Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea • May 21-26, 2001

Correcting Polyhedral Projections for Scene Reconstruction

Lluís Ros and Federico Thomas
Institut de Robòtica i Informàtica Industrial (UPC-CSIC)
Gran Capità 2-4, 08034 Barcelona, Spain
e-mail: {llros, fthomas}@iri.upc.es

Abstract

This paper presents a new algorithm for correcting incorrect projections of a polyhedral scene. Such projections arise in applications where an image of a polyhedral world is taken and its edges and vertices are extracted. Along the way, the true positions of the vertices in the 2D projection are perturbed due to digitization errors and the preprocessing. As most available algorithms for reconstructing polyhedral scenes from projections are "superstrict", they judge these noisy inputs as incorrect and fail to obtain a 3-dimensional scene from them.

The presented method overcomes this problem by moving the positions of all vertices until a very close correct projection is found. With this tool, any superstrict method for reconstructing scenes from projections is now practical, as it can be applied to the corrected projection.

1 Introduction

Consider the pictures of the plane-faced alarm devices in fig. 1. How can we tell whether such pictures actually represent the *correct* projection of a spatial object? Even more, how can we *reconstruct* all 3D objects they represent? Answering these questions and producing an algorithm able to recognize a plane-faced object from its projection, with similar results as a human gets, has been one of the goals of Computer Vision and Artificial Intelligence, with applications in Robotics, since the early seventies.

Along the years, several methods have been proposed to test the correctness of polyhedral projections and give their possible spatial reconstructions [4, 13, 17]. These tests succeed in judging as incorrect impossible figures like those in fig. 2. However, even when the projections are pictures of a real scene, the tests usually judge them as incorrect, failing to derive a spatial reconstruction. To see why, consider the examples in fig. 1-bottom, showing the projections of truncated pyramids extracted from two real scenes.

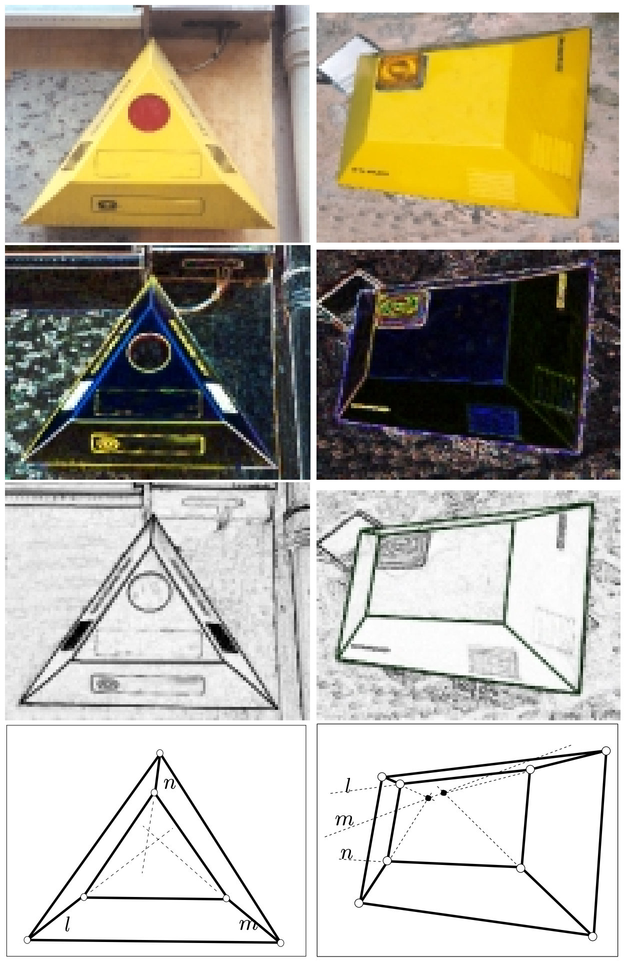

Figure 1: A picture of an alarm device (top) can be processed to detect the sharp edges, extract the straight lines, and derive a polyhedral projection (bottom) which is usually incorrect.

It can be shown that they are only correct when the three lines l, m and n meet at a common point. This is a general characteristic of most projections: they are only correct for very specific positions of the vertices, satisfying a number of *concurrence conditions* (already identified in [17] and [10]). Hence, as any geometric relationships between the vertices are lost due to digitization errors and the image processing, many existing tests are *superstrict* [13, page 111], judging a projection of a real scene as incorrect when it just deviates slightly from a correct and reconstructable configuration.

Our contribution is a new procedure to search for the closest correct projection to a given incorrect one. Such a procedure allows overcoming the superstrictness problem in an easy way: to apply a superstrict method on an incorrect projection $\mathcal{P}^{inc}$, first compute the closest correct projection $\mathcal{P}^{cor}$ to it. If the vertices on $\mathcal{P}^{cor}$ are too far from those in $\mathcal{P}^{inc}$ (according to a well-defined distance and a given tolerance), $\mathcal{P}^{inc}$ is judged as incorrect, otherwise we accept it as "practically correct" and we can start the reconstruction process from $\mathcal{P}^{cor}$.

