# Parametric Models are Versatile: The Case of Model Based Optimization

P. Fua
Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

**Abstract**

Model-Based Optimization (MBO) is a paradigm in which an objective function is used to express both geometric and photometric constraints on features of interest. A parametric model of a feature (such as a road, a building, or coastline) is extracted from one or more images by adjusting the model's state variables until a minimum value of the objective function is obtained. The optimization procedure yields a description that simultaneously satisfies (or nearly satisfies) all constraints, and, as a result, is likely to be a good model of the feature.

## 1 Introduction

Model-Based Optimization (MBO) is a paradigm in which an objective function is used to express both geometric and photometric constraints on features of interest. A parametric model of a feature (such as a road, a building, or coastline) is extracted from one or more images by adjusting the model's state variables until a minimum value of the objective function is obtained. The optimization procedure yields a description that simultaneously satisfies (or nearly satisfies) all constraints, and, as a result, is likely to be a good model of the feature.

The deformable models we use here are extensions of traditional snakes [Terzopoulos *et al.*, 1987, Kass *et al.*, 1988, Fua and Leclerc, 1990]. They are polygonal curves or facetized surfaces to which is associated an objective function that combines an "image term" that measures the fit to the image data and a regularization term that enforces geometric constraints.

In this paper we demonstrate cartographic applications of this paradigm and show that a large variety of objects can be thus modeled. More specifically we use MBO to effectively delineate 2D and 3D features—such as roads, rivers and buildings—and to recover the shape of the surrounding terrain. The algorithms described below are implemented within the Radius Common Development Environment (RCDE) [Mundy *et al.*, 1992].

## 2 2–D and 3–D Delineation

We model linear features as polygonal curves that may either be described as sequential list of vertices or, for more complex objects such as a road network or a 3–D extruded object, exhibit the topology of a network. In the latter case, to describe them completely, one must supply not only the list of their vertices but also a list of "edges" that defines the connectivity of those vertices. In addition, with some of these complex objects, one can also define "faces," that is circular lists of vertices that must be constrained to remain planar.

Our ultimate goal is to accommodate the full taxonomy of snakes described by table 1. The colums represent different type of snakes and the rows different kinds of constraints that can be brought to bear. The table entries are examples of objects that can be modeled using these combinations.

| Constraints/Type | Simple curve | Ribbon curve | Network |
|---|---|---|---|
| Smooth | Low res. roads, rivers. | High res. roads. | Road network. |
| Polygonal | Man-made structures. | City streets | Street Networks. |
| Planar | Planar structures. | City streets | Street Networks. |
| Rectilinear | Roof tops, parking lots. | City streets | Buildings. |

Table 1: Snake taxonomy. The columns represent different types of snakes and the rows different kinds of constraints that can be brought to bear. The table entries are examples of objects that can be modeled using these combinations.

## 2.1 Polygonal Snakes

A simple polygonal snake, $\mathcal{C}$, can be modeled as a sequential list of vertices, that is, in two dimensions, a list of 2–D vertices $\mathcal{S}_2$ of the form

$$\mathcal{S}_2 = \{(x_i\ y_i),\ i = 1, \ldots, n\} \quad , \tag{1}$$

and, in three dimensions, a list of 3–D vertices $\mathcal{S}_3$ of the form

$$\mathcal{S}_3 = \{(x_i\ y_i\ z_i),\ i = 1, \ldots, n\} \quad . \tag{2}$$

In the two dimensional case, the "image energy" of these curves—the term we try to minimize when we perform the optimization, is taken to be

$$\mathcal{E}_I(\mathcal{C}) = -\frac{1}{|\mathcal{C}|} \int_0^{|\mathcal{C}|} |\nabla \mathcal{I}(\mathbf{f}(s))|\ ds, \tag{3}$$

where $I$ represents the image gray levels, $s$ is the arc length of $\mathcal{C}$, $\mathbf{f}(s)$ is a vector function mapping the arc length $s$ to points $(x, y)$ in the image, and $|\mathcal{C}|$ is the length of $\mathcal{C}$. In practice, $\mathcal{E}_I(\mathcal{C})$ is computed by integrating the gradient values $|\nabla \mathcal{I}(\mathbf{f}(s))|$ in precomputed gradient images along the line segments that connect the polygonal vertices.[1] We therefore rewrite $\mathcal{E}_I$ as

$$\begin{aligned} \mathcal{E}_I &= \sum_{1 \le i < n} S((x_i, y_i), (x_{i+1}, y_{i+1})) / \sum_{1 \le i < n} L_{i,i+1} \quad , \\ S((x_i, y_i), (x_j, y_j)) &= -\int_0^1 |\nabla \mathcal{I}(x_i + \lambda(x_j - x_i), y_i + \lambda(y_j - y_i))|\, d\lambda \quad , \\ L_{i,j} &= \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad . \end{aligned} \tag{4}$$

$L_{ij}$ is the length of the individual line segments and $S((x_i, y_i), (x_j, y_j))$, the sum of the gradient values along one segment, is computed by sampling the segment at regular intervals.

In the three dimensional case, $\mathcal{E}_I(\mathcal{C})$ is computed by projecting the curve into a number of images, computing the image energy of each projetion and summing these energies. Formally, given a set of $N$

[1] The gradient images are computed by gaussian smoothing the original image and taking the $x$ and $y$ derivatives to be finite differences of neighboring pixels.

images and corresponding camera models, we write

$$\mathcal{E}_I = \sum_{1 \leq k \leq N} \mathcal{E}_I^k \quad , \tag{5}$$

$$\mathcal{E}_I^k = \sum_{1 \leq i < n} S(Pr^k(x_i, y_i, z_i), (Pr^k(x_{i+1}, y_{i+1}, z_{i+1}))) / \sum_{1 \leq i < n} L_{i,i+1}^k \quad , \tag{6}$$

where $k$ denotes the image number, $Pr^k(x, y, z)$ the pair of coordinates of the projection of point $(x, y, z)$ into image $k$ and $L_{i,j}^k$ the length of the projection into image $k$ of the segment $i, j$.

## 2.2 Smooth Snakes and Ribbons

These snakes are used to model smoothly curving features such as roads or ridge-lines.

**2–D curves.** Following Kass *el al.* [1988], we choose the vertices of such curves to be roughly equidistant and add to the image energy $\mathcal{E}_I$ a regularization term $\mathcal{E}_D$ of the form

$$\mathcal{E}_D(\mathcal{C}) = \mu_1 \sum_i (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + \mu_2 \sum_i (2x_i - x_{i-1} - x_{i+1})^2 + (2y_i - y_{i-1} - y_{i+1})^2 \tag{7}$$

and define the "total energy" $\mathcal{E}_T$ as

$$\mathcal{E}_T(\mathcal{C}) = \mathcal{E}_D(\mathcal{C}) + \mathcal{E}_I(\mathcal{C}) \tag{8}$$

The first term of $\mathcal{E}_D$ approximates the curve's tension and the second term approximates the sum of the square of the curvatures, assuming that the vertices are roughly equidistant. In addition, when starting, as we do, with regularly spaced vertices, this second term tends to maintain that regularity. To perform the optimization we could use the steepest or conjugate gradient, but it would be slow for curves with large numbers of vertices. Instead, it has proven much more effective to embed the curve in a viscous medium and solve the equation of the dynamics

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial S} + \alpha \frac{dS}{dt} &= 0, \\ \text{with } \frac{\partial \mathcal{E}}{\partial S} &= \frac{\partial \mathcal{E}_D}{\partial S} + \frac{\partial \mathcal{E}_I}{\partial S}, \end{aligned} \tag{9}$$

where $\mathcal{E}$ is the energy of Equation 8, $\alpha$ the viscosity of the medium, and $S$ the state vector that defines the current position of the curve. Since the deformation energy $\mathcal{E}_D$ in Equation 7 is quadratic, its derivative with respect to S is linear and therefore Equation 9 can be rewritten as

$$\begin{aligned} K_S S_t + \alpha(S_t - S_{t-1}) &= -\left.\frac{\partial \mathcal{E}}{\partial S}\right|_{S_{t-1}} \\ \Rightarrow (K_S + \alpha I) S_t &= \alpha S_{t-1} - \left.\frac{\partial \mathcal{E}}{\partial S}\right|_{S_{t-1}} \end{aligned} \tag{10}$$

where

$$\frac{\partial \mathcal{E}_D}{\partial S} = K_S S,$$

and $K_S$ is a sparse matrix. Note that the derivatives of $\mathcal{E}_D$ with respect to $x$ and $y$ are decoupled so that we can rewrite Equation 10 as a set of two differential equations in the two spatial coordinates

$$\begin{aligned} (K + \alpha I) X_t &= \alpha X_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial X}\right|_{X_{t-1}} \\ (K + \alpha I) Y_t &= \alpha Y_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial Y}\right|_{Y_{t-1}} \end{aligned}$$

where $K$ is a pentadiagonal matrix, and $X$ and $Y$ are the vectors of the $x$ and $y$ vertex coordinates. Because $K$ is pentadiagonal, the solution to this set of equations can be computed efficiently in $O(n)$ time using LU decomposition and backsubstitution. Note that the LU decomposition need be recomputed only when $\alpha$ changes.

