models [6, 17].

In [23, 24], Sugihara proved that a labelled line-drawing correctly represents the projection of some polyhedron if and only if the linear constraints:

$$\begin{cases} A\mathbf{w} = 0, \\ B\mathbf{w} \geq 0, \end{cases} \tag{1}$$

admit a solution. These are the *fundamental equations* associated with the line-drawing. The matrices A and B are derived from the positions of the vertices, the incidence relations between vertices, edges, and faces, and the edge labels. The unknowns $\mathbf{w}$ capture the 3D shape of the observed polyhedron.

Thus, linear programming can be used to determine whether a line-drawing is "correct", i.e., whether there exists some polyhedron projecting onto it [24]. Due to the loss of depth information in the imaging process, a correct line-drawing admits an infinite number of interpretations. In other words, if there exists a solution $\mathbf{w}$ to (1), then there exists an infinite number of distinct solutions. However, when additional cues such as intensity are available, it is possible to select a unique 3D interpretation $\mathbf{w}$ through non-linear optimization under the linear constraints (1) [23].

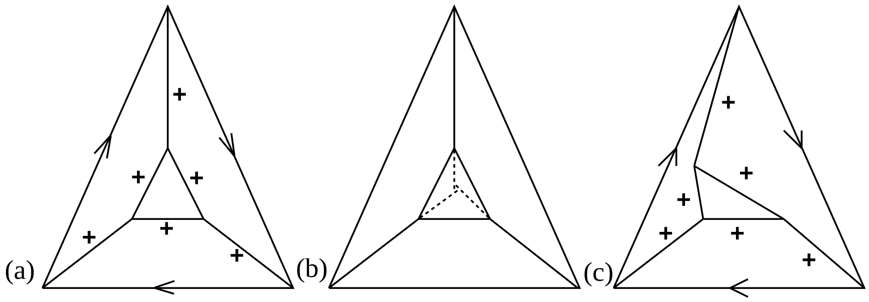


Figure 1. (a) A "correct looking" labelled line-drawing. (b) Superstrictness. (c) An incorrect line-drawing.

Sugihara's approach is rigorous and elegant. It has been called "the final breakthrough in quantitative (line-drawing) analysis" [8]. However, as remarked by Sugihara himself, a problem with his method is that condition (1) is too strict: minute perturbations of vertex positions can make a line-drawing incorrect. For example, the labelled line-drawing in Fig. 1(a) appears to correctly represent a truncated pyramid seen from above and floating in space. Closer inspection reveals that, due to error in vertex position, the three edges that (once extended) should intersect at the pyramid's apex actually do not meet (Fig. 1(b)).

This "superstrictness" problem is in fact common to most quantitative approaches to line-drawing analysis, including those based on gradient space [4]. To avoid it, Sugihara proposed to detect and delete the constraints that lead to superstrictness by using the purely combinatorial notion of *position-free incidence structures* [23]. This, unfortunately, leads to a new difficulty: The remaining conditions may not be strict enough, and incorrect line-drawings such as the one in Fig. 1(c) may be classified as correct. In addition, the missing constraints introduce gaps in the recovered surfaces [23].

Here, we avoid superstrictness by explicitly accounting for uncertainty in vertex position. Unlike Sugihara, we do not eliminate constraints that lead to a superstrict set of equations. Instead, we explicitly introduce uncertainty in these constraints. A linear form is obtained by examining the constraints imposed by edges in gradient space [4, 10, 12, 13, 15], yielding a system of equalities and inequalities similar to (1). A necessary condition for a line-drawing to be the correct projection of a polyhedron is that this system admits a solution. It can be tested, as before, through linear programming. 3D shape recovery is again reduced to optimization under linear constraints.

This basic method was introduced in [19] where we presented methods based on linear programming for rejecting incorrect line-drawings. In this paper, we refine our original method and focus on the recovery of 3D shape using complex reflectance models. We have implemented our method, using as input the imperfect results of edge and corner detection on real images of polyhedra. Although traditional approaches often use the Lambertian reflectance model for shape recovery, we can in principle recover the shape of any object whose reflectance can be accurately modelled, using graphics rendering algorithms. To demonstrate this principle we have implemented the method for the standard Lambertian model and for the Lambertian model with interreflections [3, 6, 17].

2 Background

2.1 Notations

We consider a labelled line-drawing, represented by a set of faces, vertices, labelled edges, and their incidence relations. Orthographic projection is assumed. The image is in the x, y plane, and the projection direction is along the z axis.

A face f is represented by its equation:

$$z = px + qy + r, \tag{2}$$

and the gradient vector (p, q) is denoted by $\mathbf{g}$.

A vertex v is represented by its image position (x, y) and its (unknown) depth z. In the presence of uncertainty, the actual position (x, y) is not known exactly. Instead, it is related to the measured image position $(\tilde{x}, \tilde{y})$ through a perturbation (μ, ν), with the constraints:

$$\begin{cases} x = \tilde{x} + \mu, & |\mu| \leq \epsilon, \\ y = \tilde{y} + \nu, & |\nu| \leq \epsilon. \end{cases} \tag{3}$$

In other words, the actual position is within some small rectangle of side 2ϵ from the measured position.

An edge e is represented by its endpoints v_i, v_j and its adjacent faces f_k, f_l. It is oriented from v_i to v_j, and f_k (resp. f_l) lies on its right (resp. its left). In addition, each edge is labelled as convex (+ label), concave ($-$), or occluding ($\leftarrow$ or $\rightarrow$). For occluding edges, the occluding face lies on the right when one looks in the direction of the arrow label, and the other face may be nil.

2.2 Sugihara's Fundamental Equations

Assume for the moment that the image coordinates of the vertices are known perfectly. Consider the line-drawing in Fig. 2(a). Faces f_1 and f_2 meet along concave edge e_1. Vertex v_1 lies on e_1, and thus on f_1, f_2, while v_2 lies on f_2 only. Let the equation of f_i be $z = p_i x + q_i y + r_i$ and the coordinates of v_j be (x_j, y_j, z_j), we have the following equality constraints:

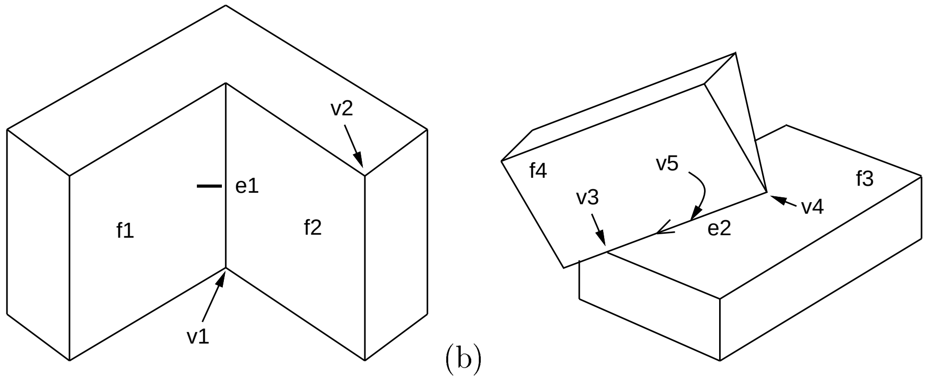
$$\begin{cases} z_1 = p_1 x_1 + q_1 y_1 + r_1, \\ z_1 = p_2 x_1 + q_2 y_1 + r_2, \\ z_2 = p_2 x_2 + q_2 y_2 + r_2. \end{cases} \tag{4}$$