The next section shortly reviews three classic correctness tests and shows, through an example, why they are superstrict. Then, in section 3, we compare our approach to two previous methods for overcoming the superstrictness. A rational parameterization of all correct projections for a given polyhedron is given in sections 5 and 6, which allows us to write down the correction problem as an *unconstrained* minimization of a rational function (section 4). This minimization can be tackled using a conjugate gradient method, but a good starting point for the search is needed and section 7 provides one. Finally, section 8 shows the results of the implementation, and section 9 summarizes the conclusions and points deserving further attention.

2 Superstrict Methods

We begin with a few definitions and assumptions used along the paper. To simplify, we will deal with orthogonal projections of *a single* spherical polyhedron, onto the XY plane, showing all edges (even the hidden ones). By *spherical*, we mean here that it is homeomorphic to a sphere. This is not too restrictive, and section 9 explains how to extend the results to projections of more complicated scenes, without hidden edges, several objects, and possible occlusions between them. So, we say that a projection is *correct*, or *reconstructable*, if there exists a spherical polyhedron that projects onto it, with distinct planes for adjacent

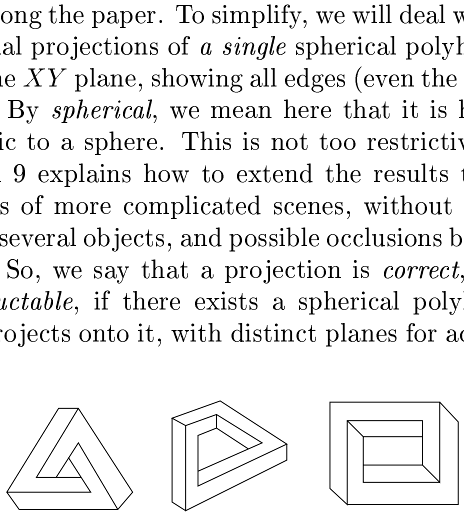
Figure 2: Some impossible projections.

faces. Such a polyhedron is called a *reconstruction* of the projection. The *edges* and *vertices* of a projection directly correspond to their spatial counterparts.

We will also assume that the projection is given along with its *incidence structure*. The incidence structure tells the combinatorial structure of the spatial reconstruction—basically, which points will be incident to which faces. More formally, it is a triple $I = (V, F, R)$, where V is the set of vertices and F is the set of its *faces*. We put a face in F for every subset of vertices that must be kept coplanar in the spatial reconstruction. $R \subseteq V \times F$ is the *incidence set*: there is an *incidence pair* (v, f) in R if vertex v must lie on face f in space. The incidence structure can be computed by applying the method in [13, page 45], after a labeling of its edges has been obtained using standard techniques like those in [2, 15, 7, 6].

In his book [13], Sugihara gives an algebraic test for correctness that, roughly speaking, consists of telling whether a system of linear equalities and inequalities has a solution, which is solvable via linear programming. This system contains an equation of the form

$$(v_x, v_y, v_z, 1) \cdot (A_f, B_f, 1, D_f)^T = 0 \qquad (1)$$

for every incidence pair $(v, f) \in R$, to express the constraint that, in any reconstruction, vertex v must lie on the plane of face f: $A_f x + B_f y + z + D_f = 0$. To have a set of necessary and sufficient conditons for correctness, Sugihara also adds other depth relations, but, for simplicity, these are omitted here.

One can easily see that after collecting all the equations (1) for the projection in fig. 3a this linear system has a solution space of dimension four, corresponding to the heights of four vertices that one must fix to get a spatial reconstruction. However, the reader can easily check the superstrictness of this test: after moving slightly v_6 the dimension of the solution space drops to three, meaning that the only reconstruction is a flat object, with *all* vertices coplanar, and the projection is judged as incorrect.

In Whiteley's cross-section test [17], a projection of a spherical polyhedron is correct if, and only if, it is possible to draw a *compatible cross-section* of it. The cross-section is a diagram showing the lines of intersection of all faces with one selected face of the polyhedron. Fig. 3b shows (in bold gray) a cross-section of a correct truncated tetrahedron: every line L_f is the intersection of a face plane f with the background face f_5. The cross-section is *compatible* if the line of any edge between a pair of faces contains the point of intersection of the cross-section lines of these faces.