In practice $\alpha$ is computed in the following manner. We start with an initial step size $\Delta_p$, expressed in pixels, and use the following formula to compute the viscosity:

$$\alpha = \frac{\sqrt{2n}}{\Delta_p} \left| \frac{\partial \mathcal{E}}{\partial S} \right|, \tag{11}$$

where n is the number of vertices. This ensures that the initial displacement of each vertex is on the average of magnitude $\Delta_p$. Because of the non linear term, we must verify that the energy has decreased from one iteration to the next. If, instead, the energy has increased, the curve is reset to its previous position, the step size is decreased, and the viscosity recomputed accordingly. This is repeated until the step size becomes less than some threshold value. In most cases, because of the presence of the linear term that propagates constraints along the whole curve in one iteration, it takes only a small number of iterations to optimize the initial curve.

**3–D Curves.** To extend the smooth snakes to three dimensions, we add one term in $z$ to the deformation energy of Equation 7 and $\mathcal{E}_D$ becomes

$$\begin{aligned} \mathcal{E}_D(\mathcal{C}) &= \mu_1 \sum_i (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2 \\ &+ \mu_2 \sum_i (2x_i - x_{i-1} - x_{i+1})^2 + (2y_i - y_{i-1} - y_{i+1})^2 + (2z_i - z_{i-1} - z_{i+1})^2 \end{aligned} \tag{12}$$

Since the derivatives of $\mathcal{E}_D$ with respect to $x$, $y$, and $z$ are still decoupled, we can rewrite Equation 10 as a set of three differential equations in the three spatial coordinates:

$$\begin{aligned} (K + \alpha I) X_t &= \alpha X_{t-1} - \left. \frac{\partial \mathcal{E}_I}{\partial X} \right|_{X_{t-1}} \\ (K + \alpha I) Y_t &= \alpha Y_{t-1} - \left. \frac{\partial \mathcal{E}_I}{\partial Y} \right|_{Y_{t-1}} \\ (K + \alpha I) Z_t &= \alpha Z_{t-1} - \left. \frac{\partial \mathcal{E}_I}{\partial Z} \right|_{Z_{t-1}} \end{aligned}$$

where $X$,$Y$, and $Z$ are the vectors of the $x$,$y$, and $z$ vertex coordinates.

The only major difference with the 2–D case is the use of the images' camera models. In practice, $\mathcal{E}_I(\mathcal{C})$ is computed by summing gradient values along the line segments linking the vertices' projections. These projections, and their derivatives, are computed from the state vector $S$ using the camera models. Similarly, to compute the viscosity, we use the camera models to translate the average initial step $\Delta_p$, a number of pixels, into a step $\Delta_w$ expressed in world units and use the latter in Equation 11.

**Ribbons** 2–D snakes can also be extended to describe ribbon-like objects such as roads in aerial images. A ribbon snake is implemented as a polygonal curve forming the center of the road. Associated with each vertex $i$ of this curve is a width $w_i$ that defines the two curves that are the candidate road boundaries. The state vector $S$ becomes the vector $S = \{(x_i\ y_i\ w_i)\},\ i = 1, \ldots, n\}$ and the average edge strength the sum of the edge strengths along the two boundary curves. Since the width of roads tends to vary gradually, we add an additional energy term of the form

$$\mathcal{E}_W(\mathcal{C}) = \sum_i (w_i - w_{i-1})^2 \tag{13}$$

$$\Rightarrow \frac{\partial \mathcal{E}_W}{\partial W} = LW,$$

where $W$ is the vector of the vertices' widths and $L$ a tridiagonal matrix. The total energy can then be written as

$$\mathcal{E}_(\mathcal{C}) = \lambda_D \mathcal{E}_D(\mathcal{C}) + \lambda_W \mathcal{E}_W(\mathcal{C}) + \lambda_G \mathcal{E}_I(\mathcal{C})$$

where $\lambda_D$ and $\lambda_W$ wheigh the contributions of the two geometric terms. At each iteration the system must solve the three differential equations:

$$\begin{aligned}
(K+\alpha I)X_t &= \alpha X_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial X}\right|_{X_{t-1}} \\
(K+\alpha I)Y_t &= \alpha Y_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial Y}\right|_{Y_{t-1}} \\
(K+\alpha I)W_t &= \alpha W_{t-1} - \left.\frac{\partial \mathcal{E}_I}{\partial W}\right|_{W_{t-1}}
\end{aligned}$$

2–D ribbons can be turned into 3–D ones in exactly the same way 2–D snakes are turned into 3–D ones. The state vector $S$ becomes the vector $S = \{(x_i\ y_i\ z_i\ w_i)\},\ i = 1, \ldots, n\}$ and at each iteration the system must solve four differential equations, one for each coordinate.

## 2.3 Network Snakes

The 2–D and 3–D "network snakes" are a direct extension of the polygonal snakes of Section 2.1.

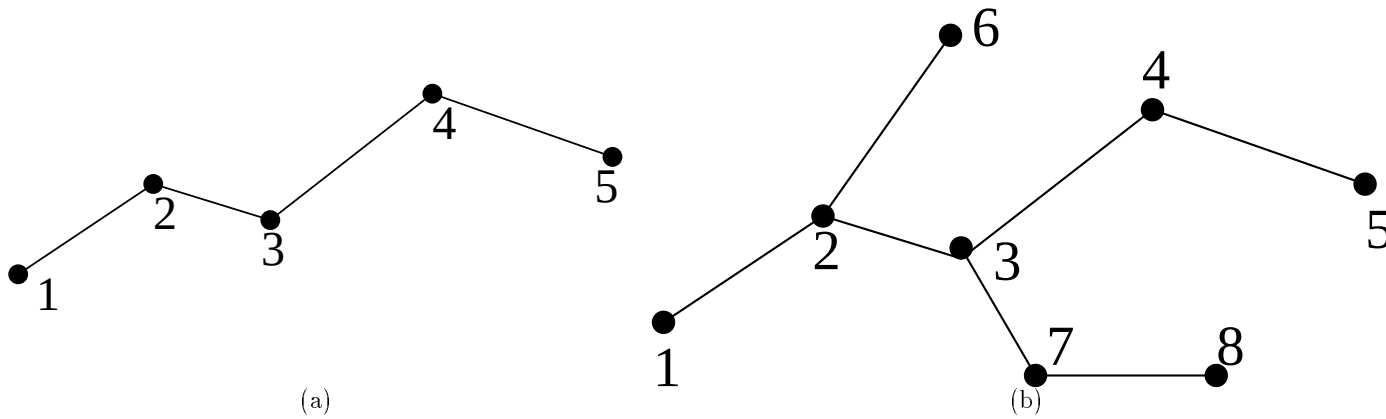
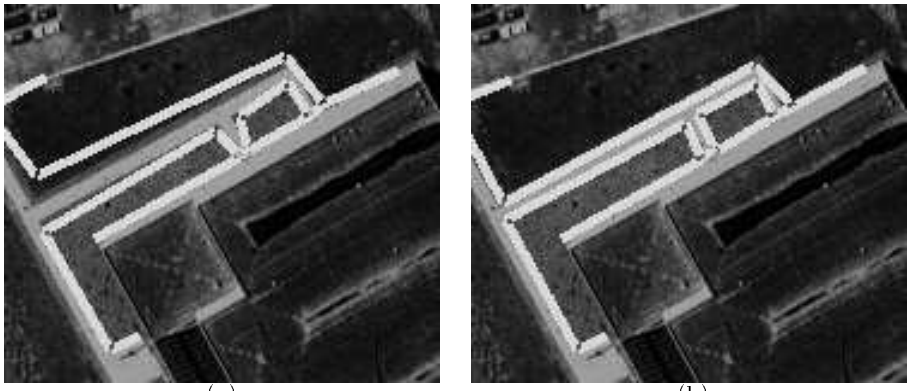



Figure 1: Snake Topology. (a) A simple polygonal curve is described by a sequential list of vertices $v_i$— $v_{1\leq 5}$ here. (b) A network is decribed by a list of vertices $v_i$— $v_{1\leq 8}$ here , and a list of edges—((1 2) (2 3) (3 4) (4 5) (2 6) (3 7) (7 8)) here.

