Automating Landscape Illustration with Pen and Ink Style Rendering

James E. Mower

ABSTRACT: This paper introduces object space procedures for extracting silhouettes, slope lines, and drainage features from digital elevation models (DEMs) to direct the rendering of landform features in the style of pen-and-ink landscape illustrations. Unlike image space procedures which generally extract feature information through 2-dimensional (2D) rendered image processing techniques, object space procedures operate directly on 3-dimensional (3D) surface models. The ultimate goal of this work is to produce fully automated tools that can imitate the styles characterized by the traditional pen-and-ink techniques of Lobeck, Imhof, Raisz, and other scientific illustrators to bring the beauty of their effective and economical visual techniques to automated cartographic environments. Through the implementation of a Java 3D application programming interface (API) prototype, a testing platform was established for the application of stylistic elements to linework representing surface form lines. This paper explores the aesthetic effects of silhouette lines, creases, and slope lines on the rendering of terrain features. It also introduces the use of adaptively resampled triangulated irregular networks (TINs) as a basis for perspective rendering applications.

Introduction

The art of perspective landscape illustration has a long history in the study and description of landforms and their underlying geological structures. Imhof (2007, pp. 1-14) traced its development from Mesopotamian relief representations through Renaissance copper engravings and on to modern illustrations derived from topographic map construction. Flemer (1895) provided an exhaustive treatment of the theory and photographic technology supporting ground-based, perspective imaging of landforms. His work specifies methods for capturing panoramic images based on numerous individual photographs with known fields of view, and for constructing topographic maps from the panoramas. Raisz, himself one of the foremost practitioners of block and landform diagram construction, attributed the modern introduction and perfection of this form to the work of Grove Karl Gilbert and William Morris Davis (Raisz 1948, p. 301-308). Armin Lobeck, whose *Block Diagrams* has remained one of the most influential guides to their manual construction, argued that these pen-and-ink illustrations gain much of their effectiveness from their visual simplicity (Lobeck 1958, p. 1). This research explores the application of Lobeck's aesthetic to the automatic construction of perspective landscape illustrations from digital elevation models (DEMs) under the rubric of non-photorealistic rendering. It reviews earlier work on automated systems, introduces a new testing platform for stylistic experimentation, presents the relevant underlying algorithms for the production of the major visual elements, and discusses the contributions of these elements to good expression of landscape form.

James E. Mower, University at Albany, Department of Geography and Planning, AS 218, University at Albany, Albany, New York 12222 USA. E-mail: <jmower@albany.edu>. Tel: 518-442-4779; Fax: 518-442-4742.

Previous Related Work

Over the past decade, the rapid development cycle of fast, relatively inexpensive graphics processing chips, along with concomitant developments in 3D application programming interfaces (APIs), has fostered a period of intense growth in perspective rendering applications in cartography, GIS, and in non-geographic disciplines. Much of this work has been related to realistic renderings of landscapes, whether for gaming, site planning, or other uses that require a relatively "complete" description of a scene. Kennelly and Kimerling (2006) noted that another rendering paradigm, non-photorealistic rendering (NPR), provides a close analogy to the construction of traditional landscape line drawings by fostering the communication of specific, often scientific information through a language composed of symbols with well understood meanings. The goals of NPR are consistent with those of cartographic forms that use the rules and con-