This is also a superstrict test. For example, after moving v_6 slightly (fig. 3e), the cross-section is not

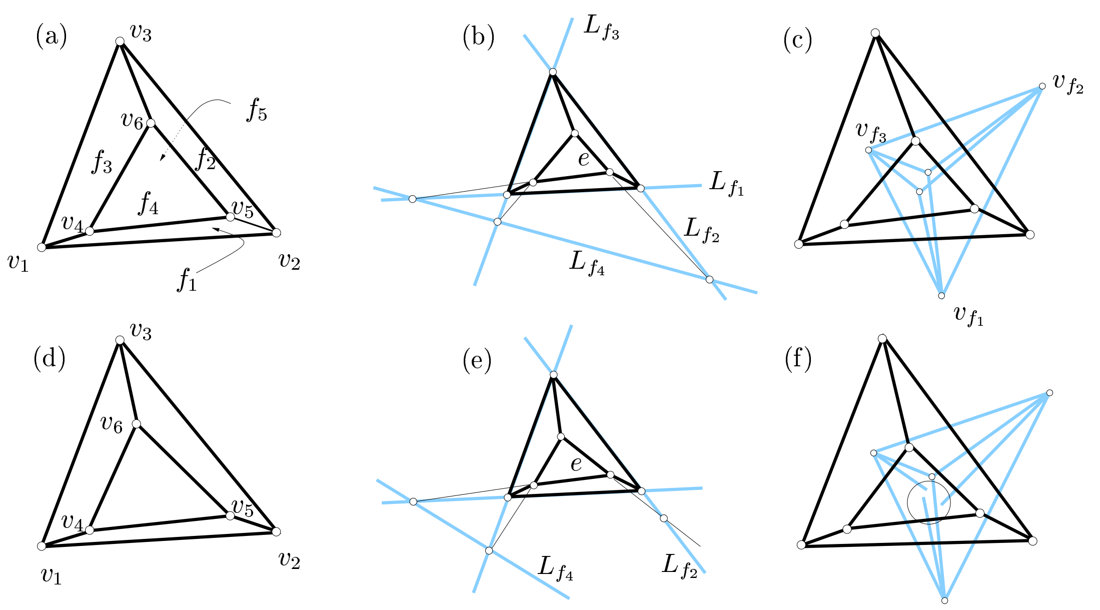

Figure 3: A correct truncated tetrahedron (a), with two compatible diagrams: the cross-section (b), and the gradient-space diagram (c). If v_6 is slightly moved (d), the diagrams are not compatible anymore (e and f).

compatible anymore: edge e does not meet the intersection of lines L_α and L_β.

Huffmann [3] and Mackworth [4] use the so-called *dual diagram* for the same purpose [1]. For a projection to be correct, it must have a *compatible* dual diagram. In this diagram, there is a dual vertex v_f for every face f in the projection, and there is a dual edge joining two dual vertices if their corresponding faces share an edge in the projection. The dual diagram is compatible if every edge in the projection is orthogonal to its dual edge in the diagram. Hence, it is possible to generate a dual diagram for a correct truncated tetrahedron, but not for an incorrect one (fig. 3, c and f), and "almost correct" projections are judged as incorrect.

3 Related Work

Up to the authors' knowledge, the Literature offers two approaches to overcome the superstrictness issue: a correction strategy due to Sugihara [13, chapter 7] and an explicit handling of uncertainty, due to Ponce and Shimshoni [8, 12].

Roughly speaking, Sugihara's method works as follows. Think about the truncated tetrahedron in fig. 3a. We see that fixing the heights of v_1, v_2, v_3 and v_4 is enough to determine the heights of the others, as we can use the coplanarity constraint of each face to derive them. However, the height of v_6 is overconstrained, as it can be deduced from *both* the coplanarity of f_3 and f_2. Only when the projection is correct, this height will be identical when computed from both faces. As fig. 3d is incorrect, a possibility is to drop out the constraint that v_6 must lie on f_3, f_2 and f_4, fix the heights of v_1, v_2, v_3, v_4, compute the resulting planes for the faces, and intersect f_2, f_3, f_4 to get a corrected position for v_6.

[1]Actually, this diagram was already discovered in the last century by Maxwell. See for example [5].

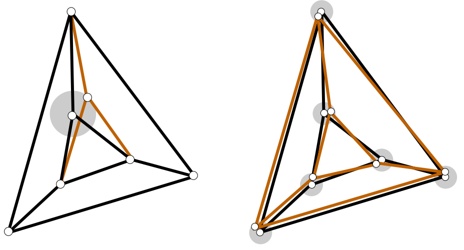

Figure 4: Corrected projections (in grey) of a truncated tetrahedron.

In sum, the strategy is to remove some vertices until a reconstructable projection appears, then reconstruct it, and derive the new positions for the removed vertices. Sugihara gives a characterization of when the incidence structure yields a reconstructable projection with vertices in generic position and an efficient graph flow algorithm to remove a minimal set of incidences until the reconstruction is possible.

However, as already noted by himself, this correction is not possible when the removed vertices are incident to more than three non-triangular faces. Unfortunately, one can find projections where this does happen (see the analyis by Whiteley in [16]). Also, another problem of this technique is that the corrected projection may deviate substantiously from some original vertices, when, moving just a bit all of them, one can find projections that fall in a smaller neighborhood (compare figs. 4-left and right).