In the two-dimensional case, the extension is straightforward. A network snake is now defined by a list of $n$ vertices $\mathcal{S}_\epsilon$ as before and list of edges $\mathcal{A} = \{(i,j)$ where $1 \leq i \leq n$ and $1 \leq j \leq n\}$. Figure 1 depicts such a network snake. $\mathcal{E}_I(\mathcal{C})$ is computed as

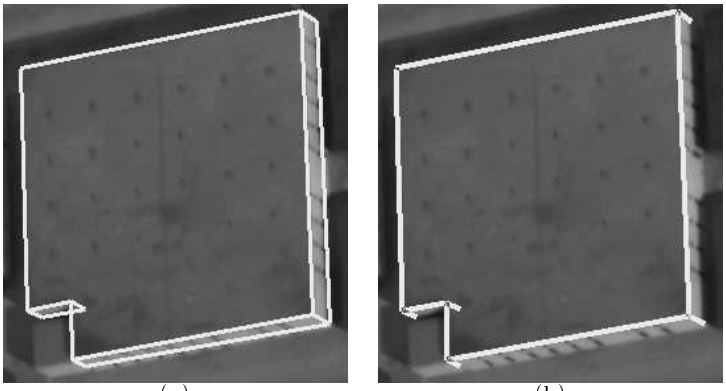
$$\mathcal{E}_I(\mathcal{C}) = \sum_{(i,j)\in\mathcal{A}} S((x_i, y_i)(x_j, y_j)) / \sum_{(i,j)\in\mathcal{A}} L^k_{i,j} \quad , \tag{14}$$

where $S$ and $L$ are the functions defined in Equation 5. It is optimized using either steepest gradient descent or conjugate gradient. In Figure 2, we show an example of such a network. When constraints, such as planarity or rectilinearity, are imposed on the network, constrained optimization can also be used [Gill *et al.*, 1981, Brechbühler, 1995, Brechbühler *et al.*, 1995].



(a) (b)

Figure 2: Optimizing a two-dimensional polygonal network. (a) Initialization (b) Network after optimization.



(a) (b)

Figure 3: Edge Visibility. (a) An RCDE "extruded-object." Only the visible faces, that is those whose normal is oriented towards the viewer are drawn. Note that this heuristic does not account for non-convexity, as a result the faces in the lower left corner of the image are improperly drawn. (b) The network snake generated to optimize the extruded-object. It includes roof-edges and vertical wall-edges. The edges at the back of the building are not drawn—and not used during the computations involving these views—because they belong to hidden faces. The edges at the base of the building are treated as invisible because their appearance is unreliable in typical imagery.

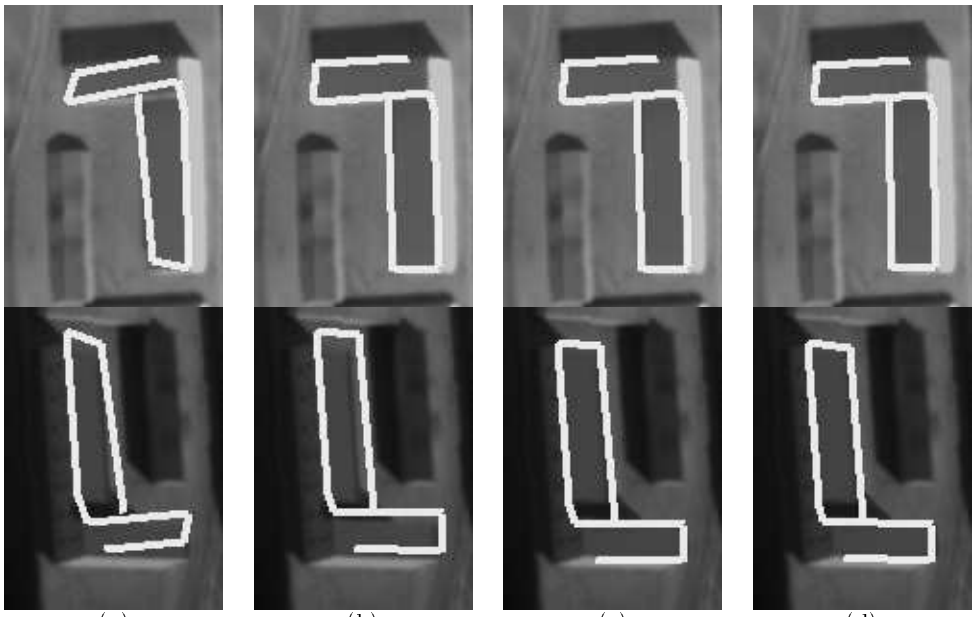
In the three-dimensional case, one must take into account the fact that not all the network's edges are visible in all views. As a result one must also provide, for each projection of the snake into all the images, a list of visible edges. We compute this list by using the face-visibility methods embedded in RCDE: we assume

that only edges that belong to visible faces are visible. This is effective for convex objects but my fail for concave ones. Figure 3 illustrates the strengths and weaknesses of this approach. A better way to compute visibility would be to use the Z-buffering capabilities of SGI machines; unfortunately this impractical for the time being because the graphics libraries supplied by SGI cannot currently be loaded into RCDE, due to limitations of the Lucid Common Lisp compiler.

Formally, given a set of $N$ images, we define a visibility list $\mathcal{A}^k_{1\leq k\leq N}$ for each image and we rewrite the image energy of Equation 6 as

$$\mathcal{E}_I = \sum_{1\leq k\leq N} \mathcal{E}^k_I \quad , \tag{15}$$

$$\mathcal{E}^k_I = \sum_{(ij)\in\mathcal{A}^k} S(Pr^k(x_i,y_i,z_i),(Pr^k(x_{i+1},y_{i+1},z_{i+1})))/\sum_{1\leq i<n} L^k_{i,i+1} \quad . \tag{16}$$



(a) (b) (c) (d)

Figure 4: Three-stage optimization of a 3–D Network. (a) The object is hand-entered using RCDE. By default the vertex heights are that of the underlying terrain model. (b) The object is optimized using only the top view. The object matches the roof outline in the top view but not the lower one because the object is higher than the terrain. (c) The height of the vertices is computed approximately by searching a range of $z$ values while maintaining the shape of the object's projection in the top view. (d) The network's 3–D shape is further refined by simultaneously optimizing the $x$, $y$ and $z$ values of the vertices' positions. Its projections then match image features in both views, guaranteeing that the 3–D shape of the underlying objects has been recovered.

The optimization typically is a three step process and is illustrated by Figure 4:

1. We optimize a snake using a single image. Since a single view underconstrains the three degrees of freedom of the individual vertices, we fix one of them for each vertex. The $z$ value is fixed and only the $x$ and $y$ coordinates are allowed to change. As shown in Figure 4(b), at the end of this first step,

the network's projections match image features in the view that was used but not necessarily in any other view because the fixed $z$ values usually are erroneous.

2. The height of the vertices is estimated by taking the network to be horizontal and searching through a range of $z$ values, calculating the $x$ and $y$ values for each vertex so that the projection of the network remains the same in the view used in the previous step and retaining the $z$ value that yields the optimal value of $\mathcal{E}_I$, the image energy of Equation 16.