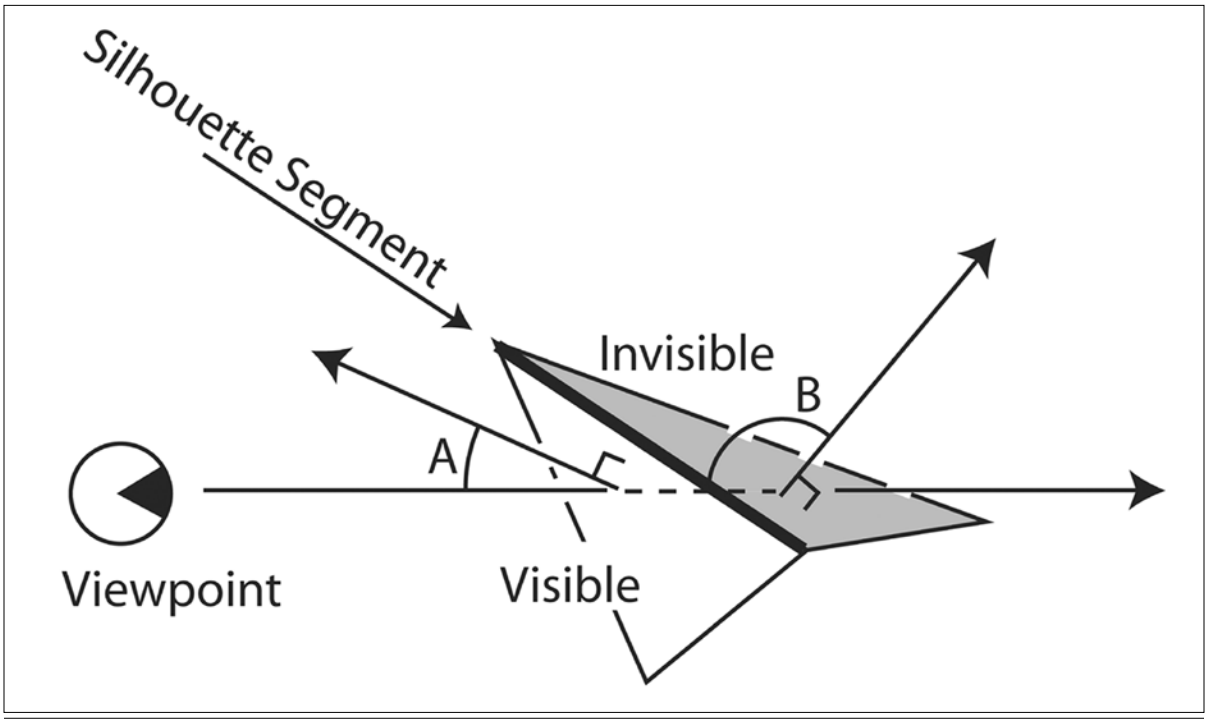

Figure 1. An edge between two surface facets is a silhouette segment if one facet is visible (with a normal less than 90 degrees from the line of sight (angle A)) and the other is invisible (normal greater than 90 degrees away from the line of sight (angle B)).

straints of generalization to establish a message for an image and to focus a viewer's attention upon that message.

Saito and Takahashi (1990) laid out many of the ground rules for NPR in image space, using information from projected 2D images of 3D objects. Their successful demonstration of techniques for generating simplified line drawings from photographed 3D objects established a vocabulary for further NPR research on the extraction of edges (rendered as silhouette lines) and components of surface curvature, represented by hachures. Isenberg et al. (2003) provided a comprehensive classification of NPR modeling techniques for both image and object space approaches along with definitions of terminology. Kennelly and Kimerling (2006) provided an excellent overview of NPR approaches that are specific to renderings of landforms, placing them in the context of the manual methods defined and described by Raisz, Imhof, Lobeck, and other 20th century cartographers and illustrators.

Most current NPR research in landform representation has focused on object space approaches that derive perspective line-drawn images entirely from 3D spatial data represented in a digital elevation model (DEM). Lesage and Visvalingam (2002) used this approach to implement p-stroke sketching, which is itself closely related to the inclined contour method of Tanaka (1932). Buchin et al. (2004) modeled surface slope to apply scanned, hand-drawn slope and loose lines from a look-up table, dependent on surface slope and lighting conditions.

This paper focuses on the development of a robust technique for representing surface form lines from adaptively resampled DEMs structured as variable resolution triangulated irregular networks (VTINs). By resampling a DEM with greater tolerance for surface error away from a selected viewpoint, areas of triangles remain relatively constant (given a surface with low variability) with distance from the viewpoint under perspective rendering. It is thus possible to extract and render features from a resampled DEM without line simplification as a post-rendering operation. This paper also introduces the extraction of drainage features from DEMs to aid in the rendering of creases—view-independent features that contribute visual cues for surface interpretation.

Modeling Surfaces with Linework

Non-photorealistic terrain renderings must account for both view-dependent and view-invariant features. A silhouette, representing the profile of a ridge or hill against the sky or another background feature, is unique to a given observer. A crease, on

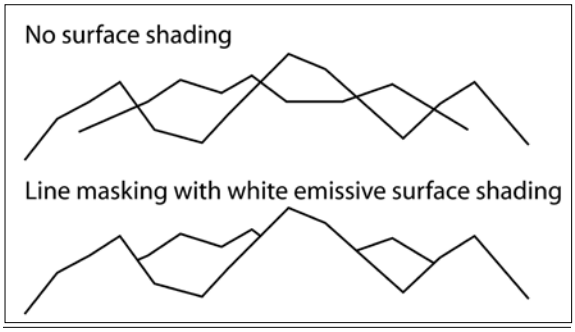

Figure 2. In the upper image, silhouettes are rendered without their associated surfaces and intersect inappropriately. In the lower image, surfaces are rendered with emissive shading in the background color, masking background linework. The latter technique is used implicitly by all of the project's line rendering algorithms.

the other hand, represents a feature like a prominent hydrologic ridge, a spur, a drainage channel, or some other surface discontinuity that holds important visual cues regardless of its orientation to the viewer. A crease is frequently accompanied by slope lines that show surface curvature away from the discontinuity. This research implements silhouettes, creases, and slope lines to render surfaces in the style of landscape illustrations.

Throughout this paper, all facets are represented as 3D triangles within a VTIN. Each VTIN triangle is defined by its vertices and edges, where each edge record includes the IDs of its bordering triangles. The network of triangles is traversable both by triangle and edge IDs.

