Fast, Accurate and Consistent Modeling of Drainage and Surrounding Terrain

P. Fua*
Computer Graphics Lab (LIG), CH-1015 Lausanne, Switzerland
and
SRI International, Menlo Park, CA 94025

Abstract

We propose an automated approach to modeling drainage channels—and, more generally, linear features that lie on the terrain—from multiple images. It produces models of the features and of the surrounding terrain that are accurate and consistent and requires only minimal human intervention.

We take advantage of geometric constraints and photommetric knowledge. First, rivers flow downhill and lie at the bottom valleys whose floors tend to be either V- or U-shaped. Second, the drainage pattern appears in gray-level images as a network of linear features that can be visually detected.

Many approaches have explored individual facets of this problem. Ours unifies these elements in a common framework. We accurately model terrain and features as 3–dimensional objects from several information sources that may be in error and inconsistent with one another. This approach allows us to generate models that are faithful to sensor data, internally consistent and consistent with physical constraints. We have proposed generic models that have been applied to the specific task at hand. We show that the constraints can be expressed in a computationally effective way and, therefore, enforced while initializing the models and then fitting them to the data. Furthermore, these techniques are general enough to work on other features that are constrained by predictable forces.

*This work was conducted at SRI International and supported in part by contracts from the Defense Advanced Research Projects Agency.

1 Introduction

We propose an automated approach to modeling drainage channels—and, more generally, linear features that lie on the terrain—from multiple images, which results not only in high-resolution, accurate and consistent models of the features, but also of the surrounding terrain.

This is an important problem from both a practical point of view—drainage modeling is an essential component of map making—and a theoretical point of view: We must address two key generic problems. The first is the obvious requirement to replace reliance on generally unavailable prior knowledge of explicit shape with more general ways of recognizing and describing natural objects. The second is the necessity to merge several sources of information that may not be consistent with one another.

In our specific case, we have chosen to exploit the fact that

- Rivers flow downhill and lie at the bottom of local depressions perpendicular to the stream's direction.
- Valley floors tend to be "V" or "U" shaped and locally horizontal in the direction perpendicular to the main valley at the river's location.
- The drainage pattern appears as a network of linear features that can be visually detected in single gray-level images.

Different approaches have explored individual facets of this problem. There is extensive literature on the extraction of valleys from terrain models, for example see [O'Callaghan and Mark, 1984, Band, 1986, Fairfield and Leymarie, 1991] among many others. The terrain model, however, is almost always assumed to be error-free, which, in practice, only rarely is the case. Furthermore, Koenderink and Van Doorn have shown [1993] that the strictly local differential criteria many of these systems use to detect valleys have inherent problems that must be addressed using a more global approach. Much work has also been devoted to the extraction of linear patterns from single images using techniques such as dynamic programming [Fischler and Wolf, 1983, Merlet and Zerubia, 1995] or graph-based techniques [Fischler *et al.*, 1981]. These techniques typically do not use the terrain information or guarantee that the recovered drainage pattern satisfies the physical constraints discussed above. Furthermore they do not take advantage of the fact that multiple images of the same site may be available.

Our approach unifies these elements in a common framework. Because the features and the physical constraints we handle are fundamentally 3–D and because we want to be able to deal with an arbitrary number of images, there are very significant advantages in using an object-centered 3–D representation of the terrain surface. We therefore take advantage of the Model-Based Optimization (MBO) paradigm that we have developed in earlier work [Fua, 1996] to express geometric, photometric and physical properties of the features of interest and to enforce hard constraints among these features.

We have chosen to concentrate on the extraction and the refinement of drainage patterns because they are potentially complex but obey well-understood physical constraints and therefore constitute a very good test case for our research. However, we will argue that the same techniques are robust enough to work on other linear features that are constrained by predictable forces. For example, roads typically comply with known engineering limits for slope, side slope, radius of curvature, and so forth. Mountain ridges exhibit well-known differential properties that are comparable to those satisfied by river valleys [Koenderink and van Doorn, 1993].

We view the contribution of this paper as proposing a general approach to accurately modeling terrain and features from several information sources that may be in error and inconsistent with one another. This approach allows us to generate models that are faithful to sensor data, internally consistent and consistent with physical constraints. We have proposed generic models that have been applied to the specific task at hand—river delineation and digital elevation model (DEM) refinement—and shown that the constraints can be expressed in a computationally effective way and, therefore, enforced while initializing the models and then fitting them to the data.

We first introduce our overall framework. We then review our approach to modeling the terrain and estimating its curvature and present the techniques we use to quickly sketch the drainage pattern and to automatically enforce the consistency constraints. Finally, we evaluate our results against different kinds of "ground truth."

2 Approach

We model the terrain as a triangulated mesh that can be refined by minimizing an objective function. The resolution of the mesh is chosen so that riverbeds are a few facets wide and we can represent the rivers' centerline as polygonal paths

- that lie on the terrain surface,
- that are located where the largest principal curvature of the terrain surface is locally maximal in the direction normal to the path,
- whose tangent vectors are the directions of maximal elevation decrease,
- whose altitude decreases monotonically.

We start by recovering the approximate shape of a terrain mesh by minimizing a multi-image stereo score [Fua and Leclerc, 1995] and computing a curvature map. From this map, we extract paths of maximal curvature using dynamic programming. This simple approach is depicted by Figure 1(a). It would be sufficient if the recovered terrain surface was perfect and if the terrain was steep enough for all streams to flow at the bottom of the sort of "V" shaped valleys that erosion produces [Gilluly *et al.*, 1968]. In practice, this is

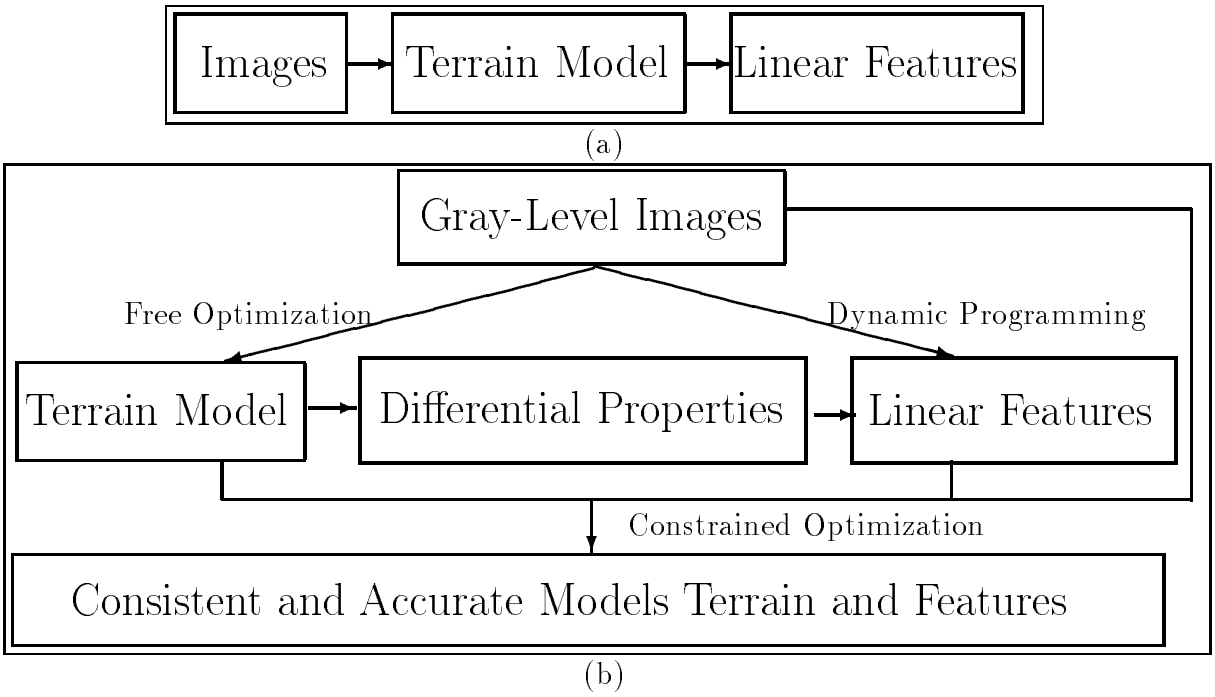


Figure 1: Approaches to 3–D delineation. (a) Simple approach: The features are extracted directly from the terrain model. If there is not enough relief or if the terrain model is not precise enough, the resulting features may be poorly located or inconsistent. For example, a river may seem to flow uphill or on the side of a hill. (b) Refined approach: First, the terrain, its differential properties and the actual gray-level images are all used to compute the location of the features. Then, the terrain and features are refined under consistency constraints that prevent problems such as the ones described here. Finally, as new images become available, terrain and features can be further refined.

not always the case. There may not be enough relief to tell the real but shallow valleys from spurious valleys that may be present in the recovered terrain surface. Furthermore, even if there are deep and easy-to-detect "V" shaped valleys, vegetation tends to be taller on river banks, thus making the elevations computed by our surface-reconstruction algorithm unreliable. As a result, the recovered path may not follow the true valley bottom and may not exhibit monotonically decreasing elevations.

To solve these problems, we have developed the approach depicted by Figure 1(b):

- We use both the terrain model and the actual gray-level images to extract a rough estimate of the features' locations, thus preventing the estimate from being too far off if there are errors in the terrain model.

- We simultaneously refine the models of the terrain and features under consistency constraints that ensure that they fit the image data as well as possible, while conforming to the physical constraints known to apply.

Using images from a number of different sites, we will show that this technique allows the quick generation of consistent 3–D models of the drainage channels and the surrounding terrain with minimal manual intervention. We will also show that enforcing consistency does not detract from the accuracy of the reconstruction.

3 Terrain Modeling and Curvature Estimation

Many object-centered surface representations could be used to represent the terrain. However, practical issues are important in choosing an appropriate one. First, it should be relatively straightforward to generate an instance of a surface from standard data sets such as depth maps or digital elevation models. Second, 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. Finally, it should be natural to express the geometric constraints inherent to the problem we are attempting to solve.

A regular 3–D triangulation such as the one shown in Figure 2(c,d) is an example of a surface representation that meets the criteria stated above, and is the one we have chosen for our previous work [Fua and Leclerc, 1995]. 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. These facets tend to form hexagons and can be used to construct arbitrary surfaces

3.1 Recovering the Shape of the Terrain

The shape of a mesh $\mathcal{S}$ is defined by the position of its vertices. It can be refined by minimizing a regularized objective function that accounts for the stereo information present in multiple images of a cartographic site to produce models such as the one shown in Figures 2(e) and 3(c). In other words, this technique uses 3–D triangulations not only as a representational tool but also as a computational one.