3. The 3–D positions of the network's vertices are further refined by optimizing $\mathcal{E}_I$ with respect to all three degrees of freedom of the vertices simultaneously.

As in the case of the 2–D networks, the optimization of Steps 1 and 3 can be performed using either steepest steepest gradient descent, conjugate gradient or constrained optimization.

The number of degrees of freedom of generic 3–D networks can be reduced by forcing them to be planar. We do this either by defining a plane of equation

$$z = ax + by + c \tag{17}$$

and imposing that the vertices lie on such a plane or imposing planar constraints on sets of four vertices. In both cases, we replace the $n$ degrees of freedom necessary to specify the elevation of each vertex by the three degrees of freedom required to define the plane.

These 3–D networks can be further specialized to handle objects that are of particular interest in urban environments: trihedral corners found on building roofs and extruded objects that are used in RCDE to model building outlines.

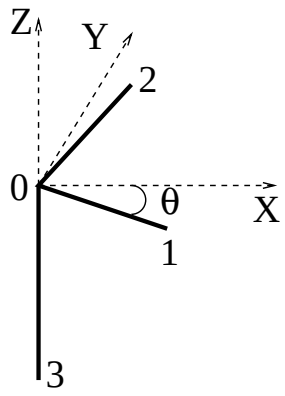
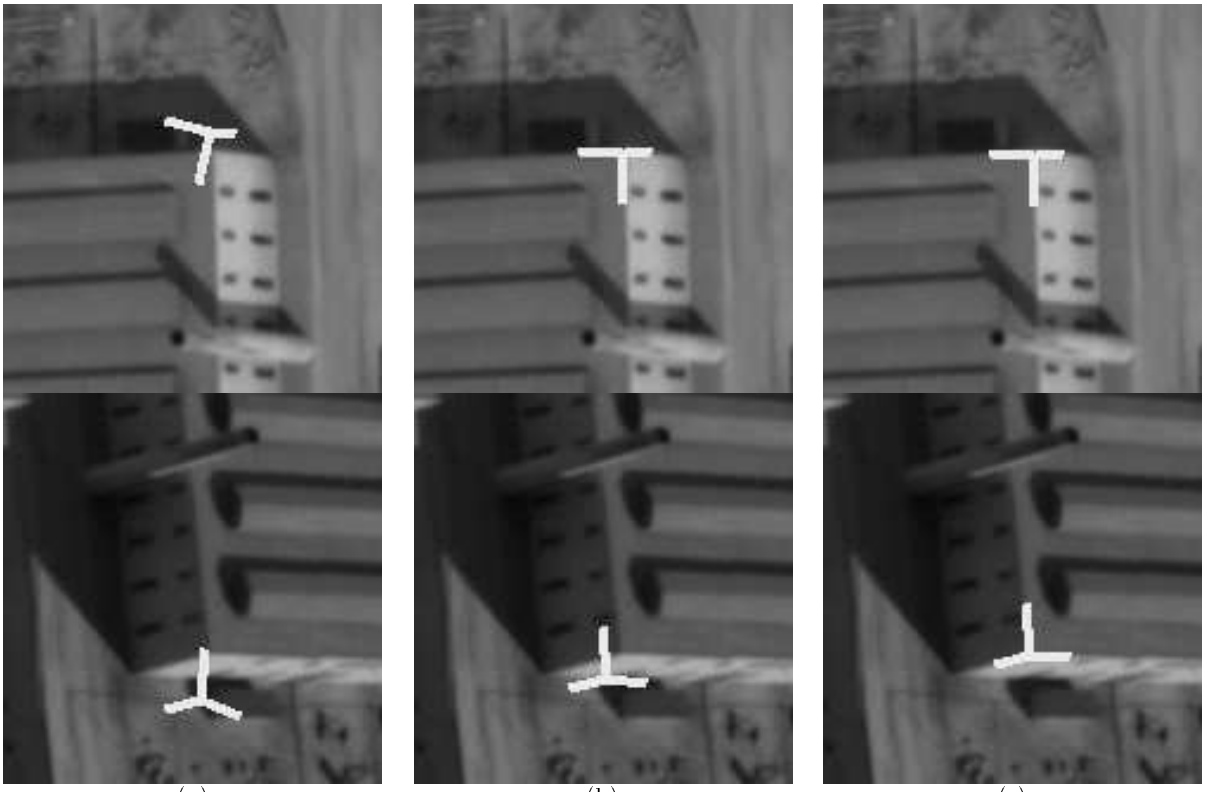



Figure 5: Topology of a trihedral corner. It has four vertices and three edges that all share one vertex. The edges form 90 degrees angles, Two of them may be constrained to be horizontal. In this case the corner has only four degrees of freedom, three for the position of vertex 0 and 1 for the orientation of the horizontal edges.

**Trihedral corners.** They are modeled as networks with four vertices and three edges forming 90 degrees angle with each other, as shown by Figure 5. We typically impose the additional constraint that one edge be vertical while the two other are horizontal. Under such constraints, the trihedral corner has only four degrees of freedom: three for the position of the vertex that is shared by all three edges and one for rotation about the vertical axis. When optimizing using only one image, fixing the altitude removes one additional degree of freedom. In both cases, the optimization is much more constrained than for generic 3–D networks and, as a result, the convergence properties are substantially improved.



(a) (b) (c)

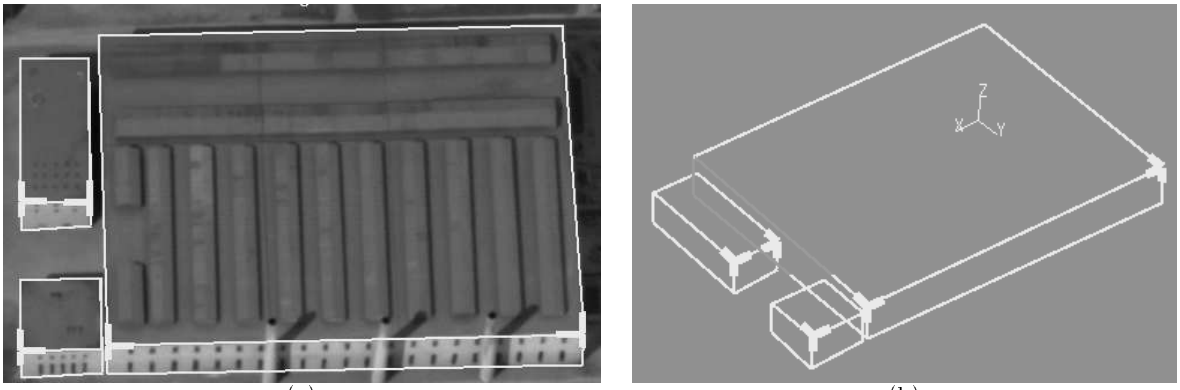
Figure 6: Optimization of a trihedral corner. (a) Initial position. (b) After optimization using only the top view, the corner's projection matches the image features in the top view only. (c) After optimization using both views, the corner's projections match the image features in both views.

In Figure 6, we show the recovery of such a trihedral corner. Figure 7 shows additional corners recovered and superimposed on a manually-entered 3–D wireframe models of the correspondng buildings. Because the corners are fully 3–dimensional objects, they can be viewed from different viewpoints in which they match the 3–D structure of the underlying objects.

Note that, in order to accurately recover the corner's 3–D position, the camera models associated with the images must be fairly precise—which they are in the examples presented here. However, if the camera models were less accurate, we could still perform the single-view optimization in each image separately. We could then feed the results of optimizing several corners to a resection program and refine the camera models.

**Extruded objects.** Extruded objects are typically used to model buildings such as those of Figure 7. For optimization purposes, we define extruded networks that are composed of a polygonal closed contour that corresponds to the roof outline and of vertical edges that correspond to the intersections of the vertical walls as shown in Figure 8. As discussed above (see Figure 3), for each view, $k$, in which the extruded object is visible, we define a list $\mathcal{A}^k$ of edges that are visible and use only those to compute the image energy $\mathcal{E}_I$.