Since e_1 is concave, v_2 must lie *above* the plane of f_1, yielding the inequality constraint:

$$z_2 > p_1 x_2 + q_1 y_2 + r_1. \tag{5}$$

Consider now the line-drawing in Fig. 2(b). Face f_4 occludes f_3 along e_2. Let v_3 and v_4 be two consecutive vertices along e_2, and v_5 be their mid-point, we have the following inequalities:

$$\begin{cases} z_3 \geq p_3 x_3 + q_3 y_3 + r_3, \\ z_4 \geq p_3 x_4 + q_3 y_4 + r_3, \\ z_5 > p_3 x_5 + q_3 y_5 + r_3. \end{cases} \tag{6}$$



(a) (b)

Figure 2. Line-drawings and constraints.

Equality is allowed in the first two inequalities, but not in the third one. This is because the occluding edge e_2 can touch f_3 at some point, but not at every point (otherwise it would be a concave edge) [23]. Three equality constraints should be added, corresponding to the fact that v_3, v_4, v_5 belong to f_4.

The above constraints are linear in the unknowns p_i, q_i, r_i, z_j. Sugihara gathered all such constraints into (1), where A and B are matrices whose coefficients are functions of the vertex coordinates x_j, y_j, and $\mathbf{w}$ is obtained by concatenating the unknowns for all faces and vertices. He proved that a necessary and sufficient condition for a line-drawing to correctly represent the projection of a polyhedron is that these equations admit a solution.

In the next section, we will derive similar constraints in the case where the x, y coordinates of the vertices are not known exactly.

3 Constraints under Uncertainty

We first examine the constraints imposed by incident vertices and faces, and then derive gradient space constraints associated with convex and concave edges. More details can be found in [19].

3.1 Vertex Constraints

Writing that a vertex v_i lies on two faces f_k and f_l and eliminating the unknown depth z_i yields:

$$(p_k - p_l)x_i + (q_k - q_l)y_i + (r_k - r_l) = 0, \tag{7}$$

which can be rewritten as:

$$(p_k - p_l)\tilde{x}_i + (q_k - q_l)\tilde{y}_i + (r_k - r_l) + (p_k - p_l)\mu_i + (q_k - q_l)\nu_i = 0. \tag{8}$$

The quadratic terms can be eliminated by introducing the new variables $a_{ki} = p_k\mu_i, b_{ki} = q_k\nu_i$:

$$(p_k - p_l)\tilde{x}_i + (q_k - q_l)\tilde{y}_i + (r_k - r_l) + (a_{ki} - a_{li}) + (b_{ki} - b_{li}) = 0. \tag{9}$$

By (3) a_{ki}, b_{ki} satisfy non-linear constraints:

$$\begin{cases} |a_{ki}| \leq \epsilon |p_k|, \\ |b_{ki}| \leq \epsilon |q_k|. \end{cases} \tag{10}$$

A similar derivation gives the following constraint for occluding edges:

$$(p_k - p_l)\tilde{x}_i + (q_k - q_l)\tilde{y}_i + (r_k - r_l) + (a_{ki} - a_{li}) + (b_{ki} - b_{li}) > 0. \tag{11}$$

The next subsection shows how to use gradient space constraints to eliminate the four quadratic constraints.

3.2 Edge Constraints

Consider an edge e with extremities v_i, v_j, and two adjacent faces f_k, f_l. Assume that e is either convex or concave. Writing that v_i and v_j belong to f_k and f_l, and eliminating the unknown depths z_i, z_j yields:

$$(\mathbf{g}_k - \mathbf{g}_l) \cdot \mathbf{t}_e = 0, \tag{12}$$

where $\mathbf{t}_e$ is the vector joining the projections of the vertices v_i and v_j into the image plane. This is the well known constraint imposed by an edge on the gradients of the adjacent faces [13].

Observe that $\mathbf{t}_e$ is necessarily contained in the double cone joining the uncertainty regions centered in $\tilde{\mathbf{x}}_i = (\tilde{x}_i, \tilde{y}_i)$ and $\tilde{\mathbf{x}}_j = (\tilde{x}_j, \tilde{y}_j)$ (Fig. 3).

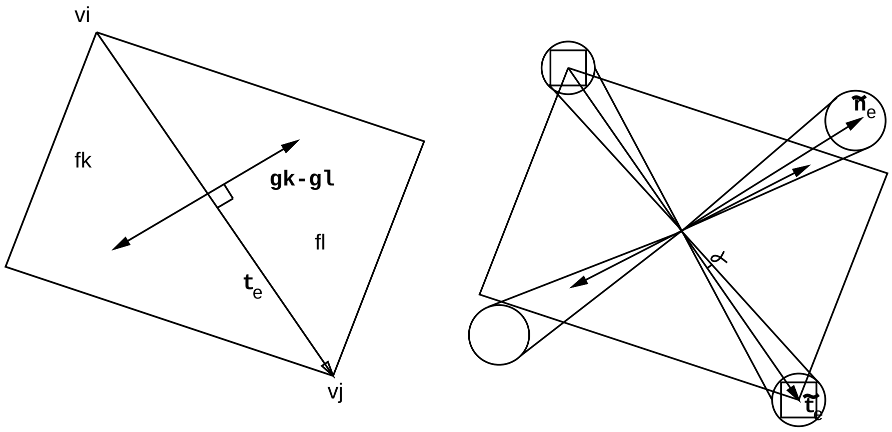


Figure 3. Edge constraints. The double arrow indicates that the orientation of $\mathbf{g}_k - \mathbf{g}_l$ is unknown.

Let $\tilde{\mathbf{t}}_e = \tilde{\mathbf{x}}_j - \tilde{\mathbf{x}}_i$ be the axis of this cone, α be its half-angle, and let $\tilde{\mathbf{n}}_e$ be the vector obtained by rotating $\tilde{\mathbf{t}}_e$ 90° counterclockwise. Since $\mathbf{g}_k - \mathbf{g}_l$ and $\mathbf{t}_e$ are orthogonal, it follows that $\mathbf{g}_k - \mathbf{g}_l$ lies in the double-cone centered on $\tilde{\mathbf{n}}_e$ with half-angle α, i.e.,

$$|(\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{t}}_e| \leq |(\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{n}}_e| \tan\alpha. \tag{13}$$

This condition is not linear. Next, we show that the sign of $(\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{n}}_e$ is actually known, which brings us back to the realm of linear algebra.