The objective function $\mathcal{E}(\mathcal{S})$ is taken to be

$$\mathcal{E}(\mathcal{S}) = \mathcal{E}_D(\mathcal{S}) + \mathcal{E}_{St}(\mathcal{S}) \quad , \tag{1}$$

where $\mathcal{E}_D(\mathcal{S})$ is a regularization term that is quadratic in terms of the vertices' coordinates and $\mathcal{E}_{St}(\mathcal{S})$ is a multiple-image correlation term. It is derived by comparing the gray

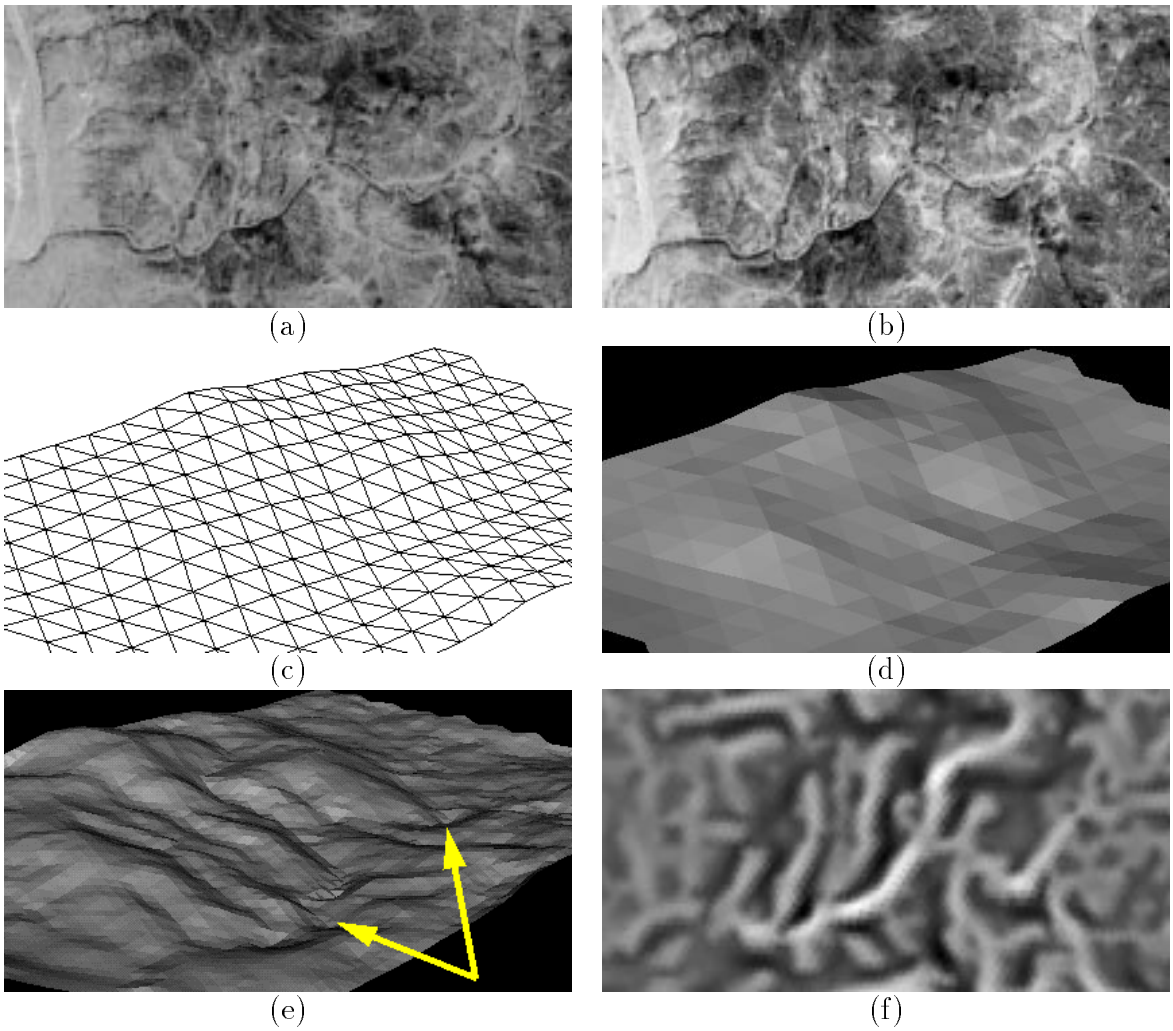


Figure 2: Terrain modeling at the National Training Center (NTC), Ft. Irwin. (a,b) A stereo pair of a hilly site. The linear structure that runs horizontally in the middle of the images is a streambed and is indicated by the cluster of arrows at the bottom of the figure. A second streambed runs vertically from the top of the image into the first one. (c) One of the hexagonally triangulated meshes we use for surface reconstruction shown as a wireframe. (d) A shaded view of the same mesh. (e) A shaded view of the mesh after subdivision and optimization. The two arrows point at the valley of the horizontal streambed shown in (a). The second streambed is hidden behind one of the hills. (f) The curvature image registered to the image shown in (a). Regions of high positive curvature—that is, candidate valley regions—are shown in white.

levels of the points in all the images for which the projection of a given point on the

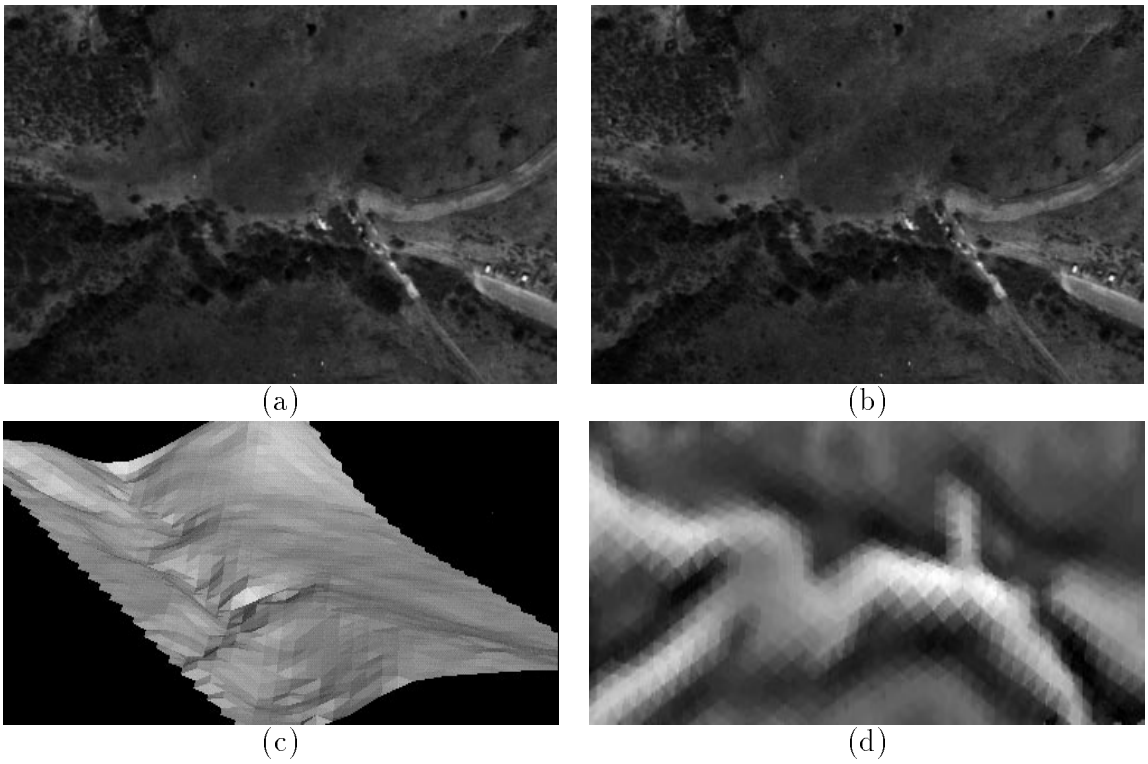


Figure 3: Terrain modeling at the Martin-Lockheed UGV site, Denver. (a,b) A stereo pair of with two streambeds forming a Y. (c) A shaded view of the terrain mesh recovered by our algorithm. (d) The curvature image corresponding to image (a) and computed using the mesh shown in (c).

surface is visible. This comparison is done for a uniform sampling of the surface. $\mathcal{E}_{St}$ is closely related to the terms used by Wrobel [1991] and Heipke [1992] in their least-squares approaches. This method allows us to deal with arbitrarily slanted regions and to discount occluded areas of the surface. For more details, we refer the interested reader to our previous publication [Fua and Leclerc, 1995].

$\mathcal{E}(\mathcal{S})$, the total energy of Equation 1 is a sum of two terms whose magnitudes are respectively geometry- and image-dependent and are therefore not necessarily commensurate. One therefore needs to scale them appropriately. The dynamics of the optimization are controlled by the gradient of the objective function. As a consequence, we have found that an effective way to normalize the contributions of the various components is to multiply them by constant weights computed so that the ratios of the gradients of $\mathcal{E}_D(\mathcal{S})$ and $\mathcal{E}_{St}(\mathcal{S})$ has a given value—the same for all examples shown in this paper—at the beginning of the optimization [Fua and Leclerc, 1990, Fua and Leclerc, 1996].

In our application, we fix the x and y coordinates of the vertices of $\mathcal{S}$, and the free variables are the z coordinates. The process is started with an initial estimate of the elevations typically derived from a coarse DEM. Because $\mathcal{E}_D$ is quadratic, these meshes are amenable to "snake-like" optimization [Kass *et al.*, 1988, Fua and Leclerc, 1996]. In the course of the optimization, we progressively refine the mesh by iteratively sub dividing the facets into four smaller ones whose sides are still of roughly equal length, thus preserving the regularity of the mesh.

3.2 Differential Properties of the Terrain Surface

In Section 4, we will show that we can combine the differential properties of the terrain surface—specifically, its largest principal curvature—with the information present in the gray-level images to automate the delineation of the drainage pattern. It is therefore important to be able to represent both kinds of information in a common frame of reference. In our application, we deal with near-vertical aerial imagery and we either use an orthophoto or the vertical-most available image.

We use the largest principal curvature rather than the mean or Gaussian curvature because we expect valley bottoms to be be curved in one direction and horizontal in the other. As suggested by Koenderink and Van Doorn [1991], we could also use the ratio of largest to smallest principal curvature. For our specific application, we have found this latter choice to yield fairly similar results to those presented here.

Following Sander and Zucker [1990], we estimate the maximal curvature at each vertex of the surface by fitting a quadric to the vertices in the neighborhood of that vertex [Lengagne *et al.*, 1996]. For each point on the surface, we then take the curvature to be a weighted average of the curvatures of the three vertices of the facet to which it belongs. We have found experimentally that this approach to estimating curvature is slower but more stable than other methods such as the one proposed by Taubin [1995], mainly because the quadric fitting is expensive but introduces a much-needed element of smoothing.

More specifically, the altitude z of vertex $V(x, y, z)$ is approximated by

$$z(x, y) = ax^2 + bxy + cy^2 + dx + ey + f \quad .$$

The tangent plane to the surface is defined by the two vectors $\vec{v_1} = \dfrac{\partial V}{\partial x}$ and $\vec{v_2} = \dfrac{\partial V}{\partial y}$. The normal to the tangent plane is defined as $\vec{n} = \vec{v_1} \wedge \vec{v_2}$. We compute the matrices of the two fundamental forms of the surface Φ_1 and Φ_2 and the matrix of the Weingarten endomorphism $W = -\Phi_1^{-1}\Phi_2$. The largest eigenvalue of W is an estimate of the maximal curvature at vertex V.

Using this method and given a surface triangulation, we can compute, for each original gray-level image, a "curvature image" that is registered with it such as the ones shown in Figures 2(f) and 3(d).