On a different approach, Ponce and Shimshoni explicitly consider that a vertex true position (x_i, y_i) is unknown in the projection, but that must fall in a square of side 2ϵ around the measured position $(\tilde{x}_i, \tilde{y}_i)$. Then, they take Sugihara's system of linear equations (1) and do the change of variables $x_i = \tilde{x}_i + \mu_i$, $y_i = \tilde{y}_i + \nu_i$ for every vertex (x_i, y_i), and add the constraints $|\mu_i| \leq \epsilon$, $|\nu_i| \leq \epsilon$.

This leads to a system of *nonlinear* equalities and inequalities that, after a clever addition of gradient-space constraints, and some algebraic manipulation, they are able to linearize again. The linearization, however, is gained at the cost of the sufficiency of the test and, as they note, the resulting constraints are only necessary for a projection to be correct.

The approach we present overcomes the problems of both methods. On the one hand we allow the movement of *all* vertices, to get closer corrected projections. On the other hand, as we avoid adapting to a specific correctness test, any one offering neccessary and sufficent conditions can be applied to the corrected projection.

4 The Overall Algorithm

Given an incorrect projection $\mathcal{P}^{inc}$, our goal is to obtain a correct one $\mathcal{P}^{cor}$, with the same incidence structure as $\mathcal{P}^{inc}$, which is as close to $\mathcal{P}^{inc}$ as possible. As a function measuring the distance between the two projections, we have chosen the sum of the squared euclidean distances between pairs of corresponding vertices in $\mathcal{P}^{inc}$ and $\mathcal{P}^{cor}$.

The problem can be stated as follows. If v_i^{inc} and v_i^{cor} denote, respectively, the 2D coordinates of the ith vertex of $\mathcal{P}^{inc}$ and $\mathcal{P}^{cor}$, we want to minimize

$$\sum_{i=1}^{n} \|v_i^{inc} - v_i^{cor}\|^2$$

subject to the constraint that the vertices v_i^{cor} define a correct projection $\mathcal{P}^{cor}$ with the same incidence structure as $\mathcal{P}^{inc}$.

However, we will show that it is possible to parameterize the 2D coordinates of the vertices of all correct projections with a given incidence structure. More precisely, given an incidence structure $I^{cor} = (V, F, R)$, it is possible to write the coordinates

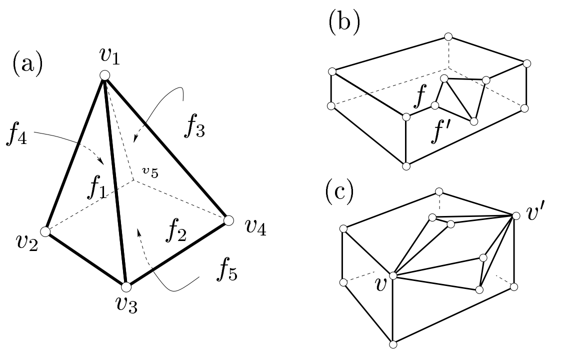

Figure 5: (a) A pyramid with a quadrangular base. (b) Two faces sharing more than two vertices. (c) Two vertices sharing more than two faces.

(x_i^{cor}, y_i^{cor}) of every vertex $v_i^{cor} \in V$ as functions

$$x_i^{cor} = \chi_i(p_1, p_2, ..., p_n)$$
$$y_i^{cor} = \psi_i(p_1, p_2, ..., p_n),$$

in such a way that any tuple of parameters $(p_1, p_2, ..., p_n)$, $p_i \in \Re^n - \mathcal{F}\ \forall i$, where $\mathcal{F}$ is a zero-measure subset of $\Re^n$, fixes a correct projection. χ_i and ψ_i are rational functions and, thus, everything reduces to the unconstrained minimization of the rational function

$$\sum_{i=1}^{n} \|v_i^{inc} - (\chi_i(p_1, p_2, ..., p_n), \psi_i(p_1, p_2, ..., p_n))\|^2,$$

which can be solved computationally by starting a gradient search at an initial correct projection that estimates the final solution.

The next section presents the *resolvable sequence*, the key concept that leads to this parameterization.

5 The Resolvable Sequence

Is there a set of independent choices that can be made to construct a polyhedron in a consistent manner? To illustrate this question, let us focus on the simple example of fig. 5a, a pyramid with a quadrilateral base. The shape of this polyhedron can be fixed by, for example, giving the coordinates of all its vertices or the face plane coefficients of all its faces. But care must be taken in any of the two ways. If we arbitrarily fix all planes, then f_1, f_2, f_3 and f_4 will not probably have a common point of intersection and vertex v_4 will be inconsistently defined. On the contrary, if we arbitrarily fix all vertices, v_2, v_3, v_4 and v_5 need not be coplanar and face f_5 might be inconsistently defined.