We have not used the convexity (or concavity) of the edge yet. Suppose e is convex, let $\mathbf{n}_e$ be the real direction of its normal, and z_l and z_k be the points where the line parallel to the z axis in $\mathbf{x}_i + \mathbf{n}_e$ intersects the planes of f_l and f_k (Fig. 4).

Since e is convex, we know that $z_k > z_l$, yielding, after some algebraic manipulation:

$$(\mathbf{g}_k - \mathbf{g}_l) \cdot \mathbf{n}_e > 0, \tag{14}$$

which determines the side of the double-cone where $\mathbf{g}_k - \mathbf{g}_l$ lies (Fig. 5). This is again a well known constraint [13]. We already know that $\mathbf{g}_k - \mathbf{g}_l$ and $\mathbf{n}_e$ are aligned, so:

$$(\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{n}}_e > 0, \tag{15}$$

except in the degenerate case where the two vertices are so close to each other as to make $\alpha = \pi/2$.

Combining (13) and (15) yields a *linear* constraint:

$$|(\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{t}}_e| \leq ((\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{n}}_e) \tan\alpha. \tag{16}$$

The analysis applies to concave edges by inverting the orientation of $\mathbf{g}_k - \mathbf{g}_l$ (Fig. 5).

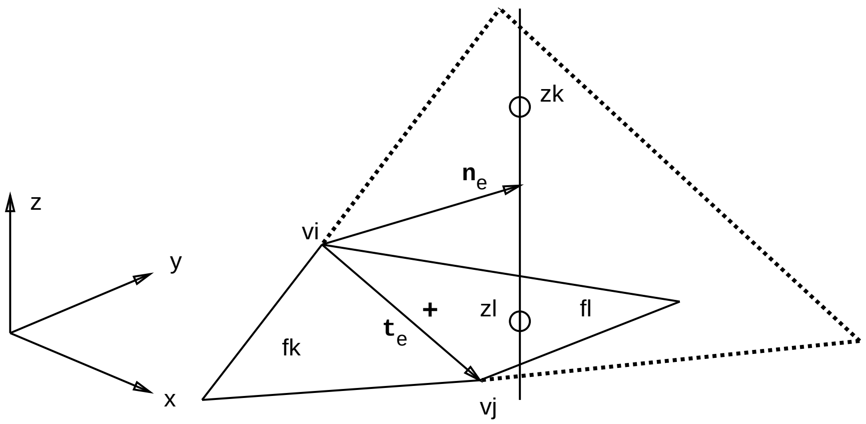


Figure 4. The geometric constraints imposed by a convex edge.

When the double cone centered in $\tilde{\mathbf{n}}_e$ does not contain the x axis, (16) determines the sign $s = \mp 1$ of $p_k - p_l$, yielding the *linear* constraint:

$$|a_{ki} - a_{li}| \leq \epsilon s(p_k - p_l). \tag{17}$$

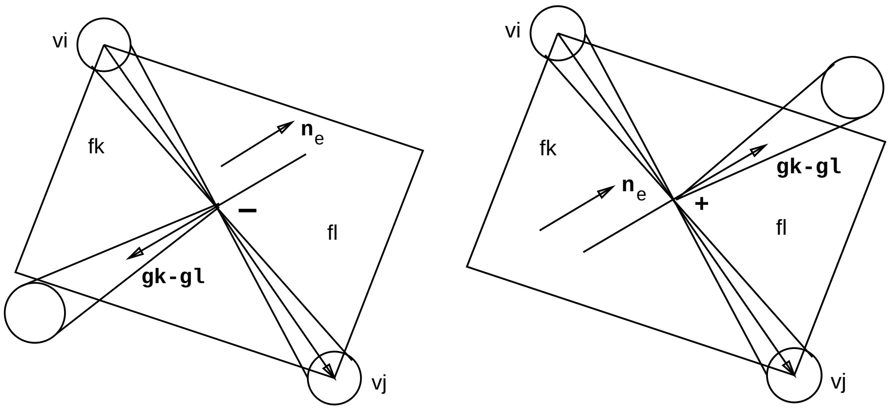


Figure 5. Constraints on the direction of $\mathbf{g}_k - \mathbf{g}_l$ imposed by concave (left) and convex (right) edges.

Similarly, when the double cone does not contain the y axis, (16) determines the sign s of $q_k - q_l$, yielding:

$$|b_{ki} - b_{li}| \leq \epsilon s(q_k - q_l). \tag{18}$$

In general, a convex or concave edge provides at least one of these two constraints for each of its extremities.

3.3 Remarks

We have shown that, even in the presence of uncertainty, a labelled line-drawing can be characterized by a set of linear constraints. Let us close this section with a few remarks. First, one of the variables r_k can be set to some arbitrary value since absolute depth cannot be recovered under orthographic projection. The variables z_i can be recovered from the other variables. Second, (11) and (15) contain strict inequalities, which is normally a difficulty for linear programming. As remarked by Sugihara [24], all the constraints are homogeneous, and strict inequalities with a zero constant term can be replaced by non-strict inequalities with an arbitrary positive constant term. Third, these constraints only provide a necessary condition for correctness. In particular, the signs of $p_k - p_l$ and $q_k - q_l$ cannot always be determined.

4 3D Shape Recovery

4.1 Principle

Like Sugihara [23], we formulate 3D shape recovery as an optimization problem. Consider a set of visual cues, such as intensity or texture. Let $\mathbf{w}$ denote the unknown geometric parameters, $\mathbf{l}$ denote the other scene parameters (e.g., light source direction and intensity, surface albedo), I_k denote the observed value of the k^{th} cue (e.g., the intensity of face f_k), and $J_k(\mathbf{w}, \mathbf{l})$ denote the value of this cue that would be observed if the actual scene parameters were $\mathbf{w}$ and $\mathbf{l}$. Recovering $\mathbf{w}$ and $\mathbf{l}$ amounts to minimizing:

$$\sum_k (I_k - J_k(\mathbf{w}, \mathbf{l}))^2 \tag{19}$$

under the linear constraints derived in Sect. 3:

$$\begin{cases} A\mathbf{w} = 0, \\ B\mathbf{w} \geq 0. \end{cases} \tag{20}$$

This is what Sugihara calls a quadric error minimization, a term we find a bit misleading since the error term is in general *not* quadratic in the unknown parameters. This is unfortunate since quadratic minimization under linear constraints can be solved exactly.

To solve the constrained minimization problem, we first use Gaussian elimination to delete all linear equalities and determine the values of a corresponding number of variables as a linear function of the remaining ones. Among the possible choices, we choose the subset of variables which yields the best numerical stability. This is done by finding which n-tuple of standard basis vectors produce a square matrix with the highest possible condition number when added to matrix A.