During the optimization, we constrain the "wall" edges to remain vertical. We can also constrain the "roof-outline" to be planar and the "roof-edges" to form 90-degree angles. As in the case of 3–D corners, these constraints greatly reduce the number of degrees of freedom and allow for better convergence properties.



(a) (b)

Figure 7: Recovering building corners. (a) Several building corners superimposed on manually entered 3–D wireframe models of the buildings. (b) The same corners and wireframes seen from a different viewpoint. The recovered corners also are 3–D objects that match the underlying objects.

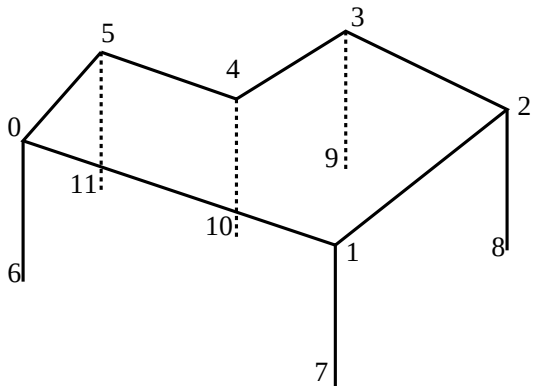
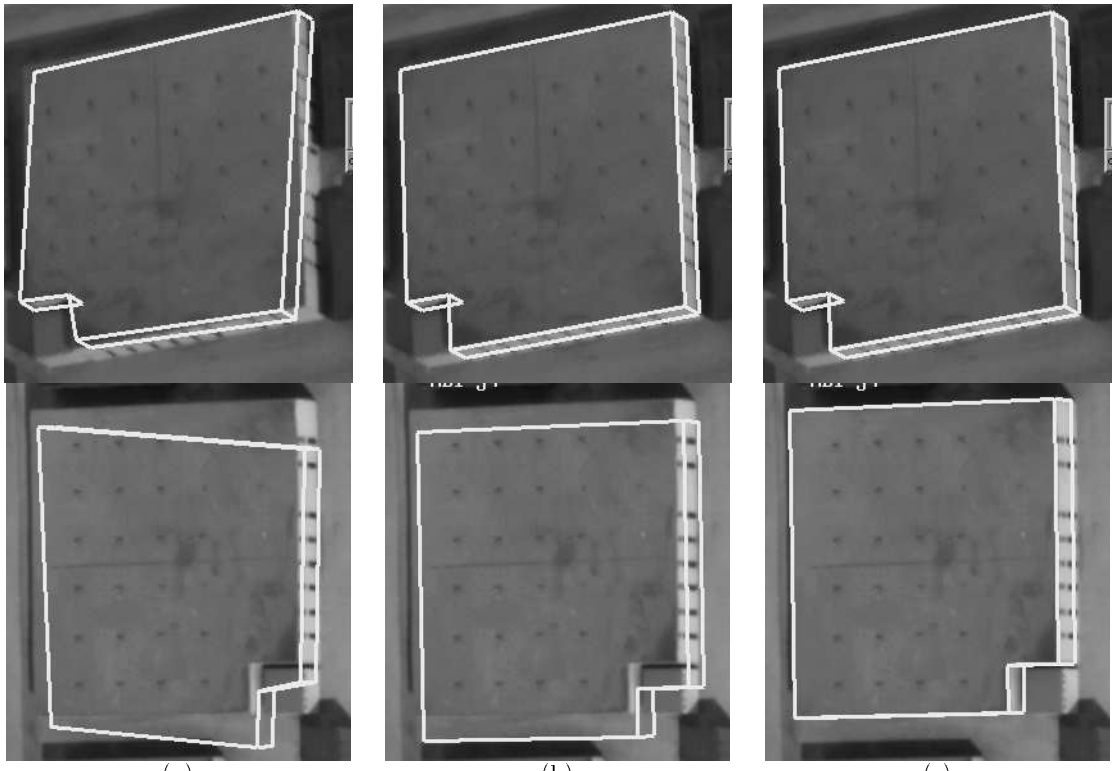



Figure 8: Topology of an extruded object. It has a polygonal outline that corresponds to the roof outline and vertical edges that correspond to the intersections of the vertical walls. In this example the complete edge-list is ((0 1) (1 2) (2 3) (3 4) (4 5) (5 0) (0 6) (1 7) (2 8) (3 9) (4 10) (5 11)). Note, however, that, due to occlusions, the list of visible edges for the particular projection shown here would be be the sublist ((0 1) (1 2) (2 3) (3 4) (4 5) (5 0) (0 6) (1 7) (2 8)). The visible edges are shown as solid lines and the hidden ones as dashed lines.

Figure 9 illustrates the recovery of a building using all the constraints described above. For comparison's sake, in Figure 10, we show the result of the optimization using the same starting point but without imposing the rectilinearity constraint.

In Figure 11, we show several buildings modeled by roughly entering their outlines within RCDE and optimizing the shapes in three views simultaneously using our extruded snakes . The use of the snakes has allowed us to perform this task much faster than we would have if we had had to precisely delineate all five buildings by hand.



(a) (b) (c)

Figure 9: Optimization of an extruded object. (a) Initial position. (b) The edges are assumed to form 90 degrees angles. After optimization using only the top view, the object's projection matches the image features in the top view only. (c) After optimization using both views, the object's projections match the image features in both views.

# 3 3–D Surface Reconstruction

Given the task of reconstructing a surface from multiple images whose vantage points may be very different, we need a surface representation that can be used to generate images of the surface from arbitrary viewpoints, taking into account self-occlusion, self-shadowing, and other viewpoint-dependent effects. Clearly, a single image-centered representation is inadequate for this purpose. Instead, an object-centered surface representation is required.

Many object-centered surface representations are possible. However, practical issues are important in choosing an appropriate one. First, the representation should be general-purpose in the sense that it should be possible to represent any continuous surface, closed or open, and of arbitrary genus. Second, it should be relatively straightforward to generate an instance of a surface from standard data sets such as depth maps or clouds of points. Finally, there should be a computationally simple correspondence between the parameters specifying the surface and the actual 3-D shape of the surface, so that images of the surface can be easily generated, thereby allowing the integration of information from multiple images.

A regular 3–D triangulation is an example of a surface representation that meets the criteria stated above, and is the one we have chosen for our previous work. In our implementation, all vertices except those on the edges have six neighbors and are initially regularly spaced. Such a mesh defines a surface composed of three-sided planar polygons that we call triangular facets, or simply facets. Triangular facets are particularly

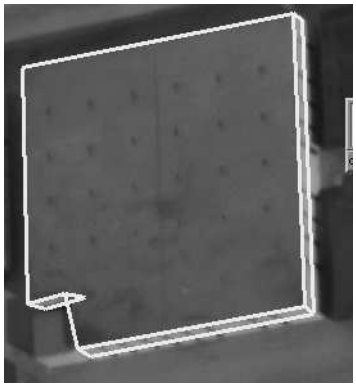


Figure 10: Extruded object of Figure 9 optimized without imposing the constraint that the roof-edges form 90-degree angles. The initial position was the one shown in Figure 9(a). Note that the corner in the lower left corner is not properly recovered.

easy to manipulate for image and shadow generation; consequently they are the basis for many 3-D graphics systems. These facets tend to form hexagons and can be used to construct virtually arbitrary surfaces. Finally, standard triangulation algorithms can be used to generate such a surface from noisy real data [Fua and Sander, 1992, Szeliski and Tonnesen, 1992].

**Sources of information.** A number of information sources are available for the reconstruction of a surface and its material properties. Here, we consider two classes of information.

The first class comprises those information sources that do not require more than one image, such as texture gradients, shading, and occlusion edges. When using multiple images and a full 3-D surface representation, however, we can do certain things that cannot be done with a single image. First, the information source can be checked for consistency across all images, taking occlusions into account. Second, when the source is consistent and occlusions are taken into account, the information can be fused over all the images, thereby increasing the accuracy of the reconstruction.