Silhouettes

A silhouette is composed of individual line segments. Each silhouette segment is a rendered edge separating visible and invisible facets, where visibility is determined by the angle made between the viewer's line of sight and the facet's surface normal (Figure 1). In this sense, visibility has nothing to do with the presence or absence of obscuring surfaces between the viewer and the facet in question. A silhouette can be thought of as a "visibility ridge" from the viewer's point of view. Silhouette segments are computed locally as specified by Algorithm *Build Silhouettes*:

Algorithm *Build Silhouettes*
To build silhouette line segments:
1. For each facet:
   1.1. Determine the angle between the viewer's line of sight and the surface normal.
   1.2. If the angle is between 0 and 90 degrees, mark the facet as visible. Otherwise, mark it invisible.
2. For each edge,
   2.1. If one of the bordering facets is visible and the other is invisible.
   2.2. Render the edge as a silhouette segment.

Since hidden line removal is not an integral part of this algorithm, silhouette segments are rendered along with their referent surfaces, themselves rendered with emissive lighting in the background color of the rendered frame. Emissive lighting provides no differentiation in surface shading with respect to light source direction and surface normals. When rendered in the background color, surfaces disappear, yet their opaque shading supplies appropriate masking for silhouette segments and other form lines, as shown in the lower of the two images in Figure 2. This technique is used implicitly by each of the line rendering algorithms presented in this paper.

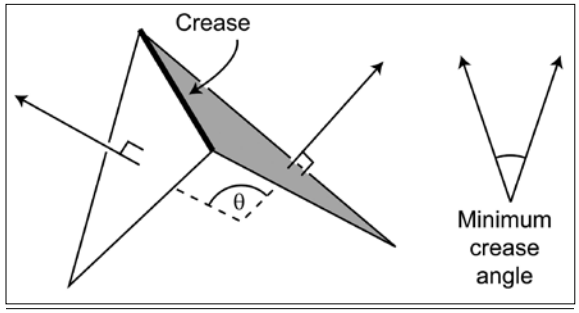

Figure 3. If the angle θ between the two adjacent facets is greater than a preselected minimum crease angle, the edge between the two facets will be rendered as a crease.

Surface Creases

In the NPR literature, a crease is a line representing an edge that carries important visual information independent of the viewer's line of sight (Isenberg et al. 2003, p. 28). Drainage features such as channels and ridges represent creases in landscape illustrations and can be extracted from DEMs in several ways. One approach examines the angle made by normals of adjacent triangles and renders their shared edge as a crease if the angle exceeds a pre-selected value (e.g., 20 degrees), either as a convex or concave feature with respect to the projected plane of the ellipsoid (Figure 3).

Although this analysis may provide adequate rendering of long and sharply defined edges for buildings or other objects in the built environment, it does not provide particularly good results for topographic surfaces. In 10-m horizontal reso-

lution elevation models (such as those used for this paper), ridge and channel features rarely exhibit sudden slope breaks along connected triangle edges. In Figure 4, an actual surface (represented by the underlying block diagram) presents a visually interesting feature but the feature is "spread out" among a region of triangles with relatively small slope transitions along adjacent edges. The detection method illustrated in Figure 3 was originally implemented for this paper but was later abandoned when it was found to produce an abundance of disjoint crease segments that did little to suggest surface curvature.

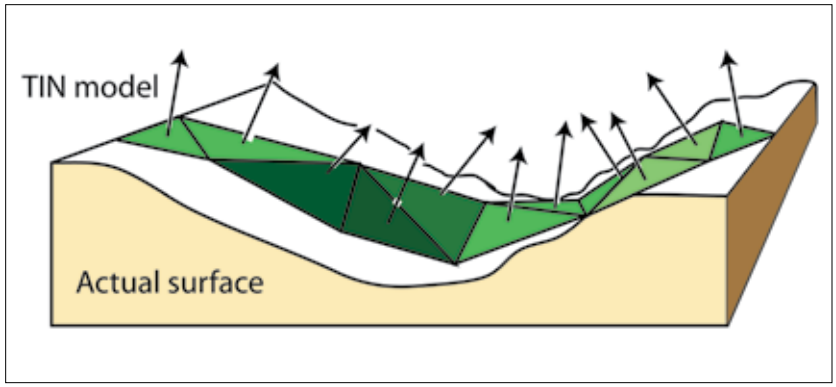

Figure 4. A surface that presents a problem to the crease detection method illustrated in Figure 3.

It was found that a drainage accumulation model provided a visual solution much closer to that of hand-drawn pen-and-ink illustrations. In the case of TIN-based DEMs (the data structure used in this research), a drainage accumulation model identifies triangles that represent drainage channel segments given a sufficiently small drainage threshold value. A drainage direction map is first created to specify which triangle of each triangle's neighbors serves as its drainage outlet. The drainage accumulation model then starts by providing each triangle with a unit of water. The model iterates, draining the contents of each triangle to its downhill neighbor on each pass and records the total amount of water (the accumulation) that has passed through each triangle. After all the water has drained into triangles with no downhill neighbors, the accumulated values for each triangle are compared to a pre-selected threshold value. If the accumulated value is greater than the threshold, the triangle is considered to lie along a drainage channel. Algorithm *Build Creases* summarizes this approach.