In general, we say that a polyhedron is *resolvable* if it is possible to list its vertices and faces in a sequence $\mathcal{S} = (\ldots, v_i, \ldots, f_j, \ldots)$ in such a way that:

(C1) when a vertex occurs in $\mathcal{S}$, it is incident to at most three previous faces;

(C2) when a face occurs in $\mathcal{S}$, it is incident to at most three previous vertices;

(C3) when two faces f and f' share three or more vertices (fig. 5b), f and f' appear earlier in $\mathcal{S}$ than the third of the common vertices;

(C4) when two vertices v and v' are incident to three or more common faces (fig. 5c), both v and v' appear earlier than the third of the common faces.

$\mathcal{S}$ is called a *resolvable sequence* for the polyhedron. Note that if such a sequence exists, then we can construct the polyhedron in a consistent way. We just need to fix its vertices and faces, one by one, following the order in $\mathcal{S}$. Along the way, when an element is underconstrained by previous choices, additional choices can be taken arbitrarily.

In 1934, Steinitz proved that all polyhedra whose graph of vertices and edges is planar and 3-connected are resolvable [18]. Actually, for these polyhedra it suffices to find a sequence satisfying conditions (C1) and (C2) above as their 3-connectedness ensures they have no face sharing more than two vertices, nor any pair of vertices sharing more than two faces.

Recently, in [14] Sugihara has extended Steinitz's results, finding that actually all spherical polyhedra are resolvable. He also gives an algorithm to compute a resolvable sequence in $O(v+f)$ time, where v and f are the number of vertices and faces of the polyhedron, respectively.

6 Parameterizing Correct Projections

The resolvable sequence induces a parameterization of all polyhedra with a given incidence structure. For example, a trivial resolvable sequence for the truncated tetrahedron in fig. 3a is to first list all faces, and then all vertices: $\mathcal{S} = (f_1, \dots, f_5, v_1, \dots, v_6)$. (This is clearly valid for any *simple polyhedron*, one where every vertex has exactly three incident faces.) Thus, here, the coordinates (x_i, y_i, z_i) of every vertex v_i can be written as functions of the coefficients its three incident planes by, e.g., solving for x_i, y_i and z_i using Cramer's rule, and varying these coefficients we get different reconstructions of the truncated tetrahedron.

Note that the resolvable sequence also induces a parameterization of all correct projections with the given incidence structure, as we only need to project the (parameterized) spatial polyhedron onto the XY plane, keeping the parameterization for the X and Y coordinates of every vertex.

In the general case, we can write a parameterization of all polyhedra with a given incidence structure I as follows. First, we compute a resolvable sequence $\mathcal{S}$ for I. Then, we visit every element of $\mathcal{S}$, following the order of the sequence. If the element is a face, then:

- If it is not incident to any previous vertex, there is total freedom in choosing its position and the four coefficients of its plane are free parameters.
- If it is incident to three previous vertices, the parameters of the plane are totally fixed and no new parameter is introduced.
- If it is incident to two previous vertices, say p and q, we must select one of all the planes meeting the segment pq. Such a plane can be written as:

$$\begin{vmatrix} p_x & q_x & r_x & x \\ p_y & q_y & r_y & y \\ p_z & q_z & r_z & z \\ 1 & 1 & 1 & 1 \end{vmatrix} = 0$$

  and the three coordinates of a third point $r = (r_x, r_y, r_z)$ are introduced as new parameters.
- If a face is incident to one previous vertex p, its plane can be expressed as

$$(n_x, n_y, n_z) \cdot ((x, y, z) - (p_x, p_y, p_z)) = 0$$

  and the three coordinates of the normal vector (n_x, n_y, n_z) are chosen as parameters.

On the other hand, if the element is a vertex v, then:

- If v is incident to no previous face, there is total freedom in choosing its position and its three coordinates are taken as free parameters.
- If v is incident to three previous faces, the vertex is totally fixed and can be found computing the intersection of the three planes.
- If v is incident to two previous faces, say f_i and f_j, we can write two equations:

$$\begin{aligned} A_i v_x + B_i v_y + C_i v_z + D_i = 0 \\ A_j v_x + B_j v_y + C_j v_z + D_j = 0 \end{aligned}$$

  and solve for v_x and v_y in terms of v_z, which is introduced as a new parameter.
- Finally, if v is incident to one previous face, say f, we can freely choose v_x and v_y and get v_z from the equation of f's plane.

Note that this parameterization is rational, as at each step of its construction we can write a vertex or face coordinate as a quotient of polynomials in the parameters. Although for certain choices of the parameters it may fail to provide a polyhedron (e. g., there is an indetermination when a vertex is incident to three previous faces, and the chosen planes are not all distinct), this only happens for a zero-measure subset of the parameter space, posing no problem to the minimization, as section 8 explains.

7 A Good Starting Point

The resolvable sequence also gives an efficient way to derive a "reasonably close" correct projection to $\mathcal{P}^{inc}$. The idea is to properly place every vertex and face of the sequence, so that the 2D projection is locally close enough to the projection. Let us see this in detail.