4 Automating Drainage Delineation

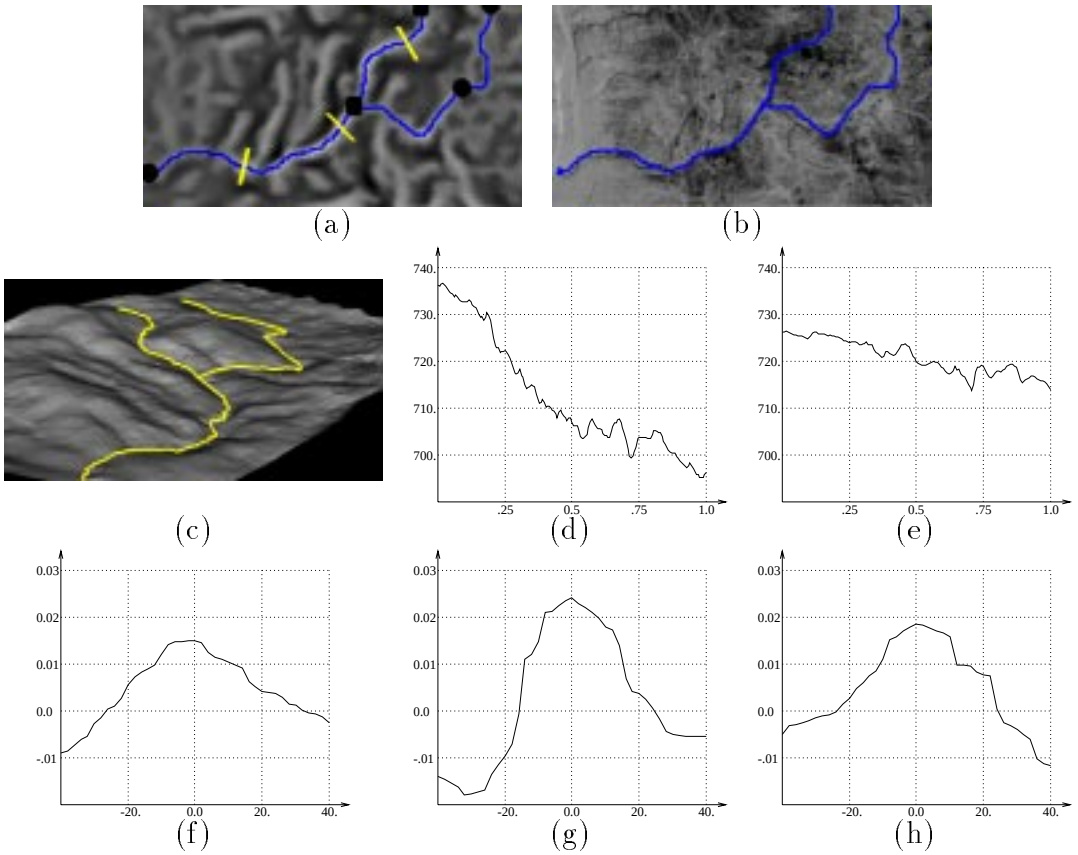


Figure 4: Sketching the rivers at the NTC site. (a) The maximal curvature paths overlaid on the curvature image of Figure 2(f). For one of the paths, we specified two endpoints and one intermediate point denoted by the black circles; for the other we specified only the two endpoints denoted by the black rectangles. (b) The paths overlaid on the original image. (c) The paths overlaid on a shaded view of the terrain mesh of Figure 2(e). (d,e) Elevations along the paths. Note that because of imprecisions in the reconstruction, they are not monotonic. (f,g,h) Curvature of the surface along the three perpendicular cross sections shown as white segments in (a). Note that the paths lie at local maxima of curvature.

We outline our approach to sketching the drainage pattern with a minimum of user intervention. We distinguish between steep terrain where the geometry of the terrain surface is usually sufficient to detect the drainage channels and flatter terrain where geometry becomes less relevant and the information present in the original images must be used more directly.

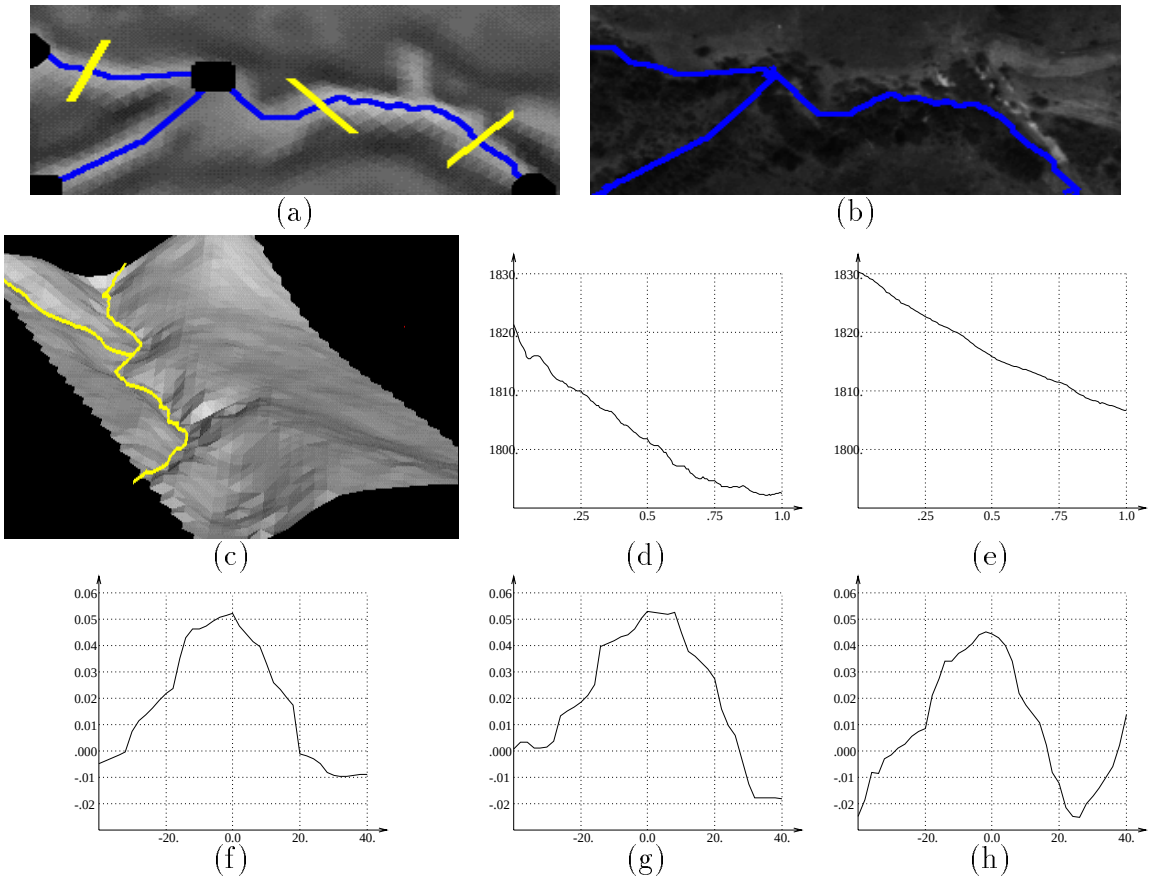
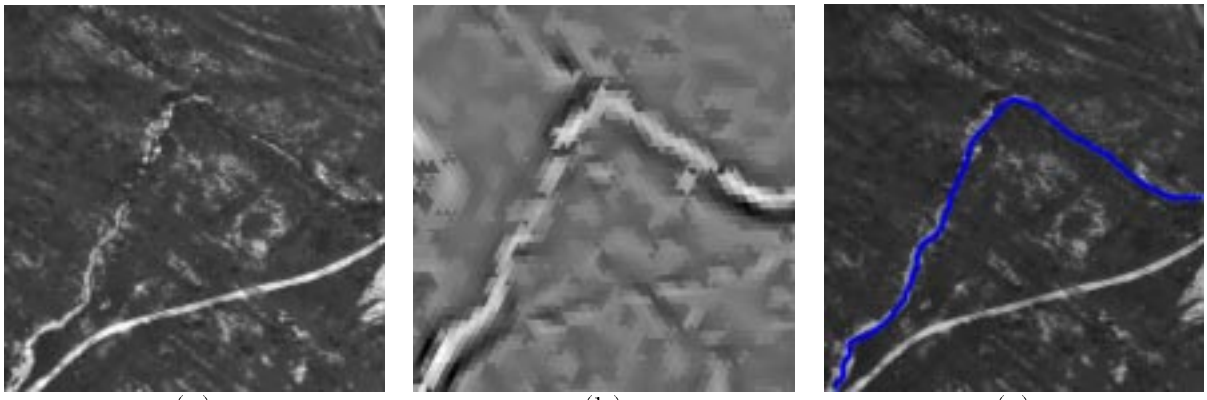


Figure 5: Sketching the rivers at the UGV site. (a) The maximal curvature paths overlaid on the curvature image of Figure 3(d). For both paths we specified only the endpoints, denoted by the black circles and black rectangles, respectively. (b) The paths overlaid on the original image. (c) The paths overlaid on a shaded view of the terrain mesh of Figure 3(c). (d,e) Nonmonotonic elevations along the paths. (f,g,h) Curvature of the surface along the three perpendicular cross sections shown as white segments in (a).

4.1 Steep Terrain

In high-relief areas, rivers create valleys by eroding the surrounding terrain and over time carve channels that typically are not completely filled with water. As a result they tend to appear as local depressions and their center lines closely match maxima of curvature in the terrain surface.

It is therefore natural to look for paths of maximum curvature in the "curvature images" introduced in Section 3.2 and computed using the terrain mesh. As shown in Figures 4, 5 and 6, this can be achieved by simply specifying endpoints and using a



(a) (b) (c)

Figure 6: River delineation at the McKenna, MOUT site, Ft. Benning. (a) One of a pair of stereo images with a meandering ditch. (b) The corresponding curvature image. (c) Maximal curvature path computed by specifying only the endpoints and overlaid on the original image.

dynamic programming algorithm [Fischler *et al.*, 1981] to find a path $\mathcal{C}$ that minimizes

$$\mathcal{E}_{Curv}(\mathcal{C}) = \int (C_{max}(\mathbf{f}(s)) - C^1_{max})^2 \, ds \quad , \tag{2}$$

where $\mathbf{f}(s)$ is a vector function mapping the arc length s to points (u, v) along the curve, $C_{max}(u, v)$ is the terrain's surface maximal curvature at image location (u, v), and C^1_{max} is the largest value of $C_{max}(u, v)$ in the curvature image. Using recent dynamic programming implementations [Mortensen and Barrett, 1995, Cohen and Kimmel, 1996], this can be done in near real time on a regular workstation, making this approach a very attractive way to sketch the drainage pattern.

In Appendix A we prove that if $\mathcal{C}$ minimizes $\mathcal{E}_{Curv}(\mathcal{C})$—that is, if it is a local minimum of $\mathcal{E}_{Curv}$ with respect to infinitesimal deformations of the curve—it verifies:

$$\frac{\partial C_{max}(s)}{\partial n} = 1/2\kappa(s)(C_{max}(s) - C^1_{max}) \ \ \forall s \quad , \tag{3}$$

where $\kappa(s)$ is the curvature of the path—as opposed to the curvature of the surface $C_{max}(s)$—and $\partial/\partial n$ denotes the derivative in the direction normal to the curve. It follows that, wherever $\kappa(s)$ is small,

$$\frac{\partial C_{max}}{\partial n} \ll (C_{max}(s) - C^1_{max}) \quad . \tag{4}$$

Therefore $\mathcal{C}$ is close to being the locus of points that are maxima of curvature in the direction normal to the curve. We therefore refer to these paths as "maximal curvature" paths.

In practice, because $\mathcal{C}$ is computed using dynamic programming, it is discretized and made of points with integer coordinate values. Therefore, Equation 4 does not hold strictly. But, as illustrated by Figures 4 and 5, we have verified experimentally that the points of $\mathcal{C}$ are within a pixel of the actual maxima of curvature of the terrain surface.

4.2 Flat Terrain

As the terrain's relief becomes less pronounced, the channels become increasingly difficult to detect from the geometry of the surface mesh alone. In the limit, a river meandering through an almost flat flood plain could not be sketched using the technique described above.