Algorithm *Build Creases*
To build creases:
1. Initialize the drainID for each facet to RIDGE
2. For each facet:
   2.1. Find the centroid of the neighboring facet with the steepest downhill slope from the current facet. Note its ID as the current facet's drainID. If a facet has no downhill neighbor, call it a pit.
   2.2. Initialize all facets with 1 unit of input water and 0 units of accumulated water.
3. While any but pit facets contain greater than 0 units of input water:
   3.1. For each facet.
      3.1.1. Decrement its total input water value and increment both the drainID facet's input water value and its accumulated water value by the total input water value.
4. For each facet:
   4.1. Render a line between the current facet's centroid and that of the drainID if the drainID facet's accumulated water value is greater than a preset threshold value.

Build Creases provides a localized method for constructing drainage lines, requiring only that each facet knows the ID of its downhill neighbor. Steps 1 and 2 create a drainage direction map, indicating the drain for each triangular facet. Following step 2, those triangles with no uphill source are on a ridge. Pits are defined here as triangles that do not drain to a neighboring triangle.

This research uses TIN-based DEMs derived from USGS 1:24,000 format grid cell DEMs at 10 m horizontal resolution, produced by the New York State Department of Environmental Conservation (Cornell University Geospatial Information Repository 2005). Most drainage direction models built from data originally extracted from grid cell DEMs will find numerous internal pits. False internal pits are by-products of grid sampling patterns, occurring when an elevation sampled along a stream channel is lower than its eight surrounding neighbors on the bank above (O'Callaghan and Mark 1984; Mower 1994). Since the channel itself is not explicitly extracted, channel samples are frequently separated from one another and thus have no continuous drainage path. Unlike the O'Callaghan and Mark or Mackay and Band (1998) hydrologic drainage models, the crease algorithm does not, for now, "flood" false pits to assume a regular drainage pattern. This feature will be added in the next version.

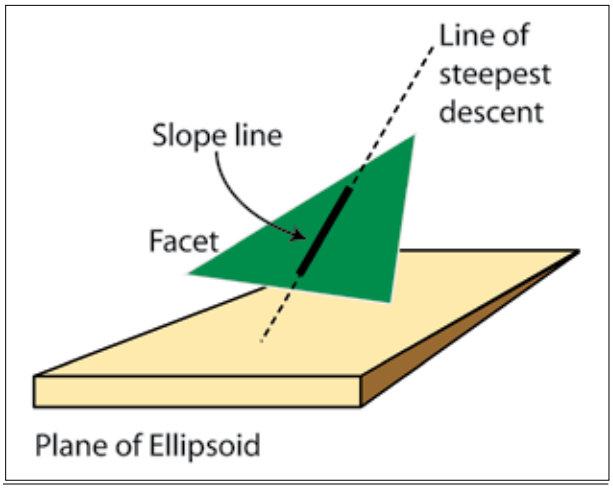

Figure 5. A hachure is computed as a segment of the line of steepest descent through the facet.

Step 3 runs a drainage accumulation model through the drainage direction map. At each iteration, an uphill neighbor transfers its water supply to its downhill neighbor (indicated by drainID), which tracks the total amount of water passing through itself over the life of the procedure. Step 3 ends when all the remaining water is in the pit facets.

Step 4 inspects the drainage accumulation of each downhill neighbor of a facet. If the accumulated value is above a preset threshold, a line segment is rendered between the two triangle centroids. Depending on the value of the threshold, the procedure will draw more or less of the drainage network detail. For the moment, at current threshold values, ignoring false pits does not negatively impact the line drawing. However, given that drainage networks are promising crease generators, future enhancements may require an additional flooding procedure to avoid potential artifacts.

Slope Lines

On a planimetric map, hachures are linear symbols drawn perpendicular to contours and represent paths of steepest slope across a surface patch. On a landscape illustration, slope lines perform a different function, emphasizing surface curvature as well as slope. Several strategies for using linework to show surface trends were explored: 1) employing simple hachures as lines of steepest slope; 2) limiting hachure rendering to regions adjacent to stream channels (to suggest surface curvature); and 3) sampling elevations along surface profiles away from creases to render 3D curves. The first two strategies have been implemented and the third is currently under development and is discussed below in the section on Future Work.

The first technique, rendering hachures as lines of steepest slope, adds reasonable slope information to a landscape illustration rendering but does not imitate typical manual practice. Lobeck notes that slope lines should be placed near streams and drawn as a curve along a line of steepest descent (Lobeck 1958, p.33). Comprehensive straight-line hachure rendering is discussed here because it forms the basis of the second approach which more closely resembles Lobeck's instructions. The following algorithm, *Build Hachures*, outlines a procedure for constructing straight-line segments along the line of steepest descent through a facet for all facets with a slope greater than a preset value (Figure 5).

Algorithm *Build Hachures*
To build hachures:
1. For each facet:
   1.1. Find the coordinates of the facet centroid.
   1.2. Find the equation of the plane of the facet.
   1.3. If the facet slope exceeds the preset minimum value:
      1.3.1. Find the equation of the line of steepest descent through the facet centroid.
      1.3.2. Render the segment of the line within the facet up to a preselected margin bordering the edge of the facet.