The second class comprises those information sources that require at least two images, such as the triangulation of corresponding points between input images (given camera models and their relative positions). Generally speaking, this source is most useful when corresponding points can be easily identified and their image positions accurately measured. The ease and accuracy of this correspondence can vary significantly from place to place in the image set, and depend critically on the type of feature used. Consequently, whatever the type of feature used, one must be able to identify where in the images that feature provides reliable correspondences, and what accuracy one can expect.

The image feature that we have chosen for correspondence (although it is by no means the only one possible) is simply intensity in radiometrically corrected images, for example by filtering them. Clearly, intensity can be a reliable feature only when the albedo varies quickly enough on the surface and, consequently, the images are sufficiently textured.

In contrast to our approach, simple correlation-based stereo methods often use fixed-size windows in images to measure disparities, which will in general yield correct results only when the surface is parallel to the image plane. Instead, we compare the intensities as projected onto the facets of the surface. Consequently, the reconstruction can be significantly more accurate for slanted surfaces Some correlation-based algorithms achieve similar results by using variable-shaped windows in the images [Quam, 1984, Nishihara, 1984, Kanade and Okutomi, 1990, Baltsavias, 1991, Devernay and Faugeras, 1994]. However, they typically use only image-centered representations of the surface.

As for the monocular information source, we have chosen to use shading, where shading is the change in

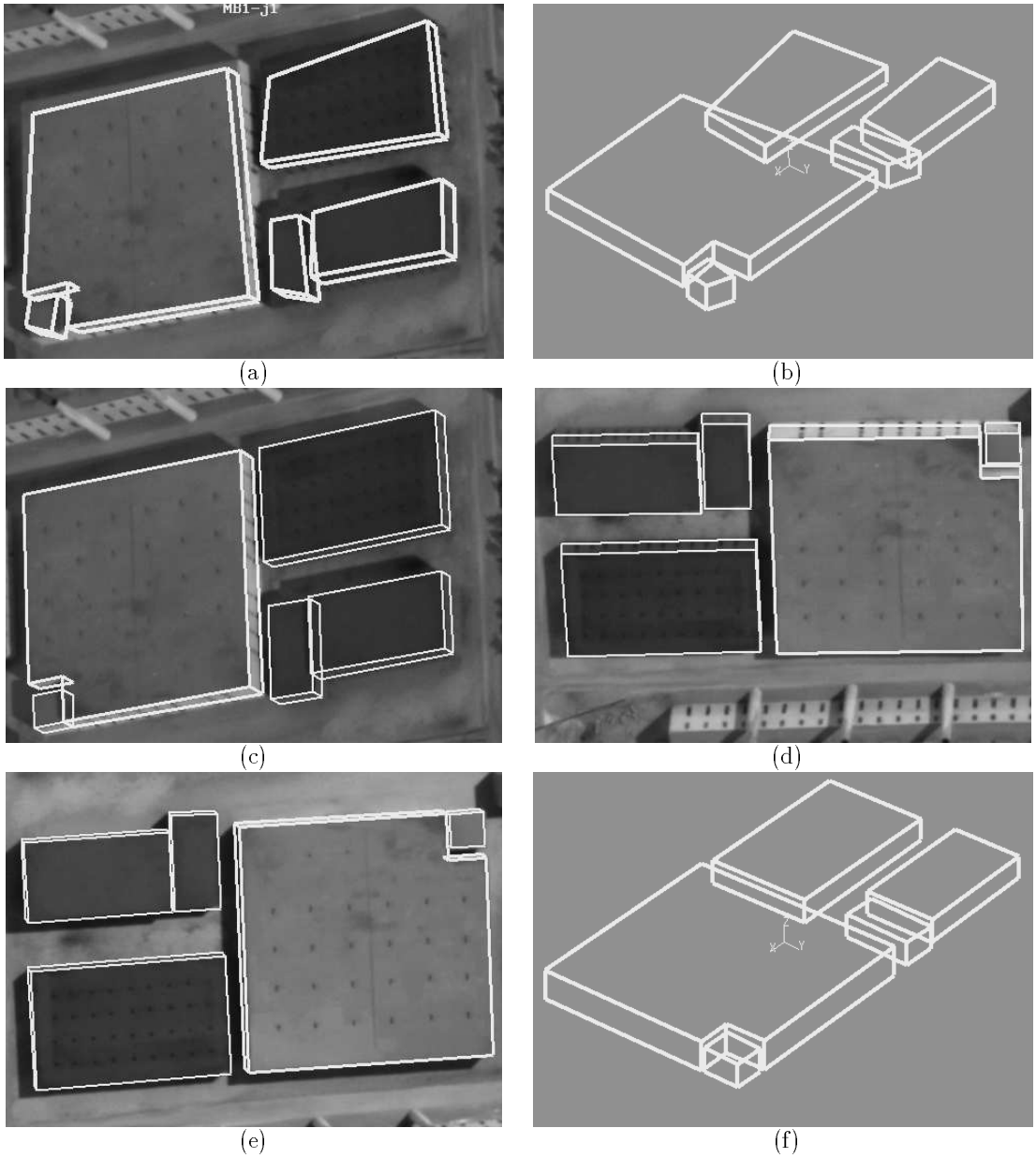



Figure 11: Buildings modeled by entering rough models within RCDE and optimizing them using the extruded-snakes. (a) Rough initial sketches overlaid on one of the images. (b) A view from a different perspective. (c,d,e) Final building outlines overlaid on the three images we used to perform the 3–D optimization. (f) A view of the buildings from the perspective of (b).

image intensity due to the orientation of the surface relative to a light source. The main reason for this is the fact that shading is most reliable when the albedo varies slowly across the surface; this is the natural complement to intensity correspondence, which requires quickly varying albedo. The complementary nature of these two sources allows us to accurately recover the surface geometry and material properties for a wide

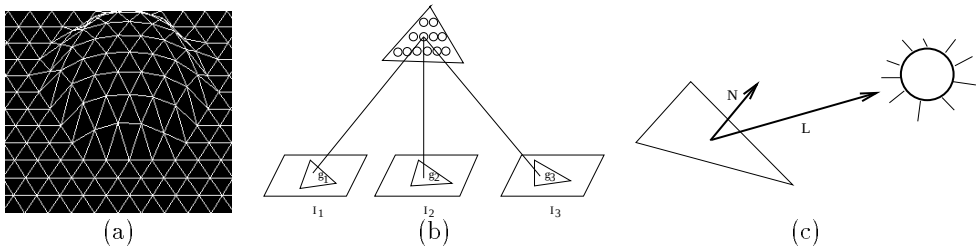



Figure 12: Mesh representation and computation of the image terms of the objective function: (a) Wireframe representation of the mesh. (b) Facets are sampled at regular intervals; the circles represent the sample points. The stereo component of the objective function is computed by summing the variance of the gray level of the projections of these sample points, the $g_i$s. (c) Each facet's albedo is estimated using its normal $N$, the light source direction $L$, and the average gray level of the projection of the facet into the images. The shading component of the objective function is the sum of the squared differences in estimated albedo across neighboring facets.

variety of images.

In contrast to our approach, traditional uses of shading information assume that the albedo is constant across the entire surface, which is a major limitation when applied to real images. We overcome this limitation by improving upon a method to deal with discontinuities in albedo alluded to in the summary of [Leclerc and Bobick, 1991]. We compute the albedo at each facet using the normal to the facet, a light-source direction, and the average of the intensities projected onto the facet from all images. We use the local variation of this computed albedo across the surface as a measure of the correctness of the surface reconstruction. To see why albedo variation is a reasonable measure of correctness, consider the case when the albedo of the real surface is constant. When the geometry of the mesh is correct, then the computed albedo should be approximately the same as the real albedo, and hence should be approximately constant across the mesh. Thus, when the geometry is incorrect, this will generally give rise to variations in the computed albedo that we can take advantage of. Furthermore, by using a *local* variation in the computed albedo, we can deal with surfaces whose albedo is not constant, but instead varies slowly over the surface.

**Current implementation.** The triangulated 3–D mesh of vertices that represents a surface, $\mathcal{S}$, is a hexagonally connected set of vertices such as the one shown in Figure 12(a). The position of a vertex $v_j$ is specified by its Cartesian coordinates $(x_j, y_j, z_j)$. The mesh can be deformed by varying these coordinates to minimize an objective function that includes terms derived from stereo and shading information.