Figure 7 illustrates this problem in an area where the terrain's shape is close to that of a slanted plane. In such cases the clues to the river's presence are to be found in the original gray-level images where they appear as elongated linear structures that can be detected using a low-resolution linear delineation (LRLD) system.

Because it has demonstrated excellent performance, we use the LRLD system developed by Fischler and Wolf [Fischler and Wolf, 1983]. It has two major components. The first component, the detector/binarizer, accepts an image and is intended to return a binary mask that retains the linear structures of interest in a form clearly visible to a normal human observer. The second component, the generic linker, uses generic criteria—continuity, contiguity, coherence, length as the basis for extracting sequences of points that represent the perceptually obvious curved lines present in the binary mask yielding results such as those shown in Figure 8(a). These linear features can then be chamfered and used to mask the curvature image. This operation produces the image of Figure 8(b) in which the curvature of all points but those that are within a given number of pixels—10, for the examples shown in this paper—of the linear structures is set to C^0_{max}, the smallest value in the original curvature image, and will therefore tend to be avoided by the dynamic programming algorithm. Using the same endpoints as previously, we obtain the paths shown in Figure 8(c) that are much closer to those that a human analyst would delineate using a stereoscope as shown in Figure 18 of Appendix C.

In Figure 9, we use a second site at the National Training Center to further illustrate the importance of combining gray-level and 3–D information.

5 Enforcing the Physical Constraints

We now turn to the physical constraints that the drainage pattern and surrounding terrain must fulfill. As illustrated by Figures 4 and 5, there is no guarantee that the features sketched using the techniques of Section 4 will be consistent with the laws of physics—for example, they may not have monotonically decreasing elevations—mainly because the terrain model may be in error.

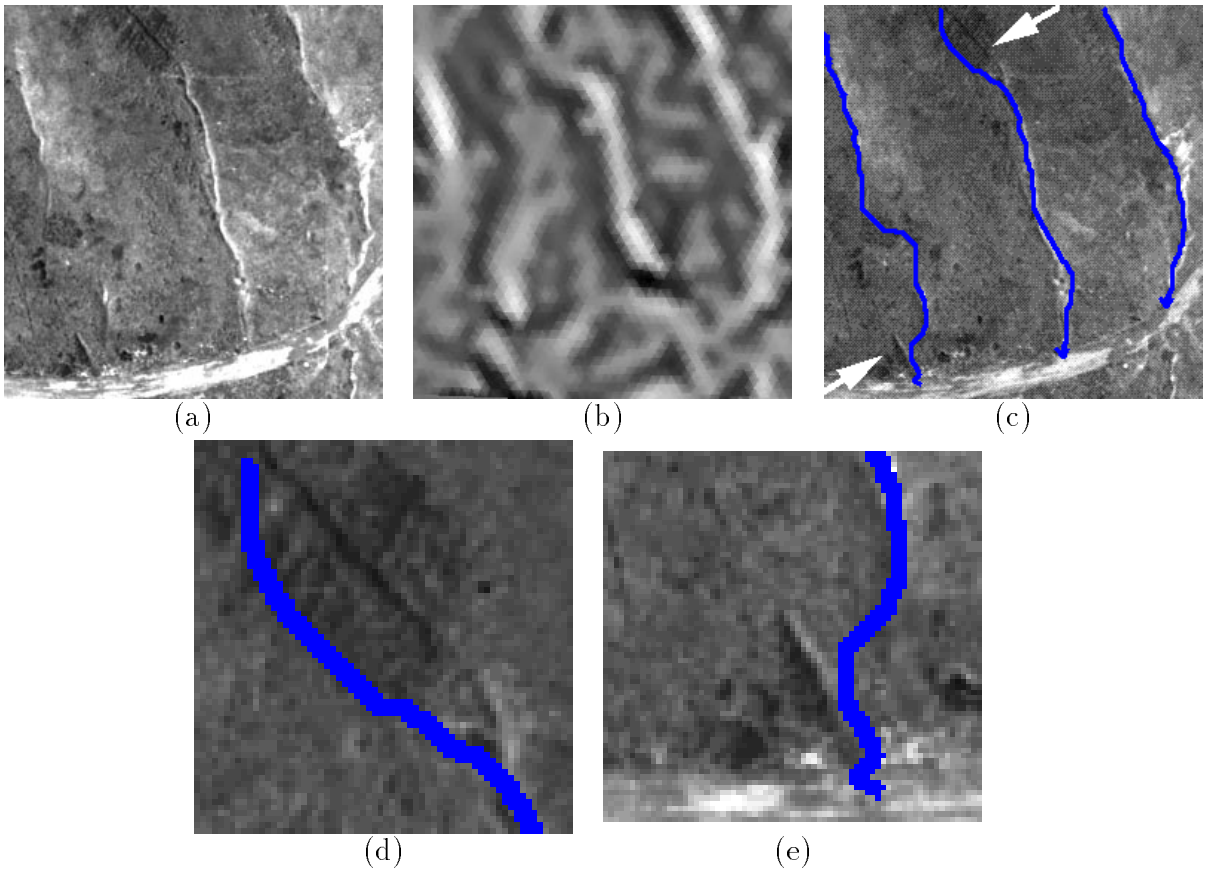
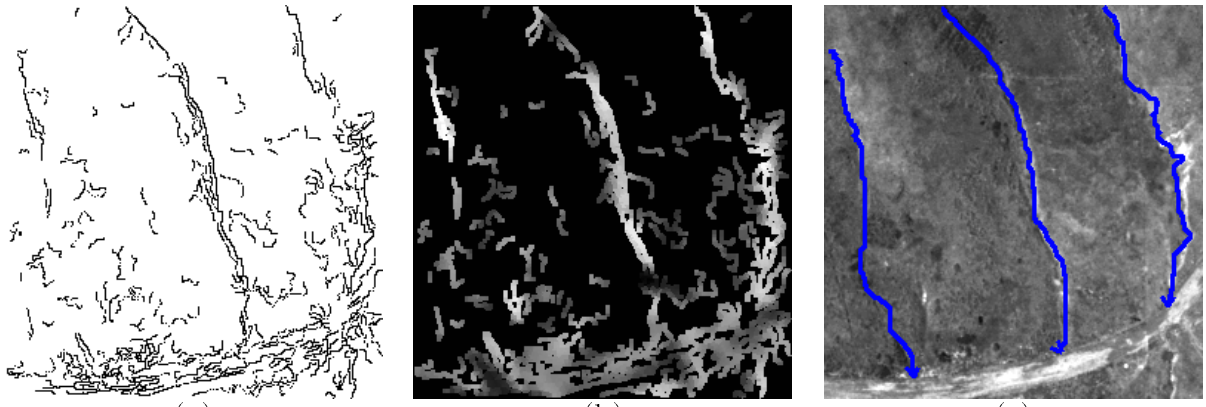


Figure 7: A flatter part of the Martin-Lockheed UGV site. (a) One of the three images of this area. The terrain is close to being a slanted plane with the highest elevations at the top of the image. Note the three gullies running from top to bottom of the image. (b) The corresponding curvature image. (c) The maximal curvature paths computed by specifying two endpoints for each gully. (d) Detail of the upper part of the middle path—denoted by the topmost white arrow in (c)—that meanders away from the clearly visible linear structure that marks the actual location of the gully. (e) Similar problem in the lower part of the leftmost path, denoted by the other white arrow in (c).

Our goal is therefore to enforce these constrains while deviating as little as possible from what the image data predicts; otherwise we might be "hallucinating" river valleys where there are none. Constrained optimization [Fletcher, 1987, Gill *et al.*, 1981, Metaxas and Terzopoulos, 1991] is an effective way to achieve this goal because it allows the use of arbitrarily large numbers of constraints while retaining good convergence properties. In fact, the more constraints there are, the smaller the search space and the better the



(a) (b) (c)

Figure 8: Combining gray-level and curvature information. (a) The linear features detected by the low-resolution linear delineation system in the image of Figure 7(a). (b) The potential image computed by using those linear features to mask the curvature image. (c) The linear structures delineated by using the potential image (b) and supplying two endpoints for each of the three linear structures.

convergence becomes. In previous work [Fua and Brechbühler, 1996], we have developed a constrained optimization algorithm that exploits the specificities of the models we use—surface meshes and polygonal curves—to reduce the required amount of computation and to allow us to impose the constraints at a very low computational cost. This method is described briefly in Appendix B and in more detail in our earlier publication.

Formally, a constrained optimization problem can be described as follows. Given a function f of n variables $S = \{s_1, s_2, .., s_n\}$, we want to minimize it under a set of m constraints $C(S) = \{c_1, c_2, .., c_m\} = 0$. That is,

$$\text{minimize } f(S) \text{ subject to } C(S) = 0 \quad . \tag{5}$$

It can be generalized to handle inequality constraints by replacing the constraints of the form $c_i(S) = 0$ by constraints of the form $c_i(S) \leq 0$.

In our application, we model the terrain as a triangulated mesh $\mathcal{S}$ and linear features as a set of l polygonal curves $\mathcal{C}_{j,1\leq j\leq l}$. We associate to each an energy term $\mathcal{E}(\mathcal{S})$ and $\mathcal{E}(\mathcal{C}_j)$. $\mathcal{E}(\mathcal{S})$ is discussed in Section 3 and $\mathcal{E}(\mathcal{C}_j)$ is introduced below. The state vector S is the vector of all the x,y and z coordinates of the vertices of $\mathcal{S}$ and of the $\mathcal{C}_j$s. $f(S)$ is taken to be

$$f(S) = \{\mathcal{E}(\mathcal{S}), \mathcal{E}(\mathcal{C}_1), ..., \mathcal{E}(\mathcal{C}_l)\} \quad ,$$

and our algorithm minimizes each component of f while attempting to satisfy the constraints. Note that in our approach, we do have not sum $\mathcal{E}(\mathcal{S})$ and the $\mathcal{E}(\mathcal{C}_j)$. It is therefore not necessary to normalize them or to compute relative weights.

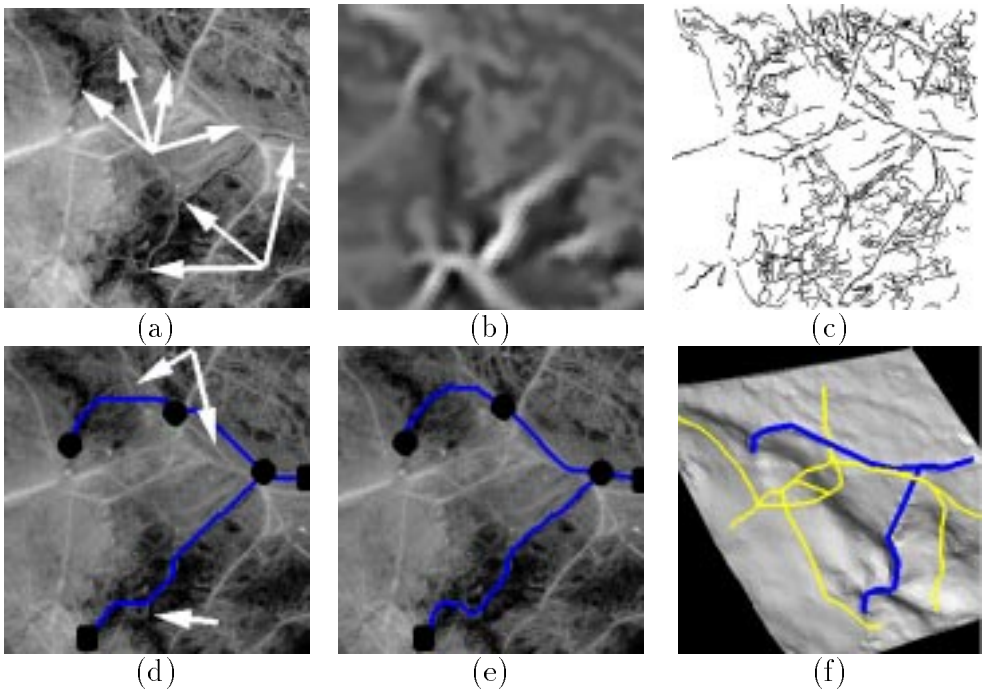


Figure 9: A different site at the National Training Center (NTC). (a) One of four aerial images with two streambeds denoted by the two clusters of white arrows. (b) The curvature image. Regions of high positive curvature are shown in white. Because the relief is not as pronounced as in the case of Figure 2, the valleys do not appear as clearly. (c) The linear features detected by the low-resolution linear delineation system. (d) The paths delineated by using only the curvature image shown in (b). For the path at the top, we specified two endpoints and one intermediate point denoted by the black circles. For the path at the bottom, we specified only the two endpoints denoted by the black rectangles. Because there is not enough relief, the paths wander away from their apparent location at the places indicated by the white arrows. (e) The paths delineated by specifying the same endpoints and using a potential image that combines the curvature image and the output of the linear delineation program shown in (c). (f) The paths, shown as dark lines, overlaid on a shaded view of the terrain mesh after constrained optimization using all four images. In this scene, there are also dirt roads that form light linear features in the gray-level images. They have been outlined using dynamic programming and appear as white lines.