The algorithm creates hachures of uniform width. Although it could be modified to scale the width of the rendered line to the slope of the facet, this is largely unnecessary. As vertical resolution decreases away from the viewpoint, the screen-space areas of rendered triangles remain relatively uniform given constant surface variability. Therefore, the separation of rendered hachures remains a good visual indicator of surface slope regardless of the 3D distance of the surface patch from the viewpoint.

Nonetheless, hand-drawn landscape illustrations rarely apply hachured renderings on surfaces that are not adjacent to stream channels. Most utilize curved lines parsimoniously to emphasize surface trends near creases. The second strategy, represented by algorithm *Build Slope Lines*, extends *Build Hachures* with a recursive technique that still renders hachures as straight lines but limits their rendering to triangles falling within a specified topological distance from a crease. Here, topological distance is defined as the minimum number of

edges that must be traversed from the interior of a given, non-crease triangle to that of a triangle containing a crease (Figure 6).

Topological distance is a convenient metric for line representation when used with a VTIN. In this project, densely-sampled TINs (DTINs) are first built from USGS 1:24,000 format grid cell DEMs using a procedure based on Heller (1990). Heller's procedure recursively divides a grid cell DEM into triangles, comparing the elevation of each sample within its enclosing triangle to its predicted value derived from the equation of the triangle's plane at the sample's horizontal coordinates. If the greatest difference between an interior sample's given elevation and its predicted value is greater than a preset tolerance, the triangle is subdivided at that location and the analysis continues on the children of the subdivided triangle, ending when no remaining elevation difference exceeds the tolerance. The DTIN builder extends Heller's algorithm by recording the difference between each sample's elevation and its predicted value. A VTIN resamples a DTIN by evaluating a sample with respect to a critical angle, calculated by dividing the scene's field of view by the display height in pixels (Mower in press). If the recorded elevation difference of the sample exceeds the tangent of the critical angle at its distance from the user-selected viewpoint, it is included in the VTIN. By matching the VTIN resampling tolerance to the target display resolution, triangle sizes are kept relatively constant with respect to image depth from the viewpoint. Consequently, a constant topological distance becomes a reasonable metric for surfaces with diverse variability.

Algorithm *Build Slope Lines*
To build slope lines:
1. Establish a maximum rendering distance (in topological distance) from a crease.
2. For each facet containing a crease segment:
2.1. Apply procedure Find Slope Region to each of the facet's neighbors with the maximum rendering distance criterion.

Recursive Procedure *Find Slope Region*
1. Decrement the rendering distance criterion
2. If the rendering distance criterion is not less than 0:
  2.1. Note that this facet has been visited.
  2.2. For each of the facet's neighbors:
    2.2.1. Find the coordinates of the neighboring facet's centroid.
    2.2.2. Find the equation of the plane of the neighboring facet.

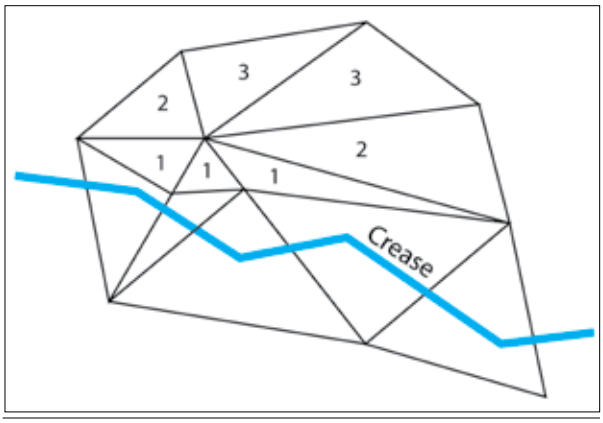

Figure 6. The topological distance of a noncrease triangle equals the minimum number of edges that must be traversed from its interior to that of the nearest triangle containing a crease. The numbers within the triangles indicate their topological distances.

    2.2.3. If the neighboring facet has not been visited, if it is not on a crease, and its slope exceeds the preset minimum value:
      2.2.3.1. Find the equation of the line of steepest descent through the facet centroid.
      2.2.3.2. Render the segment of the line within the facet up to a preselected margin bordering the edge of the facet.
      2.2.3.3. Apply procedure Find Slope Region to its neighboring facets.

This algorithm ensures that slope lines will only be drawn within a predetermined distance from a crease. By using topological distances rather than ground or pixel distances, the algorithm leaves generalization issues to the VTIN generator.

The Implementation

The landscape illustration implementation, referred to here as PenAndInk, has been written in Java and Java 3D API. Although earlier work on this and related rendering projects used C# and Direct3D API, Direct3D (the Microsoft 3D rendering library) was found to provide insufficient control for line widths, a feature which is critical to the successful implementation of automated NPR landscape illustration. OpenGL, an alternative to Direct3D, provides such control, and since Java 3D API makes OpenGL calls on a client computer by default, this environment provides a natural platform for experimentation and collaboration across the Internet.