We solve our constrained optimization problem using an active set method [14], which reduces the original problem to the *unconstrained* minimization of:

$$\sum_k (I_k - J_k(\mathbf{w}, \mathbf{l}))^2 + \sum_j (\min\{0, B_j \mathbf{w}\})^2, \tag{21}$$

where B_j denotes the j^{th} row of B. From the way the problem is stated, it is clear that any model J_k which can recover I_k accurately when given the scene parameters $\mathbf{w}$ and $\mathbf{l}$ can be used to recover the shape information and the scene parameters. Therefore models like the ones used in computer graphics to render physically accurate images can be substituted for J_k in the formalization of the problem.

Because optimization algorithms may fall into local minima a method must be devised to give the algorithm a starting position as close as possible to the optimal one. The method we have chosen is to divide the algorithm into two stages. In the first one we run the optimization algorithm using a simplified model (e.g., disregarding the effects of interreflections on the image). Such a model will have fewer unknown parameters than the full model or make it easier to compute $J_k(\mathbf{w}, \mathbf{l})$, but it may give less accurate results. We run the optimization using the simple model with various starting positions. In the second stage we use the best result of the first stage as the starting position for optimization using the elaborate model. For complex reflectance models we generalized this method for more than two stages, where the result of each stage is the starting position of the next stage.

4.2 Lambertian Model

In order to demonstrate the applicability of our method we have implemented shape recovery from shading information using two reflectance models: the pure Lambertian model and the Lambertian model with interreflections [22].

Under the Lambertian model we assume that there is a distant point light source whose direction $\mathbf{s}$ and intensity s are unknown and suppose that the observed object has a constant (but unknown) albedo ρ. Our method differs from classical shape from shading techniques [11] in which these parameters are assumed to be known (on the other hand those methods do not assume that the object is polyhedral). Photometric stereo [26] does not require the albedo to be constant or that the object be polyhedral but requires the direction and intensity of the light sources and more than one image.

The k^{th} cue I_k is the intensity measured on the k^{th} face. We compute J_k as:

$$J_k(\mathbf{w}, \mathbf{l}) = \frac{s\rho}{\pi}\mathbf{n}_k \cdot \mathbf{s} \tag{22}$$

where $\mathbf{n}_k = \frac{(-p_k, -q_k, 1)^T}{\sqrt{(p_k^2+q_k^2+1)}}$ is the normal to face number k and the scene parameters $\mathbf{l}$ are $\mathbf{s}$,s, and ρ. In this case we cannot decouple ρ from the intensity s of the light source, therefore we recover their product.

For this reflectance model we have two stages of optimization, using the result of the first stage as the starting position of the second stage. The simplified model we use in the first stage disregards the uncertainty in vertex position. This reduces the number of unknowns considerably. In the case of superstrictness, like Sugihara [23], we delete the corresponding constraints. In the second stage we take into account all the constraints and the uncertainty in vertex position.

4.3 Lambertian Model With Interreflections

Radiosity methods are used in computer graphics to compute the interreflections of visible light and to render physically-accurate images of objects [3, 22]. They have also been used in computer vision to recover the shape of objects from images with interreflections [6, 17]. In [17] several images are required and light source parameters must be known but the albedo is unknown and does not have to be constant, and the shape of the recovered object does not have to be polyhedral.

Under the interreflection model the total irradiance $E(\mathbf{x})$ at point $\mathbf{x}$ is expressed as the sum of the irradiance $E_s(\mathbf{x})$ due to the light source and the irradiance due to all other points on the surface which can see $\mathbf{x}$. The radiance L at the point $\mathbf{x}$ is related to its irradiance E by:

$$L(\mathbf{x}) = \frac{\rho(\mathbf{x})}{\pi}E(\mathbf{x}) \tag{23}$$

where $\rho(\mathbf{x})/\pi$ is the bi-directional reflectance distribution function for a Lambertian surface. The geometric factor (the form factor) which relates the irradiance $E(\mathbf{x})$ due to the radiance of the point $\mathbf{x}'$ to the radiance of $\mathbf{x}'$ is:

$$F(\mathbf{x}, \mathbf{x}') = \mathrm{View}(\mathbf{x}, \mathbf{x}')\frac{[\mathbf{n} \cdot (\mathbf{x}' - \mathbf{x})][\mathbf{n}' \cdot (\mathbf{x} - \mathbf{x}')]}{|\mathbf{x}' - \mathbf{x}|^4}. \tag{24}$$

Here, $\mathrm{View}(\mathbf{x}, \mathbf{x}')$ equals one when $\mathbf{x}$ and $\mathbf{x}'$ can see each other, and therefore can illuminate each other, and zero otherwise. Thus the irradiance $E(\mathbf{x})$ of the surface element dA' due to the radiance of the point $\mathbf{x}'$ is:

$$E(\mathbf{x}) = F(\mathbf{x}, \mathbf{x}')L(\mathbf{x}')dA'. \tag{25}$$

From equations (23) and (25), we obtain:

$$L(\mathbf{x}) = L_s(\mathbf{x}) + \frac{\rho}{\pi}\int F(\mathbf{x}, \mathbf{x}')L(\mathbf{x}')dA'. \tag{26}$$

We use (26) to compute the intensity of the light reflected by the pixels in the image. Each pixel which is a rectangle in the image whose area is D is actually a parallelogram in space whose area is $D/(\mathbf{v} \cdot \mathbf{n})$ where $\mathbf{v}$ is the viewing direction. We assume that each pixel has uniform intensity. Applying (26) at a pixel P we obtain:

$$L(P) = s\frac{\rho}{\pi}\mathbf{n}_P \cdot \mathbf{s} + \frac{\rho[\mathbf{v} \cdot \mathbf{n}_P]}{D\pi} \int_{\mathbf{x} \in P} \sum_{P' \neq P} L(P') \int_{\mathbf{x}' \in P'} F(\mathbf{x}, \mathbf{x}') dA' dA \qquad (27)$$

where $L(P)$ is the intensity of light radiated from pixel P and $\mathbf{n}_P$ is the normal to the surface at P.

When the distance between P and the pixels it can "see" is large compared with the size of P, (27) can be approximated by:

$$L(P) = s\frac{\rho}{\pi}\mathbf{n}_P \cdot \mathbf{s} + \frac{\rho}{\pi} \sum_{P' \neq P} \frac{F(\mathbf{x}_P, \mathbf{x}_{P'}) D \; L(P')}{\mathbf{v} \cdot \mathbf{n}_{P'}} \qquad (28)$$

where $\mathbf{x}_P$ is the center of pixel P. For pixels which are close to a concave edge, we must use (27).