We must now express the fact that rivers flow downhill and lie at the bottom of

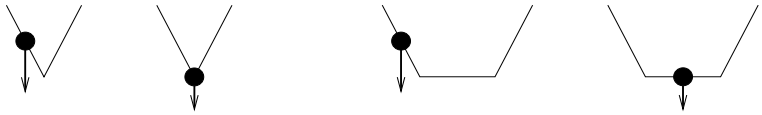


Figure 10: Minimizing elevation under constraint. (a) If the terrain was "V" shaped, the on-terrain constraints would force the z minimizing curves that we use to model drainage to settle into the position shown on the right side of the figure. (b) In fact the valley-bottom constraints force the valleys to be "U" shaped and the curve to settle on the horizontal par of the U, although not necessarily in the middle of it. The exact location is determined by the fact that the curve must minimize its own curvature.

local depressions in the terrain and that valley floors tend to be "U" shaped and locally horizontal in the direction transverse to the river's direction in terms of a set of constraints of the form $c_i(S) = 0$ or $c_i(S) \leq 0$:

- **Rivers lie at the bottom of valleys:** We treat a river as smooth 3–D curve $\mathcal{C}$. We refine its position by minimizing an energy $\mathcal{E}(\mathcal{C})$ that is the weighted sum of a regularization term $\mathcal{E}_D(\mathcal{C})$—the integral of the square curvatures along the curve—and a potential term $\mathcal{E}_P(\mathcal{C})$—minus the integral of the elevations along the curve. We write

$$\begin{aligned} \mathcal{E}_D(\mathcal{C}) &= \int \kappa(s)^2 \, ds \\ \mathcal{E}_P(\mathcal{C}) &= -\int_0^{|\mathcal{C}|} z(s) \, ds \quad , \end{aligned} \tag{6}$$

  where s is the arc length along the curve, $\kappa(s)$ is the curvature of the curve and $z(s)$ its elevation.

  Following standard snake practices [Kass *et al.*, 1988], we model $\mathcal{C}$ as a list of regularly spaced 3–D vertices $\mathcal{S}_3$ of the form

$$\mathcal{S}_3 = \{(x_i \; y_i \; z_i), \; i = 1, \ldots, n\} \quad , \tag{7}$$

  and we write

$$\begin{aligned} \mathcal{E}_D(\mathcal{C}) &= \frac{1}{2} \sum (2x_i - x_{i-1} - x_{i+1})^2 + (2y_i - y_{i-1} - y_{i+1})^2 + (2z_i - z_{i-1} - z_{i+1})^2 \\ \mathcal{E}_P(\mathcal{C}) &= \sum z_i \quad . \end{aligned} \tag{8}$$

  As discussed below, during the optimization the curve $\mathcal{C}$ is constrained to remain on the terrain while the vertices are moved to minimize $\mathcal{E}_P(\mathcal{C})$ and the elevations

of the individual vertices. As a result, and as shown in Figure 10, at the end of the optimization, the curve has to lie at the bottom of a valley.

- **Rivers flow downhill:** The z coordinates of the curve's list of n 3–D vertices $\mathcal{S}_3$ decrease monotonically, which is expressed as a set of $n-1$ inequality constraints

$$z_{i+1} \leq z_i \quad , \tag{9}$$

that we refer to as "downhill" constraints.

- **Rivers lie on the terrain:** For each edge $((x_1, y_1, z_1), (x_2, y_2, z_2))$ of the terrain mesh and each segment $((x_3, y_3, z_3), (x_4, y_4, z_4))$ of the polygonal curve representing the river that intersect when projected in the (x, y) plane, the four endpoints must be coplanar so that the segments also intersect in 3–D space. This is written as

$$\begin{vmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{vmatrix} = 0 \quad , \tag{10}$$

which yields a set of constraints that we refer to as "on-terrain" constraints.

- **The valley is horizontal in the direction transverse to the river's direction:** Each edge $((x_1, y_1, z_1), (x_2, y_2, z_2))$ of the terrain mesh that intersects, in the (x, y) plane, a segment $((x_3, y_3, z_3), (x_4, y_4, z_4))$ of the polygonal curve representing the river must have the following property: the component of the vector

$$\overrightarrow{e_{12}} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix}$$

that is perpendicular to the vector

$$\overrightarrow{e_{34}} = \begin{pmatrix} x_4 - x_3 \\ y_4 - y_3 \\ z_4 - z_3 \end{pmatrix}$$

must be horizontal. This can be written as

$$\begin{vmatrix} x_2 - x_1 & x_4 - x_3 & y_3 - y_4 \\ y_2 - y_1 & y_4 - y_3 & x_4 - x_3 \\ z_2 - z_1 & z_4 - z_3 & 0 \end{vmatrix} = 0 \quad , \tag{11}$$

because it implies that $\overrightarrow{e_{12}}$ is a linear combination of $\overrightarrow{e_{34}}$ and the vector

$$\begin{pmatrix} y_3 - y_4 \\ x_4 - x_3 \\ 0 \end{pmatrix}$$

which is both horizontal and perpendicular to $\overrightarrow{e_{34}}$. This yields another set of constraints that we refer to as "valley-bottom" constraints.

In practice, the downhill constraints are inequality constraints that are turned on and off during the optimization as required, "following an active set strategy" as explained in Appendix B. Before the start of the constrained optimization, we compute the intersections in the x, y plane between the edges of the mesh and the polygonal curves to instantiate the required number of on-terrain and valley-bottom constraints. Optionally, we could reiterate this procedure during the optimization. This would be necessary if the polygonal curves deformed a lot. However, because the delineation method of Section 4 is robust, the initial location of the curves is within a few pixels of their true locations so that, in practice, they do not deform very much.

Figures 11, 12, 13 and 14 demonstrate the improvement in consistency brought about by constrained optimization: The channels now have monotonically decreasing elevations and the rivers lie at their bottoms, which also is close to being a maximum of curvature in the direction normal to the feature. In Figure 12, we compare the elevations computed before and after constrained optimization and show that imposing the constraints improves the definition of the valley. In Figure 14 we show similar results for the images of Figure 7.

Having refined the terrain model using the technique of Section 3 and enforced physical constraints as described in Section 5, we are now confronted with the perennial Computer Vision question: How close are we to ground truth? In our specific case, we also want to know if we have paid any price—in terms of accuracy, for instance—to enforce consistency. For example, how often have we mistakenly constrained the slope of the terrain to be in a certain direction to satisfy the constraints?

In Appendix C, we use two methods to generate "ground truth." The first is to use a good algorithm and to consider its results only in areas where it is known to be particularly reliable and accurate. The second method is to do it manually to the best of the ability of a human operator. By comparing the outputs of both these methods on the scenes of Figures 2 and 7 against the results produced by our system, we show that our approach to surface modeling yields excellent accuracy whether or not we impose our physical constraints—in the order of 0.2 to 0.3 RMS error in disparity—and that imposing the constraints does not detract from the accuracy.

To highlight the generality of the approach, we show that it can also be used to delineate ridgelines and roads. The dirt roads of Figure 9 appear as distinct white lines that can easily be delineated using dynamic programming in the gray-level images alone. Ridgelines such as those of Figure 15, are characterized by extremal *negative* curvatures and appear as dark lines in the gray-level images. They can therefore also be delineated using the technique of Section 4. We have refined both roads and ridgelines by treating them as 3–D snakes [Fua, 1996] that are attracted by white or dark lines and used on-terrain constraints to force them to remain on the ground. In a more sophisticated system, we could further improve the road reconstruction by introducing constraints that reflect known construction practices such as the fact that roads do not have arbitrary slopes or curvatures.

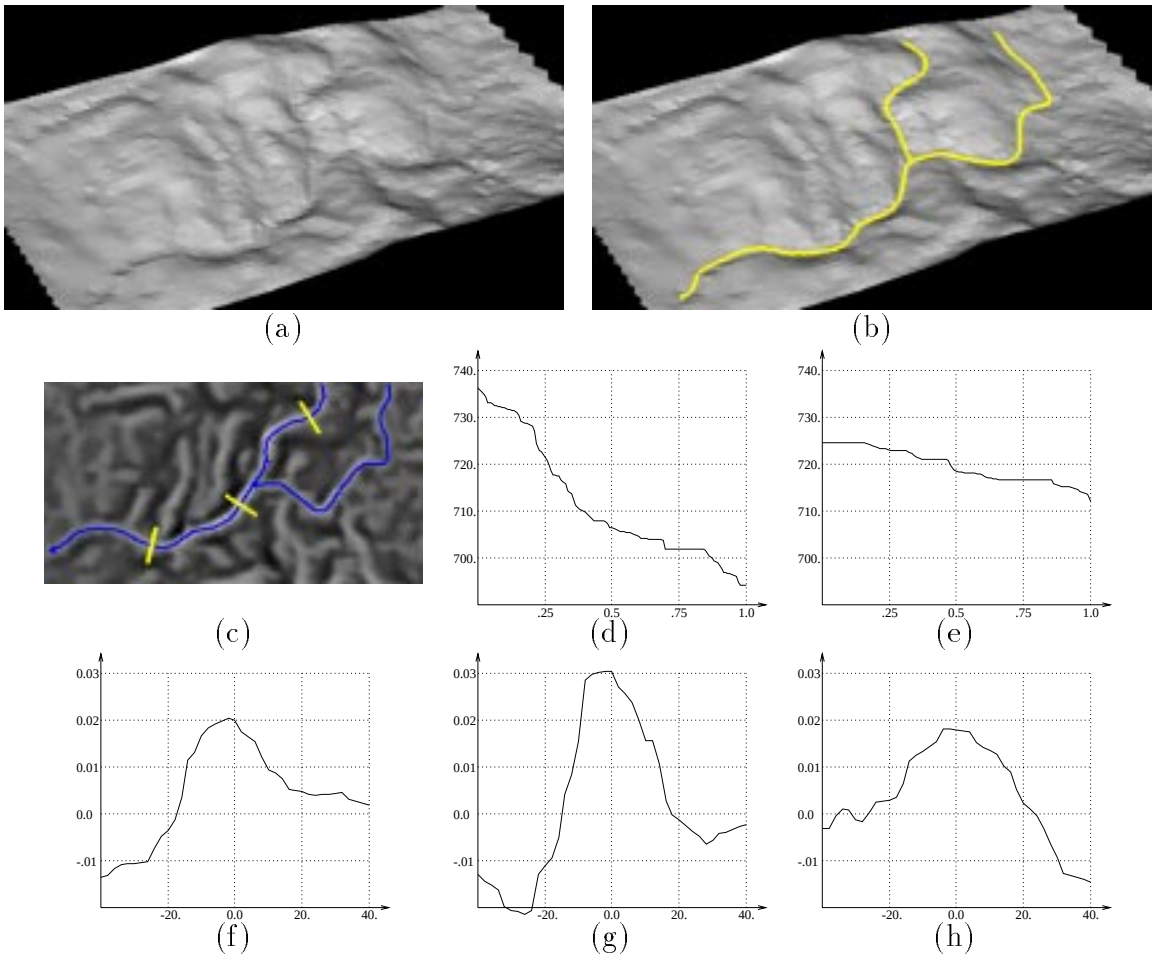
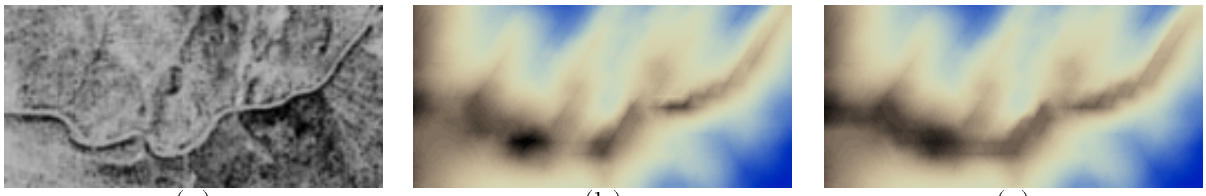