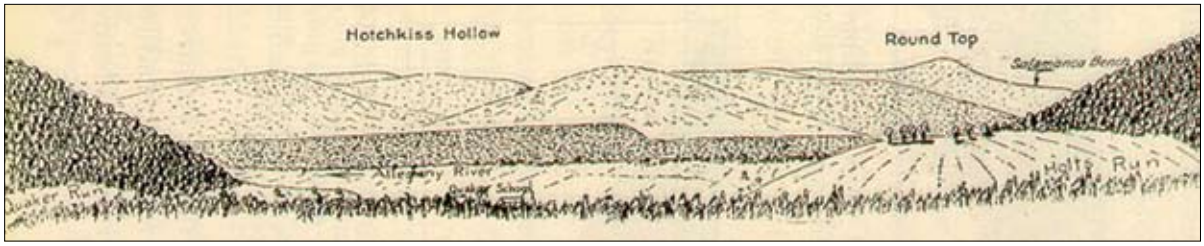

Figure 7. Lobeck's drawing of the view from a point within Allegany State Park in New York, near Quaker Run, and looking toward Hotchkiss Hollow on the west side of the Allegany River. Printed with permission of the New York State Museum, Albany New York, 12,203.

The implementation supports the rendering of a scene in layers. After loading the VTIN, the user typically renders the scene with the emissive background color. For debugging purposes, the surface can be rendered with flat shading in any non-emissive color, enabling the type of image shown in Figure 8. Once the surface has been rendered, slope lines, silhouettes, or creases can be rendered with a button push. The user may toggle a rendered layer once it has been created and can zoom, pan, or tilt the surface with the mouse. Because the creation of silhouette lines is view dependent, the user would normally render a new set of silhouette lines after a significant shift in viewing position. The user requests a new set of silhouette lines with a button push. A new set of silhouette lines is usually rendered in less than 2 sec real time on the development platform running Windows XP on a 3.2 GHz processor with 3 GB main memory and an NVIDIA GeForce video card.

Results

Figure 7 is Lobeck's rendering of a scene in the region of the Allegany State Park in New York State, included in his book *A Poplar Guide to the Geology and Physiography of Allegany State Park* (Lobeck 1927, p 192). The original image caption notes that the viewpoint is on a "high terrace near Friends Indian School at [the] mouth of Quaker run" showing the "high terrace across the river at [the] mouth of Hotchkiss hollow." Figures 8, 9, 10, 12, and 13 show renderings of a VTIN covering the region that Lobeck describes. This scene was selected for this research for several reasons: it includes several different types of terrain features; data for the scene were readily available in a format suitable for the application; and the coordinates of the viewpoint, the direction of the line of sight, and the field of view could all be reasonably estimated. Each image shares a common viewpoint (UTM zone 17N, 676,200 m E, 4,657,800 m N, elevation 450 m), view azimuth (300 degrees), view altitude (0 degrees) field of view (60 degrees), and 2X vertical exaggeration, producing a view comparable to Lobeck's rendering. The author verified that ArcGIS and Google Earth rendered similar relief to that in Figure 8 using the same viewpoint coordinates and viewing angles. All of the images were taken as screen shots from PenAndInk, captured at actual rendered size and framed with a border for this paper. Lobeck's illustration in Figure 7 was reduced to 82 percent of its original size to fit within the margins of this paper and to match the size of the rendered VTIN images.

The VTIN supporting the images in Figures 8, 9, 10, 12, and 13 was centered on the rendering viewpoint. The VTIN region is itself defined by a circle with an 8-km radius. In Figure 8, the scene has been rendered with flat shading to provide the reader with a frame of reference for subsequent line renderings. Flat shading renders each triangle with a value that is held constant across its surface. Although other types of surface rendering (such as Gouraud shading) blend values between adjacent triangle vertices to reduce the visibility of edges, flat shading is often more useful during development and debugging operations when it is helpful to see triangle edges. The image was annotated manually to identify important features found in Lobeck's drawing.

It is immediately apparent that the profile of some of the hills in Figure 8 and their representative silhouettes in Figure 9 are sharper than those presented by Lobeck. Other features, especially Round Top, are less pronounced than those in Lobeck's depiction. Three possible causes for these visual discrepancies come to mind. First, the sampling resolution of the underlying DTIN model may be insufficient to pick up the curvature that Lobeck rendered. Second, the presence of forested land may have provided the artist with a smoother silhouette than that of the denuded landscape represented by the DTIN. Third, Lobeck may have used his artistic prerogative to soften the silhouettes in the landscape, just as it appears that he provided some vertical exaggeration. A photograph of the scene taken from Lobeck's viewpoint