To recover the shape of an object we choose a random subset of the pixels in the image. When computing $J_k(\mathbf{w}, \mathbf{l})$ for a certain pixel P we compute $L(P)$ assuming that the intensity $L(P')$ of every other pixel P' in the image is as was measured in the image. When rendering an image we cannot rely on the intensity values of pixels in the image, therefore we compute $L(P)$ iteratively for all the pixels in the image:

$$L_{i+1}(P) = s\frac{\rho}{\pi}\mathbf{n}_P \cdot \mathbf{s} + \frac{\rho[\mathbf{v} \cdot \mathbf{n}_P]}{D\pi} \int_{\mathbf{x} \in P} \sum_{P' \neq P} L_i(P') \int_{\mathbf{x}' \in P'} F(\mathbf{x}, \mathbf{x}') dA' dA \qquad (29)$$

where $L_0(P) = s\frac{\rho}{\pi}\mathbf{n}_P \cdot \mathbf{s}$. $L_i(P)$ is the amount of light reflected from pixel P due to rays interreflected up to i times. The iterations are continued until $L_{i+1}(P) \approx L_i(P)$ for all pixels.

For this reflectance model we have three stages of optimization, using the result of each stage as the starting position of the next stage. In the first stage of the optimization we disregard interreflections and the uncertainty in vertex position. For each face f_k we choose the intensity I_k as the minimum intensity value of a pixel of that face. As interreflections can only add to the intensity of a pixel the minimum value should be as close as possible to the light reflected only due to the light source. In the second stage we take into account interreflections. We use (27) or (28) to compute $J_k(\mathbf{w}, \mathbf{l})$ for the selected pixels. We finally perform a third optimization stage which takes into account vertex uncertainty.

5 Experimental Results

We have implemented the approach proposed in Sect. 3 to test the correctness of a line-drawing in the presence of uncertainty using linear programming. In this case, linear programming is not used for optimization, but rather for deciding whether there exists a point satisfying these constraints. We present results classifying line-drawings as legal or illegal for different uncertainty bounds. We also present our implementation of 3D shape recovery from intensity data using the method described in Sect. 4. We have applied an active set optimization method to the solution space obtained through performing Gaussian elimination on the linear equality constraints. However, no special techniques were used exploiting the linearity of the inequalities. This did not degrade the performance of the algorithm because usually a point was found in the early stages of the algorithm that satisfied those constraints, and the main effort was put into computing and minimizing the objective function for points that satisfied all of the inequalities. The algorithm has been implemented in C using the simplex algorithm from [20] and the Levenberg-Marquardt procedure of the MINPACK library [16]. Experiments were run on a SUN SPARC Station.

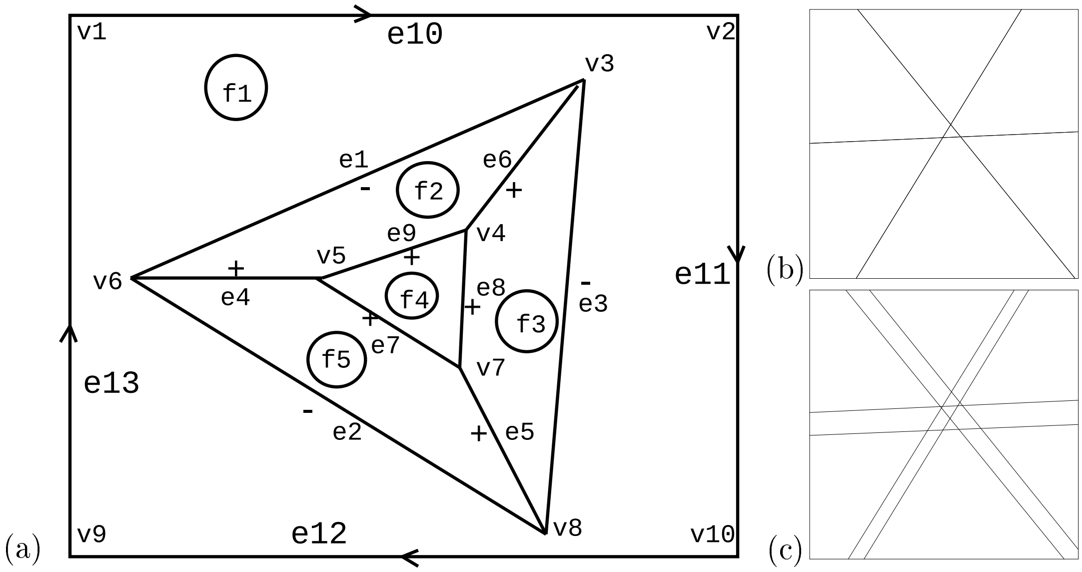


Figure 6. (a) A line-drawing from [23]. Close-ups of the region where the support lines of e_4, e_5, e_6 should intersect. (b) Incorrect line-drawing with $\epsilon = 0$. (c) Correct line-drawing with $\epsilon = 0.002$.

5.1 Line-Drawing Analysis

Figure 6(a) shows a sample line-drawing, reproduced from [23]. Of particular interest are the three edges e_4, e_5, e_6 converging toward the center of this truncated pyramid. With the nominal values given in [23], and an $\epsilon = 0$ uncertainty bound, this "correct looking" line-drawing is classified as incorrect by our program (Fig. 6(b)). The corresponding linear programming problem has 51 variables and 84 constraints. The running time of the program is 3.6s. Next, we set the uncertainty bound to $\epsilon = 0.002$. This time, the line-drawing is classified as correct by our program, as confirmed by Fig. 6(c), where the double cones corresponding to the three edges have been drawn and are shown to intersect. The number of variables is the same as before, but there are 120 constraints, and the running time of the program is 8.2s. The difference in the number of constraints comes from the fact that, for $\epsilon = 0$, each pair of inequalities corresponding to (16-18) can be replaced by a single equality.

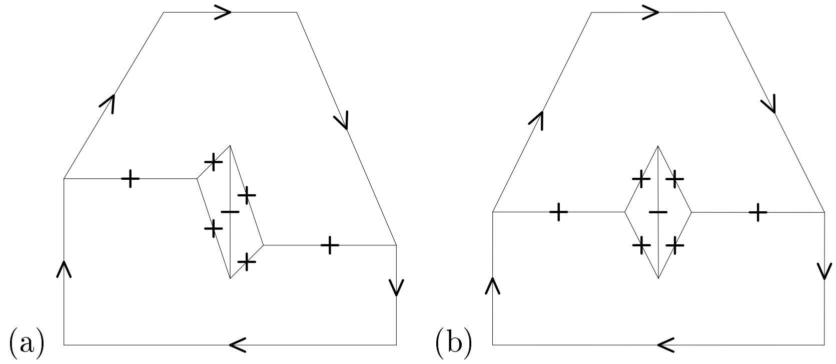


Figure 7. (a) An incorrect line-drawing. (b) A correct line-drawing.

Figure 7(a) is reproduced from [1]. It is an incorrect line-drawing, as verified by our program. In this case, the corresponding linear programming problem has 44 variables and 80 constraints, with a total running time of 4.0s. Figure 7(b) shows a correct version of the same line-drawing, with a random perturbation of range 0.01 in vertex position. Note that this perturbation is not visible in this picture where the longest edge has length 10. The line-drawing is correctly classified by our program with $\epsilon = 0.01$. The number of variables and constraints is the same as before, and the running time is 3.9s.