Figure 11: NTC model after constrained optimization. (a) A shaded model of the refined and constrained surface mesh of Figure 4 (c). Note the flat-bottomed valleys. (b) The two optimized streambeds overlaid on the mesh. (c) The optimized streambeds overlaid on a recomputed curvature image. (d,e) Their elevations are now monotonically decreasing, unlike those of Figures 4. (f,g,h) Curvature of the surface along the three perpendicular cross sections shown as white segments in (c).

6 Conclusion

We have presented an approach to terrain modeling and 3–D linear delineation that allows us to generate site models including terrain, drainage channels, roads and ridge lines that are accurate and consistent with minimal human intervention.

We have shown that, by refining an object-centered representation of the terrain and



(a) (b) (c)

Figure 12: Comparing the elevations before and after reoptimization under constraints. (a) A window of the image of Figure 2(a) centered on a section of the horizontal valley. (b) A pseudo-color depiction of the corresponding elevations computed using the terrain mesh of Figure 2(e), that is, before constrained optimization. The lowest elevations appear in black. (c) A pseudo-color depiction of the elevations using the terrain mesh of Figure 11(a), that is, after reoptimization under constraints. Note that the valley is now much better defined.

features under a set of well-designed constraints, we can generate, with a high level of automation, models that are faithful to sensor data, internally consistent and consistent with physical constraints. We have also shown that we can achieve this result with little human intervention: the operator is only required to specify a few endpoints, and the system handles everything else.

We have concentrated on the modeling of drainage patterns but the framework described here extends naturally to modeling all objects obeying known physical constraints. For example, man-made objects such as roads, railroad tracks, or buildings are built according to well-understood engineering practices. Similarly, silhouette edges can be extracted from ground-level views of mountain ridges and used to constrain the terrain modeling from aerial views.

We believe that the capabilities described here will prove indispensable to automating the generation of complex object databases from imagery, such as the ones required for realistic simulations or intelligence analysis. In such databases, the models must not only be as accurate—that is, true to the data—as possible but also consistent with each other. Otherwise, the simulation will exhibit "glitches" and the image analyst will have difficulty interpreting the models.

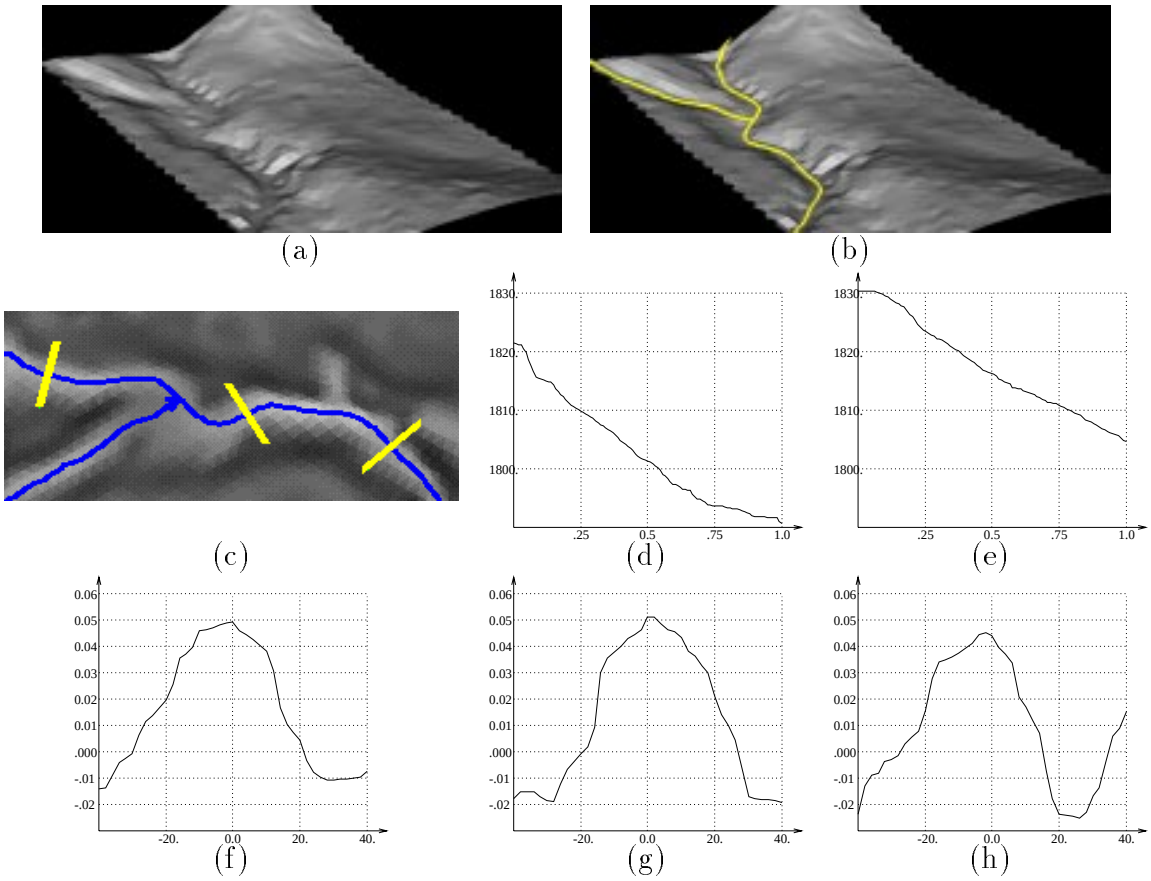
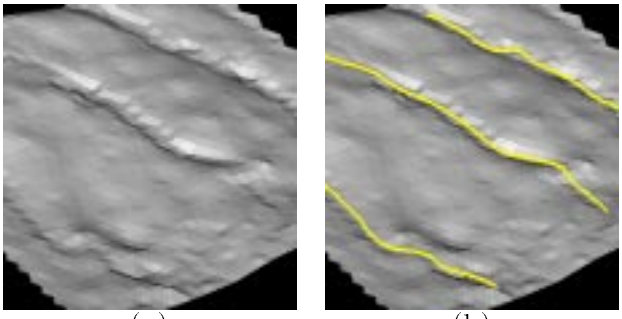
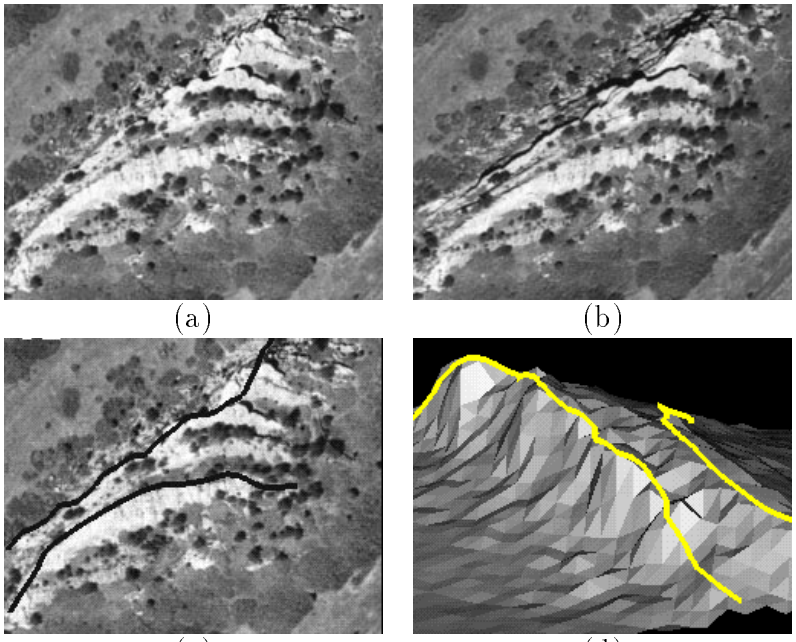


Figure 13: First UGV model after constrained optimization. (a) A shaded model of the refined and constrained surface mesh of Figure 5 (c). (b) The two optimized streambeds overlaid on the mesh. (c) The optimized streambeds overlaid on a recomputed curvature image. (d,e) Their elevations are monotonically decreasing, unlike those of Figure 5(d,e). (f,g,h) Curvature of the surface along the three perpendicular cross sections shown as white segments in (c).



(a) (b)

Figure 14: Second UGV model after constrained optimization. (a) A shaded model of the refined and constrained surface mesh of Figure 7 (c). (b) The three optimized streambeds overlaid on the mesh.



(a) (b)

(c) (d)

Figure 15: Modeling ridges at the UGV site. (a,b) Two images of a rocky outcrop. (c) The ridgeline recovered by the dynamic programming algorithm. (d) A shaded view of the terrain mesh recovered by our algorithm.

Appendices

A Maximizing Curvature

In this appendix, to prove Equation 3, we summarize a proof that has appeared in one of our earlier publications [Fua and Leclerc, 1990]. We show that if an open curve $\mathcal{C}$ is a local extremum of

$$\mathcal{E}(\mathcal{C}) = \int_0^{|\mathcal{C}|} G(\mathbf{f}(s))ds \tag{A.1}$$

with respect to all infinitesimal deformations, then

$$\frac{\partial G(\mathbf{f}(s))}{\partial n} = \kappa(s)G(\mathbf{f}(s)) \quad , \tag{A.2}$$

where the curve $\mathcal{C}$ is parameterized by its arc length s and of length $|\mathcal{C}|$, G is a C^2 function, $\mathbf{f}(s)$ is a vector function mapping the arc length s to points (x, y) along the curve, $\kappa(s)$ is the curve's curvature and and $\partial/\partial n$ denotes the derivative in the direction normal to the curve. Replacing $G(\mathbf{f}(s))$ by $(C_{max}(\mathbf{f}(s)) - C^1_{max})^2$ in Equation A.2 yields Equation 3.