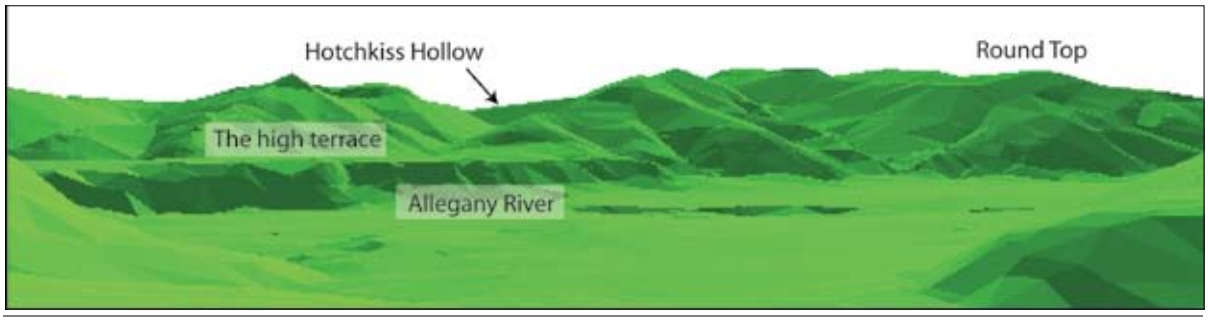

Figure 8. Flat shaded rendering of VTIN at approximate viewpoint of Figure 7. This and the following images share a common viewpoint, view angles, and other rendering parameters.

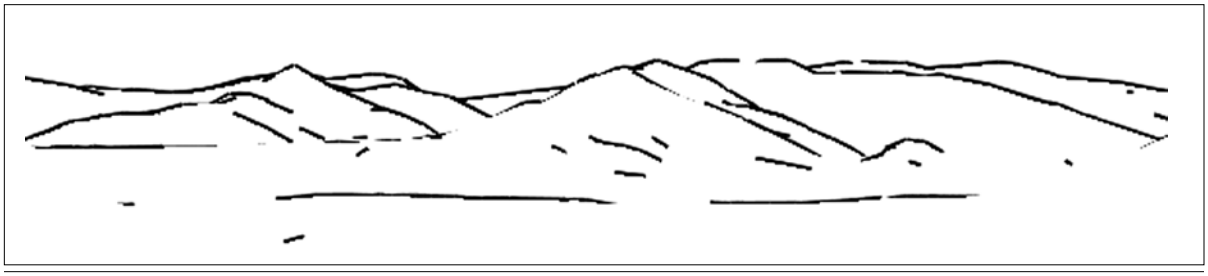

Figure 9. Silhouette rendering. This and the remaining line drawings render the surface with white emissive color to enable hidden line removal.

Figure 10. Crease rendering as drainage lines. False pits have not been removed from the drainage direction map.

could certainly confirm whether or not his drawn silhouettes were faithful to those in the landscape. If they are, the first possibility could be addressed by decreasing the vertical tolerance of the DTIN or by interpolating additional samples along curved surfaces between existing DTIN sample points. Since the DTIN data are already very near the original resolution of the source 10 m DEMs, any further curvature would likely require interpolation. Such interpolation could also be used to model the continuous surface of a forest canopy.

Figure 9 renders the scene in silhouette lines with white emissive surface shading. The image captures the profile of the mountains and provides some delineation of the foreground "high terrace" noted by Lobeck. Although the silhouette suggests a sharper profile than does Lobeck's illustration, it is faithful to the position of features on the flat shaded rendering in Figure 8. The silhouette lines were rendered with 3-point width to contrast them with the crease and slope lines rendered in 1-point width on the composite image (Figure 13).

Figure 10 renders the drainage accumulation data as a set of crease lines. The image provides additional horizontal linework that helps to focus attention on the high terrace. Although the image provides strong depth cues when combined with the silhouette and slope lines, Lobeck's image in Figure 7 uses a stipple pattern to achieve this effect. This approach is somewhat unusual for his work; he more often depicts slope with linework, as characterized by the image in Figure 11 (Lobeck 1927, p. 218) and most of the drawings shown in Lobeck (1958).

Figure 12 renders the scene with *Build Slope Lines*. Rendering was limited to only those facets with a topological distance no greater than 3 from a triangle containing a crease segment. Since the

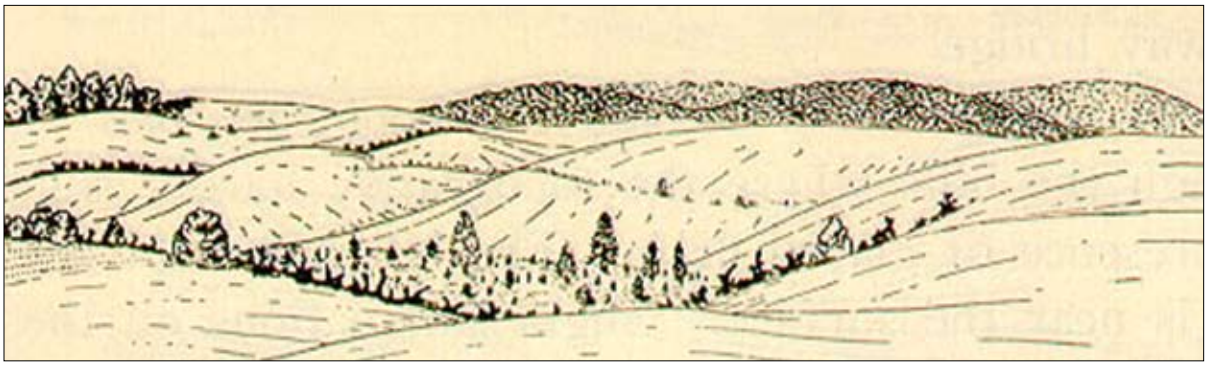

Figure 11. A scene drawn by Lobeck using linework in the foreground to suggest surface slope and curvature. His use of linework for this purpose is more typical in his images than the use of stipple patterns (as seen in the background in this image and throughout the image in Figure 7). Printed with permission of the New York State Museum, Albany New York, 12203.