We distinguish several situations, depending on whether we are fixing a vertex or a face.

Assume first that we are fixing a vertex v, whose 2D position in the incorrect projection is v^{inc}. v can be incident to zero, one, two or three previously-fixed faces. In the first case, there is total freedom in choosing v's spatial position but, to be compliant with the projection, we choose it to lie in the vertical line over v^{inc}, at any height. If v is incident to just one previous face, then we choose it over this face's plane, in the vertical line at v^{inc}. If v is incident to two faces with planes α and β (respectively), we fix v on the line of intersection of α and β at the place where its 2D projection is at a minimum distance from v^{inc}. Finally, if v is incident with three previous faces, we fix it in the intersection of their respective planes.

On the other hand, if we are fixing a face f, it can be incident to three, two, one or zero previously-fixed vertices. In the first case, there is no choice for the plane of f as it is fully determined by the three vertices. If the face is incident to two fixed vertices, say p and q, we can choose among all the planes meeting the line pq. But, which one? If some of the neighboring faces of f have already been fixed, say faces $f_{i_1}, f_{i_2}, \ldots$, then we would like that the lines of intersection of these faces with f lie reasonably close to the corresponding edges in the incorrect projection. If we label the common vertices between f and $f_{i_1}, f_{i_2}, \ldots$ as $w_1, \ldots, w_m$, and $z(w_i)$ denotes the height of vertex w_i as computed on the plane of its already-fixed face, over its position in the projection, then a reasonable way to achieve this is to fix f to the plane α that that meets the line pq and minimizes the sum of squared residuals:

$$\sum_{i=1}^{m}(z(w_i) - z_\alpha(w_i))^2$$

where $z_\alpha(w_i)$ denotes the z-coordinate of vertex w_i as given by the plane α. Obviously, if no adjacent face of f was previously fixed, we simply fix f at *any* plane meeting pq.

The remaining cases, when f is incident to *one* or to *no* previous vertex are analogous, the only difference being that we choose among all the planes meeting a fixed point in the first case, and among all possible planes of 3-space in the second.

Of course, following this strategy, the resulting correct projection may deviate substantially from $\mathcal{P}^{inc}$ at some vertices, specially if the projection is large enough. However, the experiments show that, even with a very far initial estimation, the minimization converges rapidly to a very close projection (see below).

8 Implementation and Results

The correction algorithm has been implemented in C for projections of simple polyhedra, as these have the advantage that every vertex is easily parameterized by the twelve coefficients of its three incident planes.

For the minimization, we use TNPACK, a freely available package specially suited for large-scale problems with possibly thousands of variables [11]. To minimize a function $F(X), X \in \Re^n$ TNPACK implements the iterative truncated Newton method, based on minimizing a local quadratic approximation to F at every step. For efficiency, an approximated (truncated) solution of this local minimization is allowed, which is computed through a preconditioned conjugate gradient algorithm.

The user must essentially supply three routines, returning F, its gradient, and the Hessian matrix, evaluated at a given point $X \in \Re^n$. For the gradient we directly provide the symbolic expressions, as they are easy to derive. For the Hessian matrix, we rely on an (optional) internal TNPACK routine that uses finite differences of the gradient to compute it. To prevent the minimization from falling in a point X of parameter space yielding indetermination (see section 6) the routine computing $F(X)$ is implemented to return a very high value for these configurations.

We have tested the correction process on several projections of simple polyhedra: a truncated tetrahedron, a dodecahedron, a truncated icosahedron, a rhombitruncated-cubeoctahedron, and a rhombitruncated-icosidodecahedron (fig. 6). The number of optimization variables involved in these examples is 20, 48, 104, 128, and 248, respectively—four times the number of faces. A corrected projection for them is obtained in less than 5 seconds of CPU time on a SUN Ultra-2. We note that, although simple, these projections are far more complex than those one can find in the literature [13, 12].

After the tests, the heuristic technique of section 7 reveals as a good way for computing a starting point from which local minima are avoided. But, even when the initial approximation is somewhat distant to the input projection, the minimization converges rapidly to a neighborhood of the incorrect projection. Figure 6 illustrates this, showing the sequence of intermediate projections of a dodecahedron as the minimization progresses.

9 Conclusions and Future Work

Real scenes of polyhedra differ substantiously from the model assumed: hidden edges are not visible, several objects coexist, and they possibly occlude one another (fig. 7 left). However, this paper's results can be extended to deal with them. We shortly comment how and point the reader to [9] for further details.