The stereo component of the objective function is derived by comparing the gray levels of the points in all of the images for which the projection of a given point on the surface is visible. As shown in Figure 12(b), this comparison is done for a uniform sampling of the surface. This method allows us to deal with arbitrarily slanted regions and to discount occluded areas of the surface.

The shading component of the objective function is computed using a method that does not invoke the traditional constant albedo assumption. Instead, it attempts to minimize the variation in albedo across the surface, and can therefore deal with surfaces whose albedo varies slowly. This term is depicted by Figure 12(c).

The stereo term is most useful when the surfaces are highly textured. Conversely, the shading term is most reliable where the surfaces have little or no texture. To account for this phenomenon, we take the complete objective function, $\mathcal{E}(\mathcal{S})$, to be a weighted average of these two components where the weighting is a function of texture within the projections of individual facets.

In general, $\mathcal{E}(\mathcal{S})$ is a highly nonconvex function of the vertex positions. To minimize $\mathcal{E}(\mathcal{S})$, we use the "snake-type" [Kass *et al.*, 1988] optimization technique described in Section 2.2. We define the total energy

of the mesh, $\mathcal{E}_T(\mathcal{S})$, as

$$\mathcal{E}_T(\mathcal{S}) = \lambda_D \mathcal{E}_D(\mathcal{S}) + \mathcal{E}(\mathcal{S}) \tag{18}$$

where $\lambda_D$ is a weighting coefficient that decreases as the optimization proceeds and $\mathcal{E}_D(\mathcal{S})$ is the regularization term. In practice, we take $\mathcal{E}_D$ to be a measure of the curvature or local deviation from a plane at every vertex. Because the mesh is regular, $\mathcal{E}_D$ can be approximated using finite differences as a quadratic form [Fua and Leclerc, 1995a]

$$\mathcal{E}_D(\mathcal{S}) = 1/2(X^T K X + Y^T K Y + Z^T K Z) \ , \tag{19}$$

where $X$,$Y$, and $Z$ are the vectors of the $x$,$y$ and $z$ coordinates of the vertices, and $K$ is a sparse and banded matrix. This regularization term serves a dual purpose. First, it "convexifies" the energy landscape when $\lambda_D$ is large and improves the convergence properties of the optimization procedure. Second, in the presence of noise, some amount of smoothing is required to prevent the mesh from overfitting the data, and wrinkling the surface excessively.

To speed the computation and prevent the mesh from becoming stuck in undesirable local minima, we typically use several levels of mesh size—three in the examples presented here—to perform the computation. We start with a relatively coarse mesh that we optimize. We then refine it by splitting every facet into four smaller ones and reoptimizing. Finally, we repeat the split and optimization processes one more time.

Our specific approach has led to a number of important contributions:

- Our framework can incorporate cues from many images, even if taken from widely differing viewpoints. It accommodates such viewpoint-dependent effects as self-occlusion and self-shadowing.
- Our technique for doing stereo avoids the constant depth assumption of traditional correlation-based stereo algorithms. As the Hierarchical Warp Stereo System [Quam, 1984], it effectively uses variable-sized windows in the images but does it in a more generic fashion.
- Our approach to shape from shading is applicable to surfaces with slowly varying albedo. This is a significant advance over traditional approaches that require constant albedo.
- We have proposed a dynamic weighting scheme for combining shape from shading and stereo, and demonstrated that it leads to significantly better results than using either cue alone.

With this purely image-based approach, we have obtained good results on complex surfaces such as the jagged terrain shown in Figure 13. We have also evaluated the performance of our procedure against the "ground truth" supplied to us by a photogrammetrist from Ohio State University for the images of Figure 14. In this example, we initialize a coarse resolution mesh by interpolating a correlation map derived using the images reduced by a factor of 4. We first apply our continuation method to this coarse mesh using the stereo component of the objective function. Next, we increase the resolution of both the images and the mesh, reoptimize, and repeat the process once more. At each level of resolution, as the regularization term is progressively turned down, the discrepancy between our surface model and the control points diminishes. In Figure 14(e), we plot the distance of the control points to the surface at the end of each optimization step. The final error at each level of resolution, denoted by the thick vertical lines, corresponds to an error in measured disparity that is smaller than half a pixel. Given the fact that the control points are not necessarily perfect themselves, this is the kind of performance one would expect of a precise stereo system [Güelch, 1988].

Furthermore, at higher resolutions, we have also shown [Fua and Leclerc, 1995b], that our approach can also take advantage of the geometric constraints derived from measured 3–D points and 2–D silhouettes, thereby making the reconstruction more robust.

# 4 Conclusion

We have presented object modeling techniques for 2–D and 3–D linear features as well as 3–D surfaces that rely on analogous parametric models that are extensions of traditional snakes. Using a variety of real imagery,

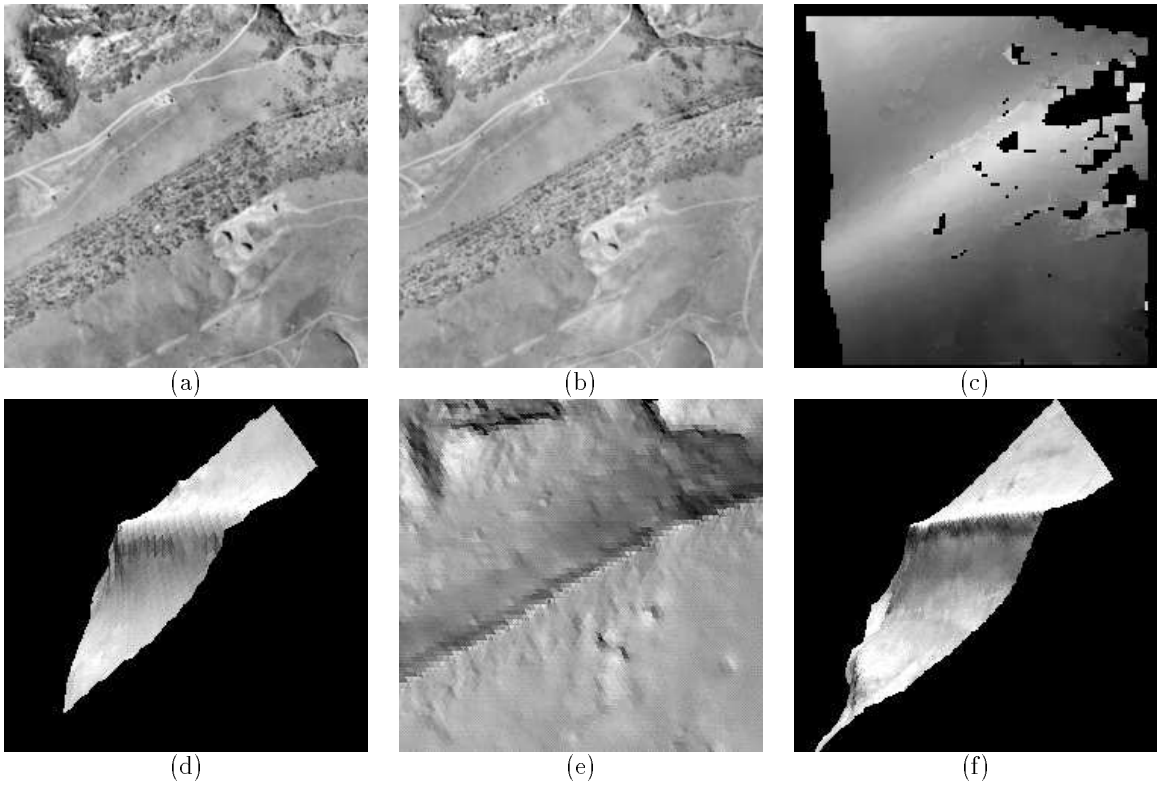



Figure 13: (a,b) A stereo pair of images of the Martin-Marietta Autonomous Land Vehicle (ALV) test site. (c) Disparity map computed using a correlation-based algorithm. The black areas indicate that the stereo algorithm could not find a match. Elsewhere, lighter grays indicate higher elevations. (d) The initial surface estimate derived by smoothing and interpolation of the disparity map. It is shown as a shaded surface viewed by an observer located above the upper left corner of the scene. (e,f) Shaded views of the mesh after optimization. Note that the ridge has become very sharp and that the shadow-casting cliffs visible in the top portion of the image are recovered. They are clearly visible at the top of (e) and the bottom right corner of (f).

we have demonstrated that the resulting methods allow powerful and flexible reconstruction. However in their current form, they suffer from two limitations:

- All these methods are optimization based and require a reasonably good starting point.
- At present all objects are optimized independently.