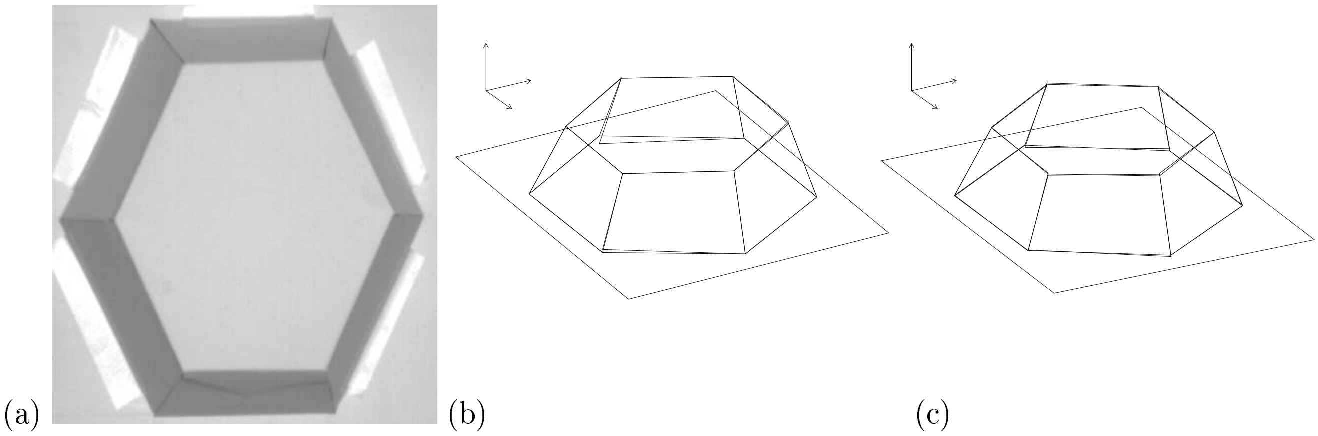


Figure 8. Results of the Lambertian model algorithm: (a) the input image; (b) first stage ; (c) final stage of shape recovery.

5.2 Lambertian Model

We have tested our shape recovery algorithm on several real images. Figure 8(a) shows the image of an object made of flat white paper. Edge detection was run on the image and a "clean" line-drawing was extracted by hand from the results. Figure 8(b) shows the reconstruction obtained using the first stage of the algorithm. Notice that some adjacent faces do not coincide. This is due to removal of constraints due to superstrictness. Figure 8(c) shows the final result when all constraints have been taken into account and uncertainty in vertex position is reduced to below ϵ.

5.3 Lambertian Model with Interreflections

We have also run the algorithm on a number of objects which have concave edges causing interreflections between adjacent faces. Figure 9(a) shows an image of an L-shaped wooden object painted with flat white paint. Figure 9(b) shows the intensity reconstruction of the image using the shape and scene parameters recovered by the optimization algorithm. Figure 9(c) shows the isointensity lines in the original image, and Figure 9(d) shows those lines in the rendered image. In Figure 9(e) the intensity of a cross-section of the two images is plotted, and Figure 9(f) shows two views of the recovered object.

We have also run the algorithm on two images of a styrofoam object (once again painted with flat white paint) which is used for packing computers. The results for the first image (Figure 10) are not quite as good as the results for the second image (Figure 11), even though for both images the results of the intensity reconstruction are good despite much texture due to the grainy appearance of styrofoam. In order to obtain a quantitative evaluation of our results we compared the angles between adjacent faces in the original objects (which we measured by hand) and in the recovered objects. For the object in Figure 9 the error is at most $5°$. For the object in Figure 11 the error is between $5°$ and $25°$. For the object in Figure 10 the recovered object is not as close to the original object, and the error is at most $65°$ even though the intensity reconstruction is good (Figure 10(e)). Further analysis must be performed to determine the factors that influence the quality of the results.

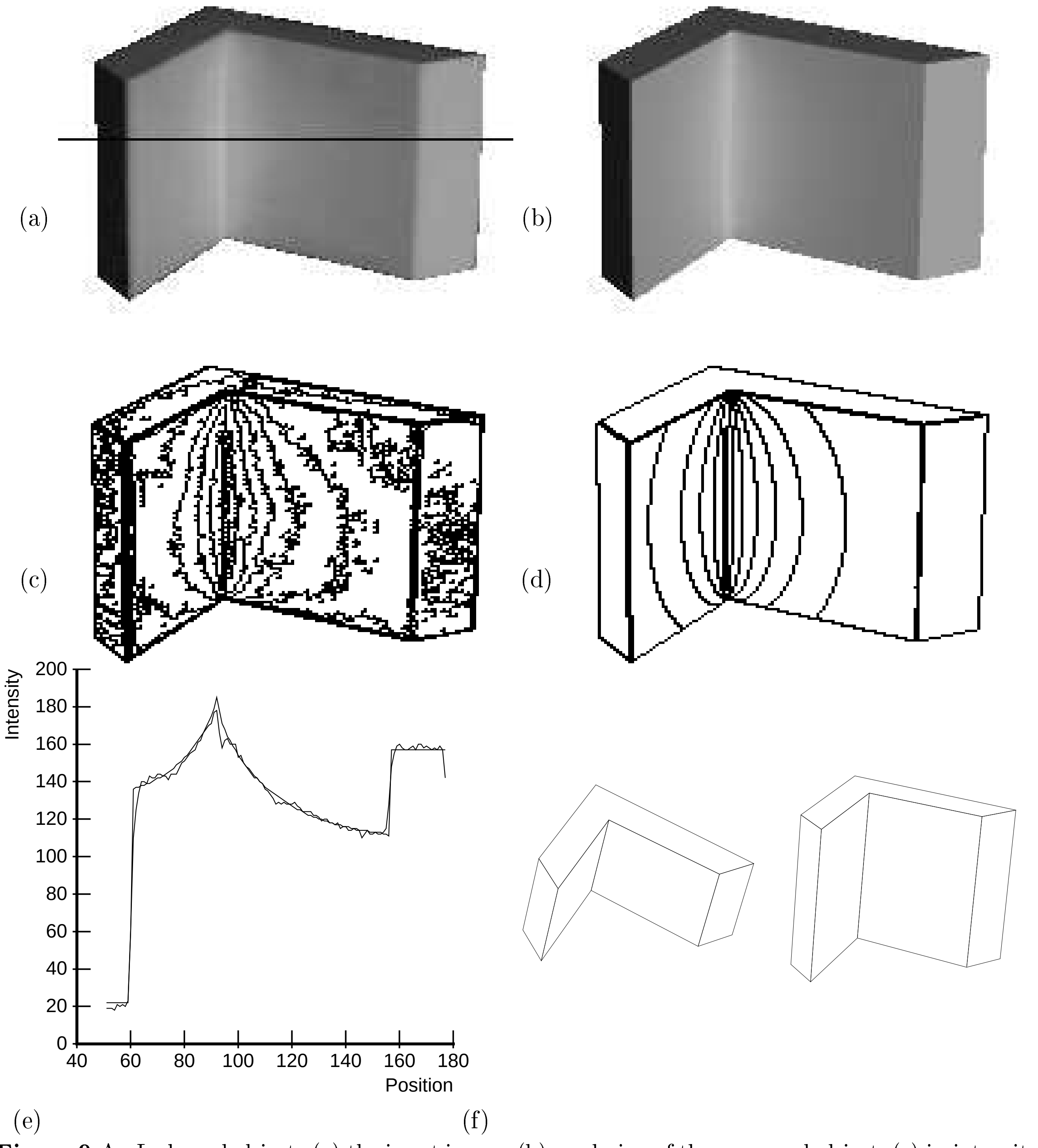


Figure 9. An L-shaped object: (a) the input image; (b) rendering of the recovered object; (c) isointensity lines in the image; (d) isointensity lines in the rendered image; (e) intensity values of the cross-section depicted by the line in (a) of the image and the rendered image; (f) views of the recovered object.