First, instead of a polyhedron, we can assume the projection depicts a projected *polysurface*: a surface made of planar polygonal panels glued along some edges, and possibly containing holes. This provides an accurate model for opaque objects, as invisible parts underneath the "topmost" visible panels are irrelevant. For such projections, we need to have every hole identified, which can be done by collecting all contours of occluding edges (as defined in [13]) after an edge-labeling algorithm has been applied. Then, we can fill every hole with triangles connecting its vertices to obtain a new polysurface with spherical incidence structure. The addition of triangles is inoquous: it can be seen that the original polysurface is correct if and only if the new one is. But, as the latter is spherical, we can derive a resolvable sequence for it, which can be used to parameterize the realizations of the original. Finally, when several polysurfaces coexist in a projection, we can treat each one separately in the same way. This strategy has been implemented and fig. 7 shows the results on a synthetic polyhedral scene.

Finally, it is worth to mention two points deserving further attention. Although the initial estimated projection is fairly good, the minimization is not guaranteed to converge to the global minimum. To mend this up, one can always start the search at several different initial estimations, derived from different resolvable sequences of the same projection, and select the best corrected projection.

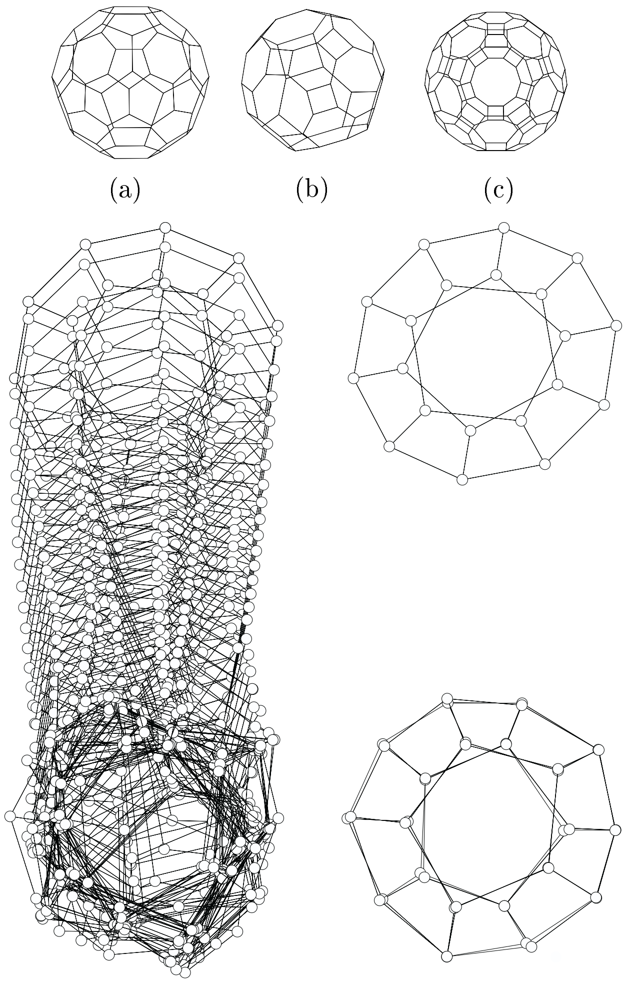

Figure 6: Top: correct projections of a truncated icosahedron (a), a rhombitruncated cubeoctahedron (b) and a rhombitruncated icosidodecahedron (c). Bottom: A correction sequence of a dodecahedron (left) from which the initial estimation and the final correction (overlapping the input incorrect projection) have been singled out (right).

Another possibility could be to derive a polynomial (rather than rational) parameterization, by working in projective instead of affine coordinates, and attempt to find the global minimum through interval arithmetic techniques [1].

Acknowledgements

Many of the ideas here have developed thanks to numerous conversations with Walter Whiteley and Kokichi Sugihara. The authors are very grateful to them.