To prove this result, consider deformations of the curve $\mathcal{C}$, which we shall call $\mathcal{C}_\lambda$, such that the mapping from arc length s to points (x, y) is of the form

$$\mathbf{f}_\lambda(s) = \mathbf{f}(s) + \lambda\left(\eta(s)\mathbf{n}(s) + \tau(s)\mathbf{t}(s)\right), \tag{A.3}$$

where $\mathbf{n}(s)$ is the normal to the curve, $\mathbf{t}(s)$ is the tangent, and $\eta(s)$ and $\tau(s)$ are arbitrary continuous and differentiable functions such that $\tau(0) = \tau(|\mathcal{C}|) = \eta(0) = \eta(|\mathcal{C}|) = 0$.

Let

$$\mathcal{E}(\mathcal{C}_\lambda) = \int G(\mathbf{f}_\lambda(s_\lambda))ds_\lambda \quad , \tag{A.4}$$

where s_λ is the arc length of $\mathcal{C}_\lambda$.

If $\mathcal{C}$ is a local extremum of $\mathcal{E}(\mathcal{C})$, then

$$\left.\frac{d\mathcal{E}(\mathcal{C}_\lambda)}{d\lambda}\right|_{\lambda=0} = 0 \quad , \tag{A.5}$$

for all $\eta(s)$ and $\tau(s)$ such that $\tau(0) = \tau(|\mathcal{C}|) = \eta(0) = \eta(|\mathcal{C}|) = 0$.

Using integration by part, it can be shown that for all such $\eta(s)$ and $\tau(s)$,

$$\left.\frac{d\mathcal{E}(\mathcal{C}_\lambda)}{d\lambda}\right|_{\lambda=0} = \int \left[\frac{\partial G(\mathbf{f}(s))}{\partial n} - \kappa(s)G(\mathbf{f}(s))\right]\eta(s)ds \quad . \tag{A.6}$$

The desired result follows immediately.

B Constrained Optimization

Formally, a constrained optimization problem can be described as follows. Given a function f of n variables $S = \{s_1, s_2, .., s_n\}$, we want to minimize it under a set of m constraints $C(S) = \{c_1, c_2, .., c_m\} = 0$. That is,

$$\text{minimize } f(S) \text{ subject to } C(S) = 0 \ . \tag{B.1}$$

In our application, there are always many more variables than constraints, that is, $n \gg m$.

While there are many powerful methods for nonlinear constrained minimization [Gill *et al.*, 1981], with the exception of the approach proposed by Metaxas and Terzopoulos [1991], few are designed for snake-like optimization: They do not take advantage of the locality of interactions that is characteristic of snakes. We have therefore developed a robust two-step approach [Brechbühler *et al.*, 1995, Fua and Brechbühler, 1996] that is closely related to gradient projection methods first proposed by Rosen [1961] and has been extended to snake optimization.

Solving a constrained optimization problem involves satisfying the constraints and minimizing the objective function. For our application, it has proved effective to decouple the two and decompose each iteration into two steps:

1. Enforce the constraints by projecting the current state onto the constraint surface. This involves solving a system of nonlinear equations by linearizing them and taking Newton steps.

2. Minimize the objective function by projecting the gradient of the objective function onto the subspace tangent to the constraint surface and searching in the direction of the projection, so that the resulting state does not stray too far away from the constraint surface.

Figure 16 depicts this procedure. Let C and S be the constraint and state vectors of Equation B.1 and A be the $n \times m$ Jacobian matrix of the constraints. The two steps are implemented as follows:

1. To project S, we compute dS such that $C(S+dS) \approx C(S)+A^t dS = 0$ and increment S by dS. The shortest possible dS is found by solving the underconstrained system $A^t dS = -C(S)$ in the least-squares sense.

2. To compute the optimization direction, we first solve the overconstrained linear system $A(S)\lambda = \nabla f$ in the least-squares sense and take the direction to be $\nabla f - A\lambda$. This amounts to estimating Lagrange multipliers, that is, the coefficients that can be used to describe ∇f as closely as possible, as a linear combination of constraint normals.

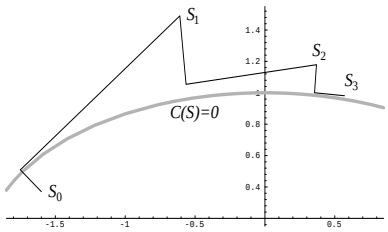


Figure 16: Constrained optimization. Minimizing $(x-0.5)^2+(y-0.2)^2$ under the constraint that $(x/2)^2+y^2=1$. The set of all states that satisfy the constraint $C(S)=0$, i.e. the constraint surface, is shown as a thick gray line. Each iteration consists of two steps: orthogonal projection onto the constraint surface followed by a line search in a direction tangent to the surface. Because we perform only one Newton step at each iteration, the constraint is fully enforced after only a few iterations.

These two steps operate in two locally orthogonal subspaces, in the column space of A and in its orthogonal complement, the null space of A^T. Because the interactions are very local, the matrix A is always very sparse and both the underconstrained and overconstrained least-squares problems can be solved reliably at a low computational cost even when A is ill-conditioned—the constraints are not truly independent—using a sparse least-squares solver such as LSQR [Paige and Saunders, 1982].

This technique can be generalized to handle inequality constraints by introducing an "active set strategy." The inequality constraints that are strictly satisfied are deactivated, while those that are violated are activated and treated as equality constraints. This requires additional bookkeeping but does not appear to noticeably slow down the convergence of our constrained-optimization algorithm.

C Quantitative Evaluation of the Results

Here, we use the images of Figures 2 and 7 to gauge the accuracy of our approach. We believe that these two examples are complementary because, in the first case, we performed our initial delineation by using only the curvature information while, in the second case, we took advantage of the low-resolution linear delineation system .

C.1 Comparison against the Output of Another Algorithm

One approach to generating "ground truth" is to use a good algorithm and to consider its results only in areas where it is known to be particularly reliable and accurate. The

STEREOSYS correlation-based stereo system [Hannah, 1988] has won an international stereo competition [Gülch, 1988]. It has been shown to yield extremely reliable results provided that one only retains the points to which it gives very high confidence scores such as the ones shown in Figure 17. The root mean square (RMS) error for those matches can usually be expected to be in the range of 0.2 to 0.4 pixels, which we will compare to the RMS errors of our own system.

Given the most reliable STEREOSYS matches, we use the camera models associated with the image pairs to compute the x,y and z coordinates of the points and we compare the results to the elevation of the terrain meshes at the same x, y location. In the first two colum of Figure 17, we histogram these differences in elevation, expressed in meters, before—2nd row—and after—3rd row—the constrained optimization of Section 5. The error distributions have barely changed, making these results statistically indistinguishable as evidenced by the RMS values of those histograms shown in the two rightmost columns of Table 1. For comparison's sake, Table 1 also contains the RMS values of the differences in elevation with the initial coarse DEM before any refinement of the meshes as described in Section 3. They are, of course, considerably higher.

The similarity of the error distributions shown in the two leftmost columns of Figure 17 is not necessarily meaningful, as one would expect most of the variations between results to be found in the immediate vicinity of the drainage patterns. To further check this, we have manually selected the subset of STEREOSYS-generated points shown at the top of the rightmost column of Figure 17 that lie in the immediate vicinity of the river. We have then performed our comparison again, using only this subset. Again, the results are statistically indistinguishable and, if anything, the reoptimized mesh appears to yield a slightly smaller RMS value than the non-reoptimized one.

Another important thing to note is that at the image resolution we are using, a 1-meter change in elevation translates to a shift of approximately 0.8 pixels for the scene of Figure 3 and 0.3 pixel for the scene of Figure 9. If we used those shift values to express the RMS values of Table 1 in terms of pixels, we would obtain numbers in the range of 0.2 to 0.3 pixel. This is close to being the uncertainty that can be expected of STEREOSYS and, as a consequence, all these results can be seen as equally good from a strict accuracy point of view.

C.2 Comparing against Hand-entered Features

Another way to generate "ground truth" is to do it manually to the best of the ability of a human operator. In Figure 18, we show the manually entered features using RCDE, the RADIUS Common Development Environment [Mundy *et al.*, 1992], as though it were a stereoscope. Here, we compare the difference in elevation between the ground truth linear features and the terrain mesh before and after constrained optimization; again, the differences appear to be statistically insignificant, as evidenced by Table 2. Note, however, that these values are higher than those of the first row of Table 1. This can be attributed to the fact that the resolution of the meshes we have used is still too coarse

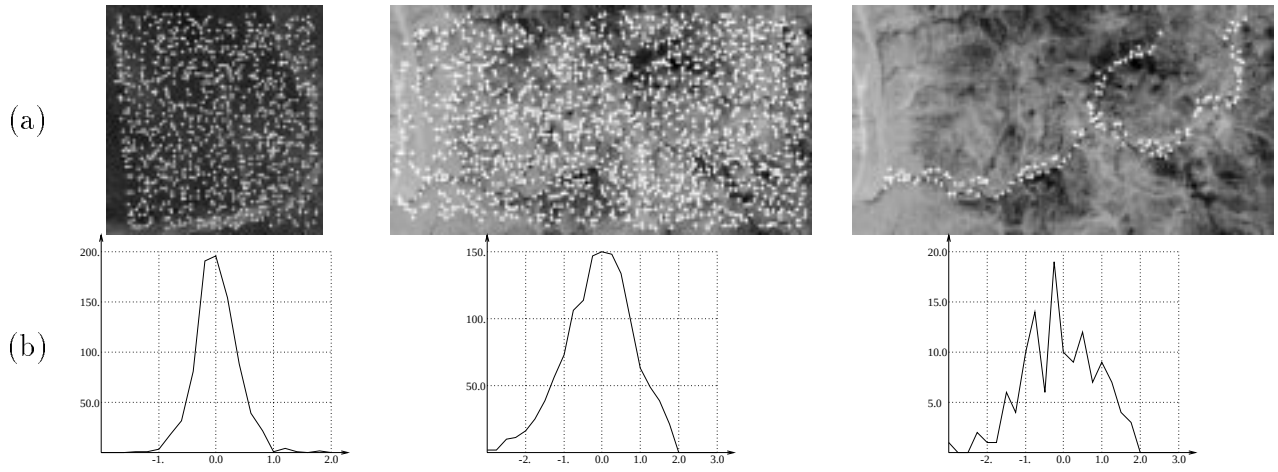


Figure 17: Comparing against the output of the STEREOSYS system. Row (a): Points matched with high confidence in the images of Figures 7 and Figure 2. In the rightmost image, we show only the subset of the latter points that lie along the streambed. Row (b): Histogram of the vertical distances of the points depicted by Row (a) to the mesh before constrained optimization. Row (c): Histogram of the vertical distances of the points depicted by Row (a) to the mesh after constrained optimization. The distances are expressed in meters. At the image resolution we use, a difference of 1 meter in elevation roughly corresponds to a shift of 0.8 pixel in the images of Figure 7 and 0.3 pixel in the images of Figure 2. Note that the histograms of rows (b) and (c) are almost indistinguishable.

to very precisely model the bottom of the gullies. The precision could be increased by further subdividing the whole mesh and reoptimizing. An even better solution would be to replace our regular meshes by irregular ones and our finite-difference implementation by a finite-element one [McInerney and Terzopoulos, 1993, Koh *et al.*, 1994], so that we could refine the triangulation *only* in the immediate vicinity of the streambeds.

It is worth noting, however, that hand-entering the three linear features very precisely while ensuring that their elevations decreased monotonically took a great deal more effort and attention—they respectively have 12, 11 and 11 vertices that must be painstakingly positioned—than specifying the pairs of endpoints of Figure 8 and allowing the system to do the rest.