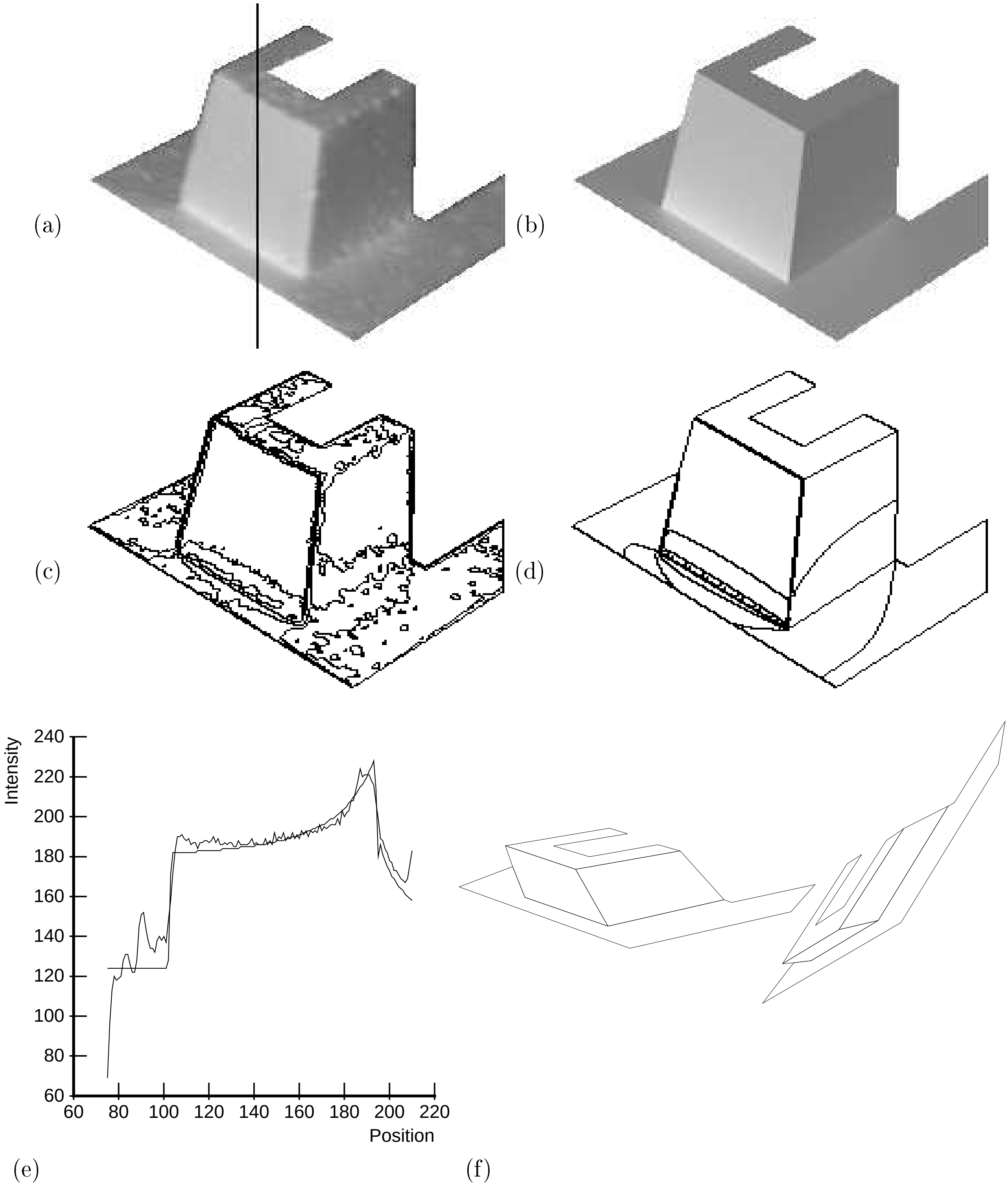


Figure 10. An image of a styrofoam object: (a) the image of the object; (b) rendering of the recovered object; (c) isointensity lines in the image; (d) isointensity lines in the rendered image; (e) intensity values of the cross-section depicted by the line in (a) of the image and the rendered image; (f) views of the recovered object.