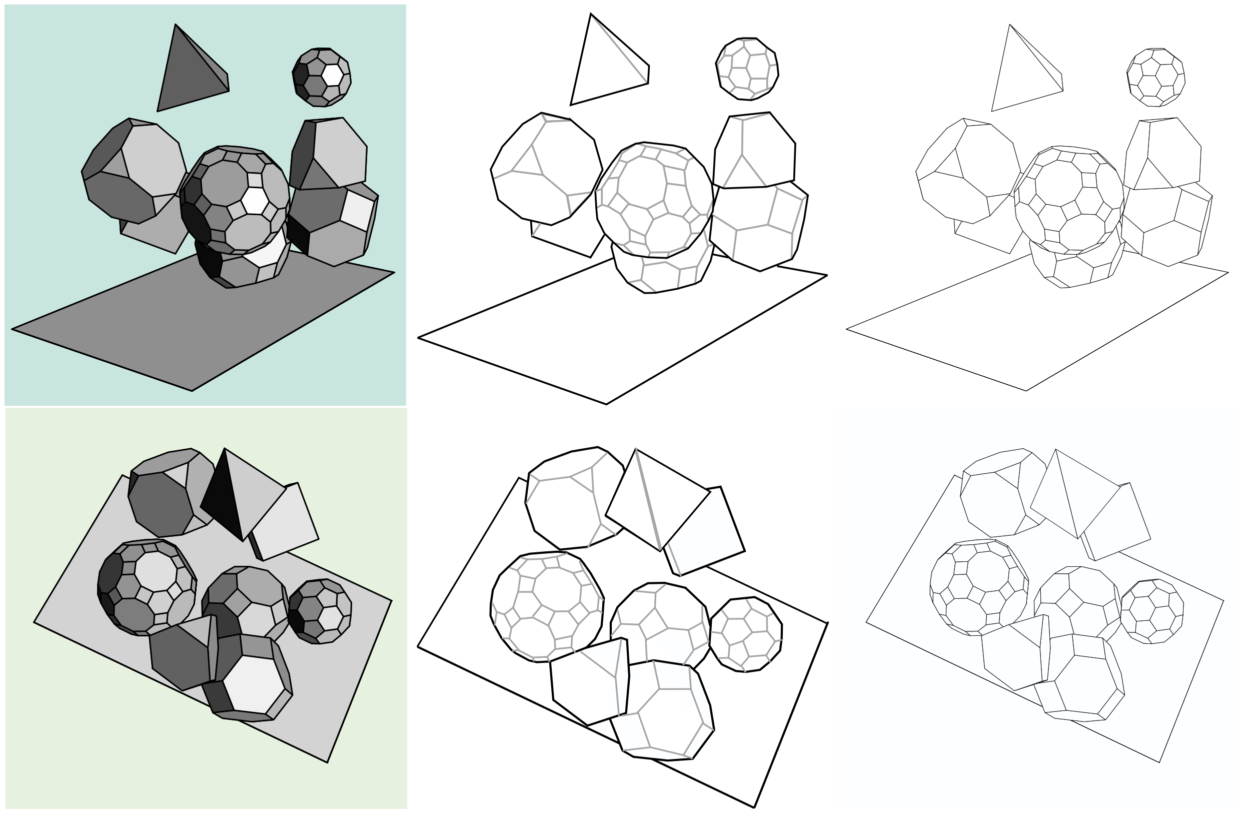

Figure 7: Two views of a same polyhedral scene (left) together with two incorrect projections of them, with the contours labelled in black (central), and the final corrections (right).

References

[1] E. Hansen. *Global Optimization Using Interval Analysis*. Pure and Applied Mathematics. Marcel Decker, Inc., New York, 1992.

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

[3] D. A. Huffman. Realizable configurations of lines in pictures of polyhedra. *Machine Intelligence*, 8, 1977.

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

[5] J. C. Maxwell. On reciprocal figures and diagrams of forces. *Philosophical Magazine*, Ser. 4, Vol. 27, pp. 250-261, 1864.

[6] P. Parodi, R. Lancewicki, A. Vijh, and J. Tsotsos. Empirically-derived estimates of the complexity of labelling line drawings of polyhedral scenes. *Artificial Intelligence*, 105(1-2):47–75, 1998.

[7] P. Parodi and V. Torre. On the complexity of labelling perspective projections of polyhedral scenes. *Artificial Intelligence*, 70(1-2):239–276, October 1994.

[8] 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, May 1992.

[9] L. Ros. *A Kinematic-Geometric Approach to Spatial Interpretation of Line Drawings.* PhD thesis, Polytechnic University of Catalonia, May 2000. Available at `http://www-iri.upc.es/people/ros`.

[10] L. Ros and F. Thomas. Analysing spatial realizability of line drawings through edge-concurrence tests. In *IEEE Int. Conf. on Robotics and Automation*, volume IV, pages 3559–3566, Leuven, Belgium, May 1998. IEEE Computer Society Press.

[11] T. Schlik and A. Fogelson. TNPACK-A truncated Newton minimization package for large-scale problems: I. Algorithm and usage. *ACM Transactions on Mathematical Software*, 18(1):46–70, March 1992.

[12] I. Shimshoni. *Interpreting Images of Polyhedral Objects in the Presence of Uncertainty.* PhD thesis, University of Illinois at Urbana-Champaign, 1995.

[13] K. Sugihara. *Machine Interpretation of Line Drawings.* The MIT Press, 1986.

[14] K. Sugihara. Resolvable representation of polyhedra. *Discrete and Computational Geometry*, 21(2):243–255, 1999.

[15] D. Waltz. Understanding line drawings of scenes with shadows. In P. H. Winston, editor, *The Psychology of Computer Vision*, pages 19–91. Mc. Graw Hill, 1975.

[16] W. Whiteley. From a line drawing to a polyhedron. *Journal of Mathematical Psychology*, 31:441–448, 1987.

[17] W. Whiteley. Weavings, sections and projections of spherical polyhedra. *Discrete Applied Mathematics, 32*, pages 275–294, 1991.

[18] W. Whiteley. How to design or describe a polyhedron. *Journal of Intelligent and Robotic Systems*, 11:135–160, 1994.