In future work, we intend to explore the use of more powerful search techniques to alleviate the first problem to provide our system with a repertoire of constraints that can be used to tie the objects together and guide the optimization.

# References

[Baltsavias, 1991] E. P. Baltsavias. *Multiphoto Geometrically Constrained Matching.* PhD thesis, Institute for Geodesy and Photgrammetry, ETH Zurich, December 1991.

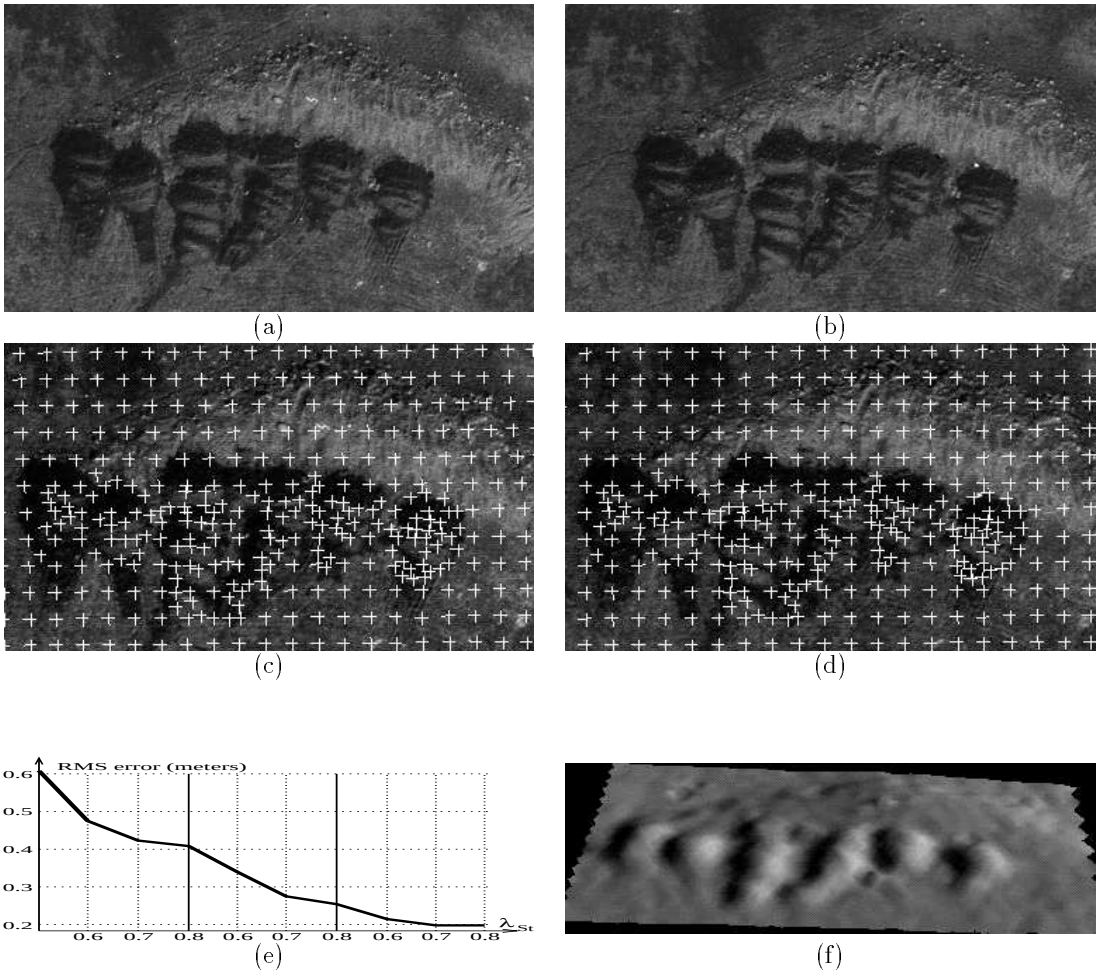



Figure 14: (a,b) An aerial stereo pair (Courtesy of Ohio State University). (c,d) Matched pair of points hand-entered by a photogrammetrist. (e) Plot of the distance, in meters, of the control points of Figure 14(c,d) to the surface as the optimization proceeds. The thick vertical lines indicate a change in resolution and the dotted ones an increase in weighting of the stereo component of the objective function. At the highest resolution, an elevation error of 0.2 meters corresponds to an error of approximately 0.4 pixels in disparity. (f) Reconstructed surface at the highest level of resolution.

[Brechbühler *et al.*, 1995] C. Brechbühler, G. Gerig, and O. Kübler. Parametrization of closed surfaces for 3-d shape description. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 61(2):154–170, March 1995.

[Brechbühler, 1995] C. Brechbühler. *Description and Analysis of 3-D Shapes by Parametrization of Closed Surfaces*. PhD thesis, ETH Zurich, Rämistrasse 101, CH-8092 Zürich, 1995. Diss. ETH No. 10979.

[Devernay and Faugeras, 1994] F. Devernay and O. D. Faugeras. Computing Differential Properties of 3–D Shapes from Stereoscopic Images without 3–D Models. In *Conference on Computer Vision and Pattern Recognition*, pages 208–213, Seattle, WA, June 1994.

[Fua and Leclerc, 1990] P. Fua and Y. G. Leclerc. Model Driven Edge Detection. *Machine Vision and Applications*, 3:45–56, 1990.

[Fua and Leclerc, 1995a] P. Fua and Y. G. Leclerc. Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading. *International Journal of Computer Vision*, September 1995. Available as Tech Note 535, Artificial Intelligence Center, SRI International.

[Fua and Leclerc, 1995b] P. Fua and Y. G. Leclerc. Taking Advantage of Image-Based and Geometry-Based Constraints to Recover 3–D Surfaces. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 1995. Accepted for publication, available as Tech Note 536, Artificial Intelligence Center, SRI International.

[Fua and Sander, 1992] P. Fua and P. Sander. Segmenting Unstructured 3D Points into Surfaces. In *European Conference on Computer Vision*, Genoa, Italy, April 1992.

[Gill *et al.*, 1981] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, London a.o., 1981.

[Güelch, 1988] E. Güelch. Results of Test on Image Matching of ISPRS WG III / 4. *International Archives of Photogrammetry and Remote Sensing*, 27(III):254–271, 1988.

[Kanade and Okutomi, 1990] T. Kanade and M. Okutomi. A Stereo Matching Algorithm with an Adaptative Window: Theory and Experiment. In *ARPA Image Understanding Workshop*, September 1990.

[Kass *et al.*, 1988] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[Leclerc and Bobick, 1991] Y. G. Leclerc and A. F. Bobick. The direct computation of height from shading. In *Conference on Computer Vision and Pattern Recognition*, Lahaina, Maui, Hawaii, June 1991.

[Mundy *et al.*, 1992] J.L. Mundy, R. Welty, L. Quam, T. Strat, W. Bremmer, M. Horwedel, D. Hackett, and A. Hoogs. The RADIUS Common Development Environment. In *ARPA Image Understanding Workshop*, pages 215–226, 1992.

[Nishihara, 1984] H.K. Nishihara. Practical Real-Time Imaging Stereo Matcher. *Optical Engineering*, 23(5), 1984.

[Quam, 1984] L.H. Quam. Hierarchical warp stereo. In *ARPA Image Understanding Workshop*, pages 149–155, 1984.

[Szeliski and Tonnesen, 1992] R. Szeliski and D. Tonnesen. Surface Modeling with Oriented Particle Systems. In *Computer Graphics (SIGGRAPH)*, volume 26, pages 185–194, July 1992.

[Terzopoulos *et al.*, 1987] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models and 3D object reconstruction. *International Journal of Computer Vision*, 1:211–221, 1987.