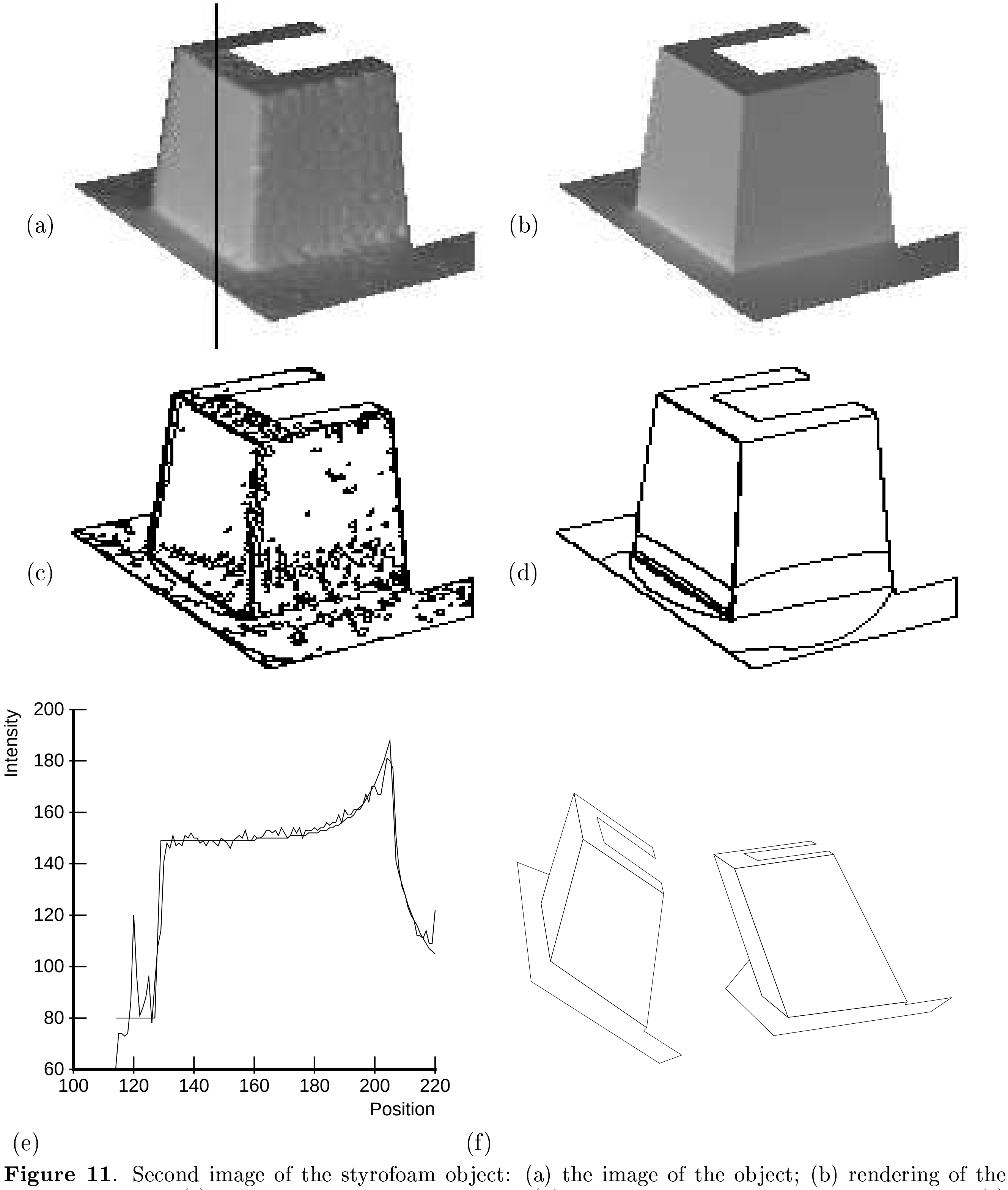


Figure 11. Second image of the styrofoam object: (a) the image of the object; (b) rendering of the recovered object; (c) isointensity lines in the image; (d) isointensity lines in the rendered image; (e) intensity values of the cross-section depicted by the line in (a) of the image and the rendered image; (f) views of the recovered object.

6 Discussion and Future Work

We have presented a new approach to 3D shape recovery from a 2D image using the geometric information contained in the line-drawing and the shading information contained in the image. The geometric constraints imposed by the line-drawing of a polyhedron are represented by a set of linear equalities and inequalities, and the uncertainty in vertex position is explicitly taken into account.

We have shown that this scheme can be used to recover the 3D structure of objects which have complex reflectance models. It can actually be used for any polyhedral object whose reflectance can be reliably modelled. This includes images with shadows, metallic objects, dielectric objects, rough and smooth objects etc... Reflectance parameters (such as albedo or roughness) and light source parameters can be recovered as part of the process.

We plan to investigate a number of extensions to this work: first, we plan to test the method on other reflectance models. Second, we plan to analyze the effects of the choice of viewing direction and light source direction on the quality of the recovery. Finally, as always in computer vision, the real challenge will be fully automating the process for real images. This will require extracting a (maybe imperfect) line-drawing from an image through edge detection, straight line and junction finding, and edge and junction labelling. It is our belief that the quantitative constraints available from partial labellings of line-drawings and shading information will help to guide this segmentation process.

References

[1] D.H. Ballard and C.M. Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[2] M.B. Clowes. On seeing things. *Artificial Intelligence*, 2(1):79–116, 1971.

[3] M. F. Cohen and D. P. Greenberg. The hemi-cube: A radiosity solution for complex environments. In *Proc. SIGGRAPH 85*, pages 31–40, San Francisco, July 1985.

[4] S.W. Draper. The use of gradient and dual space in line-drawing interpretation. *Artificial Intelligence*, 17:461–508, 1981.

[5] G. Falk. Interpretation of imperfect line data as a three-dimensional scene. *Artificial Intelligence*, 3:101–144, 1972.

[6] D. Forsyth and A. Zisserman. Reflections on shading. *IEEE Trans. Patt. Anal. Mach. Intell.*, 13(7):671–679, July 1991.

[7] A. Guzman. Computer recognition of three-dimensional objects in a visual scene. Technical Report MAC-TR-59, MIT, 1968.

[8] B.K.P. Horn. *Computer Vision.* MIT Press, Cambridge, Mass., 1986.

[9] D.A. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence*, 6:295–323, 1971.

[10] D.A. Huffman. Realizable configurations of lines in pictures of polyhedra. *Machine Intelligence*, 8:493–509, 1977.

[11] K. Ikeuchi and B.K.P. Horn. Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 17:141–184, 1981.

[12] T. Kanade. A theory of the Origami world. *Artificial Intelligence*, 13:279–311, 1980.

[13] T. Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–460, 1981.

[14] D.G. Luenberger. *Linear and nonlinear programming.* Addison-Wesley, 1984. Second edition.

[15] A.K. Mackworth. Interpreting pictures of polyhedral scenes. *Artificial Intelligence*, 4(2):121–137, June 1973.

[16] J.J. Moré, B.S. Garbow, and K.E. Hillstrom. User guide for MINPACK-1. ANL-80-74, Argonne National Laboratories, 1980.

[17] S. K. Nayar, K. Ikeuchi, and T. Kanade. Shape from interreflections. In *Proc. Int. Conf. Comp. Vision*, pages 2–11, Osaka, Japan, December 1990.

[18] V.D. Nguyen. Labeling polyhedral scenes. In *Proc. DARPA Image Understanding Workshop*, pages 1160–1165, April 1988.

[19] J. Ponce and I. Shimshoni. An algebraic approach to line-drawing analysis in the presence of uncertainty. In *IEEE Int. Conf. on Robotics and Automation*, pages 1786–1791, 1992.

[20] W. Press, B.Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1988.

[21] R. Shapira. A note on Sugihara's claim. *IEEE Trans. Patt. Anal. Mach. Intell.*, 6(1):122–123, January 1984.

[22] R. Siegal and J. R. Howell. *Thermal Radiation Heat Transfer.* Hemisphere Publishing, New York, 1992.

[23] K. Sugihara. An algebraic approach to the shape-from-image-problem. *Artificial Intelligence*, 23:59–95, 1984.

[24] K. Sugihara. A necessary and sufficient condition for a picture to represent a polyhedral scene. *IEEE Trans. Patt. Anal. Mach. Intell.*, 6(5):578–586, September 1984.

[25] D. L. Waltz. Understanding line drawings of scenes with shadows. In P.H. Winston, editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, New York, 1975.

[26] R. J. Woodham. Photometric stereo: A reflectance map technique for determining surface orientation from image intensity. *In Proc. SPIE*, 155:136–143, 1978.