C.3 Accuracy versus Consistency Tradeoff

These results suggest that

Data/Matches	All in Fig. 7	All in Fig. 2	Subset in Fig. 2
Number of points	834	1352	137
RMS of initial DEM	1.501 (1.20)	13.34 (4.00)	13.40 (4.02)
RMS of refined mesh	0.364 (0.54)	1.080 (0.32)	1.333 (0.40)
RMS of constrained mesh	0.400 (0.60)	1.045 (0.31)	1.197 (0.36)

Table 1: Root mean square values of the differences in elevation between the STEREOSYS matches and our surface meshes initially, after the unconstrained optimization of Section 3.1 and, finally, after constrained optimization. As in Figure 17, the first column corresponds to the matches in the images of Figure 7, the second one to the matches in the images of Figure 2, and the third column to the subset of those points that lie along the streambed. The errors are computed in meters. The numbers in parentheses are the same errors expressed in terms of image pixels at the resolution we use; note that these numbers are in the same range as the RMS errors of STEREOSYS.

Data/Streambed	left	middle	right
Number of vertices	12	11	11
RMS of refined mesh	0.606 (0.48)	0.800 (0.64)	1.340 (1.07)
RMS of constrained mesh	0.620 (0.49)	0.842 (0.67)	1.181 (0.94)

Table 2: Root mean square values of the differences in elevation between the vertices of the hand-entered features of Figure 18 and our surface meshes first after unconstrained optimization and, then, after constrained optimization. As in Figure 18, the first column corresponds to the leftmost streambed, the second column to the central one, and the last column to the rightmost streambed. The RMS errors are shown both in meters and, within parentheses, in pixels.

- The mesh approach to surface reconstruction of Section 3 yields excellent accuracy whether or not we impose our physical constraints. This accuracy is comparable to that of one of the best stereo-correlation algorithms available with the added advantages that
  - we produce a dense model,
  - we can simultaneously deal with an arbitrary number of images,
  - we produce an object-centered representation that can handle geometric properties such as occlusions and can be made to interact naturally with other

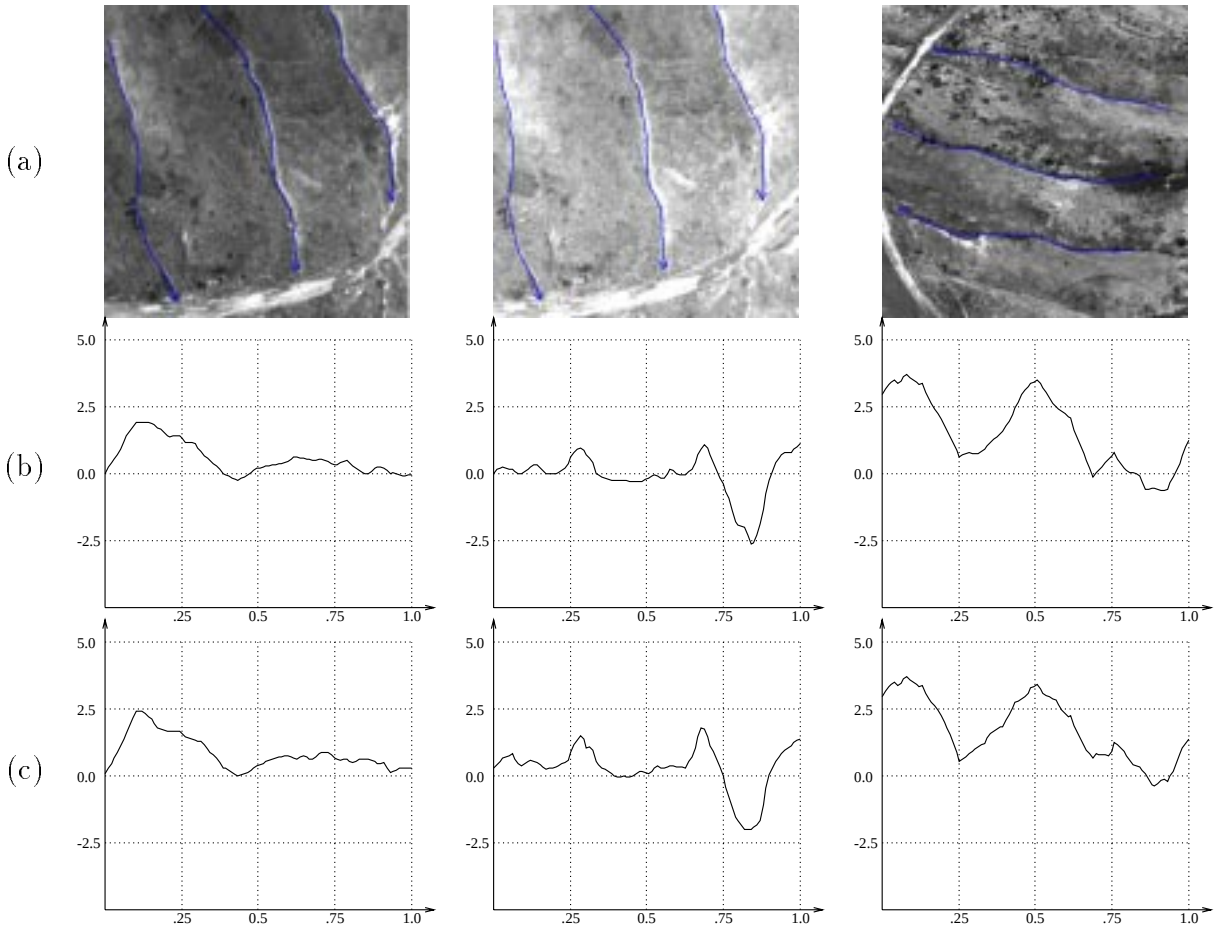


Figure 18: Comparison against hand-entered features. Row (a): Manually outlined drainage pattern using RCDE as a stereoscope. Row (b): Differences in elevation between the linear features and the terrain mesh before constrained optimization, plotted as a function of arc length along the path. Row (c): Differences in elevation between the linear features and the terrain mesh after constrained optimization.

features.

- Enforcing consistency as proposed in Section 5 does not detract from the accuracy, and therefore yields a model that is more likely to be useful for applications such as simulation and augmented reality where both faithfulness and consistency are required.

References

[Band, 1986] L.E. Band. Topographic Partition of Watersheds with Digital Elevation Models. *Water Resources Research*, 22:15–24, 1986.

[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.

[Cohen and Kimmel, 1996] L. Cohen and R. Kimmel. Global Minimum for Active Contour Models: A Minimal Path Approach. In *Conference on Computer Vision and Pattern Recognition*, pages 666–673, San Franciso, CA, June 1996.

[Fairfield and Leymarie, 1991] J. Fairfield and P. Leymarie. Drainage Networks from Grid Digital Evaluation Models. *Water Resources Research*, 27(5):709–717, May 1991.

[Fischler and Wolf, 1983] M.A. Fischler and H.C. Wolf. Linear Delineation. In *Conference on Computer Vision and Pattern Recognition*, pages 351–356, June 1983.

[Fischler *et al.*, 1981] M.A Fischler, J.M. Tenenbaum, and H.C. Wolf. Detection of Roads and Linear Structures in Low-resolution Aerial Imagery Using a Multisource Knowledge Integration Technique. *Computer Vision, Graphics, and Image Processing*, 15(3):201–223, March 1981.

[Fletcher, 1987] R. Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, Chichester, New York, Brisbane, Toronto, Singapore, 2nd edition, 1987. A Wiley-Interscience Publication.

[Fua and Brechbühler, 1996] P. Fua and C. Brechbühler. Imposing Hard Constraints on Soft Snakes. In *European Conference on Computer Vision*, pages 495–506, Cambridge, England, April 1996. Available as Tech Note 553, Artificial Intelligence Center, SRI International.

[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, 1995] P. Fua and Y. G. Leclerc. Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading. *International Journal of Computer Vision*, 16:35–56, September 1995.

[Fua and Leclerc, 1996] 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*, 64(1):111–127, July 1996. Also available as Tech Note 536, Artificial Intelligence Center, SRI International.

[Fua, 1996] P. Fua. Model-Based Optimization: Accurate and Consistent Site Modeling. In *XVIII ISPRS Congress*, Vienna, Austria, July 1996.

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

[Gilluly *et al.*, 1968] J. Gilluly, A.C. Waters, and A.O Woodford. *Principles of Geology*. W.H. Freeman and Company, San Francisco, CA, 1968.

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

[Hannah, 1988] M.J. Hannah. Digital Stereo Image Matching Techniques. *International Archives of Photogrammetry and Remote Sensing*, 27(III):280–293, 1988.

[Heipke, 1992] C. Heipke. Integration of Digital Image Matching and Multi Image Shape From Shading. In *International Society for Photogrammetry and Remote Sensing*, pages 832–841, Washington, D.C., 1992.

[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.

[Koenderink and van Doorn, 1991] J.J. Koenderink and J. van Doorn. Two-plus-One Dimensional Differential Geometry. *Pattern Recognition Letters*, 15(5):439–443, 1991.

[Koenderink and van Doorn, 1993] J.J. Koenderink and J. van Doorn. Local Features of Smooth Shapes: Ridges and Courses. In *SPIE*, volume 2031, 1993.

[Koh *et al.*, 1994] E. Koh, D. Metaxas, and N. Badler. Hierarchical Shape Representation Using Locally Adaptative Finite Elements. In *European Conference on Computer Vision*, Stockholm, Sweden, May 1994.

[Lengagne *et al.*, 1996] R. Lengagne, P. Fua, and O. Monga. Using Crest Line to Guide Surface Reconstruction from Stereo. In *International Conference on Image Processing*, Lausanne, Switzerland, September 1996.

[McInerney and Terzopoulos, 1993] T. McInerney and D. Terzopoulos. A Finite Element Model for 3D Shape Reconstruction and Nonrigid Motion Tracking. In *International Conference on Computer Vision*, pages 518–523, Berlin, Germany, 1993.

[Merlet and Zerubia, 1995] N. Merlet and J. Zerubia. New Prospects in Line Detection by Dynamic Programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4), April 1995.

[Metaxas and Terzopoulos, 1991] D. Metaxas and D. Terzopoulos. Shape and Nonrigid Motion Estimation through Physics-Based Synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):580–591, 1991.

[Mortensen and Barrett, 1995] E.N. Mortensen and W.A. Barrett. Intelligent Scissors for Image Composition. In *Computer Graphics, SIGGRAPH Proceedings*, pages 191–198, Los Angeles, CA, August 1995.

[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, San Diego,CA, 1992. Morgan Kaufmann.

[O'Callaghan and Mark, 1984] J. O'Callaghan and D.M. Mark. The Extraction of Networks from Digital Elevation Data. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 28:323–344, 1984.

[Paige and Saunders, 1982] C.C. Paige and M.A. Saunders. LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares. *ACM Transactions on Mathematical Software*, 8(1), March 1982.

[Rosen, 1961] Rosen. Gradient Projection Method for Nonlinear Programming. *SIAM Journal of Applied Mathematics*, 8:181–217, 1961.

[Sander and Zucker, 1990] P.T. Sander and S.W. Zucker. Inferring Surface Trace and Differential Structure from 3-D Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):833–854, September 1990.

[Taubin, 1995] G. Taubin. Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation. In *International Conference on Computer Vision*, pages 902–907, Cambridge, MA, June 1995.

[Wrobel, 1991] B.P. Wrobel. The evolution of Digital Photogrammetry from Analytical Photogrammetry. *Photogrammetric Record*, 13(77):765–776, April 1991.