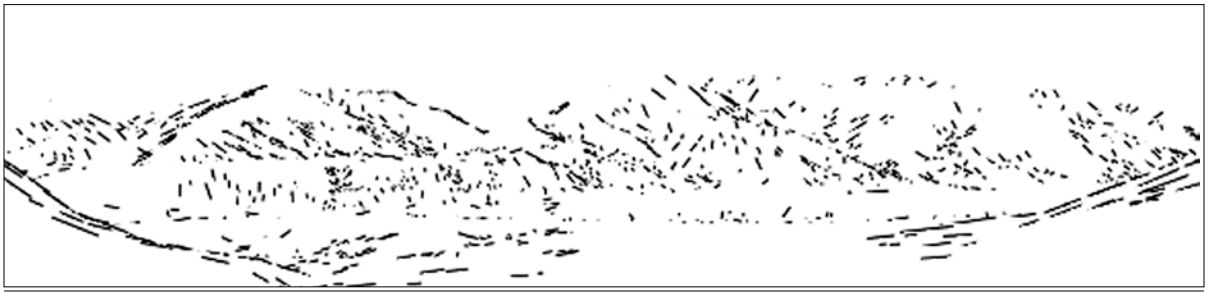

Figure 12. Slope line rendering. A line is rendered within a triangle if the slope is greater than 10 degrees and the triangle has a topological distance of 3 or less from a triangle containing a crease. The crease layer has been turned off for clarity.

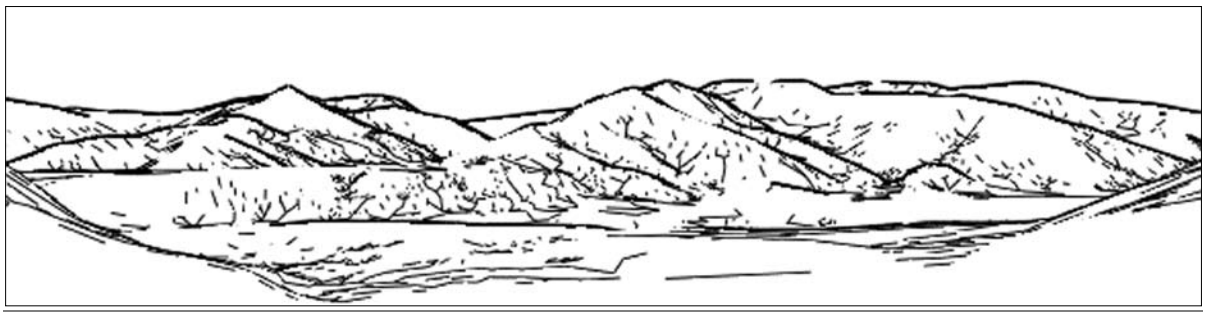

Figure 13. The composite image, including emissive white surface shading, silhouette lines, crease lines, and slope lines. Silhouettes are rendered in 3-point width; crease and slope lines are rendered in 1-point width.

VTIN maintains a relatively constant rendered triangle size regardless of scene depth, slope lines, rendered at no more than 1 per triangle, do not coalesce in the background; differences in triangle size (and thus hachure stroke length) are more strongly influenced by local surface variability. The current implementation limits the rendering of slope lines to straight-line segments. In this image, slope lines are drawn with 1 point width.

Figure 13 shows a screen shot of the composite image including the rendered surface in emissive white, the silhouette lines, crease lines, and slope lines. The image has many elements in common with Lobeck's drawing, especially with respect to the delineation of individual peaks and the extraction of the high terrace across the river. The positions of the creases and slope lines add depth to the image, producing a reasonable overall impression of a mountain scene. However, the image still lacks some of the flowing quality achieved by Lobeck's illustration. The remaining portion of this paper will outline several research paths that will move toward a more faithful pen-and-ink style representation, resulting in an improved visual experience.

Future Work

The next phase of this project will investigate three major themes: 1) changing the represen-

tation of slope and crease lines from simple, straight-line segments to complex curves; 2) exploring other techniques for crease generation; and 3) implementing several features to support on-the-fly calculations.

Build Slope Lines produces renderings that are still somewhat deficient for the landscape drawn by Lobeck in Allegany Park. It was expected that triangles near creases would have lines of steepest descent more or less perpendicular to their referent creases. Instead, lines more than a single triangular facet away from a crease have a tendency to follow surface trends that are parallel to, rather than normal to the drainage creases. Figure 14 illustrates the problem in a close-up view of slope line and crease renderings. In A, most slope lines run parallel to their referent creases. In B, slope lines were drawn by hand to demonstrate a more reasonable approach. The solution illustrated in Figure 14, inset B is encapsulated in the algorithm *Build Curved Slope Lines*.

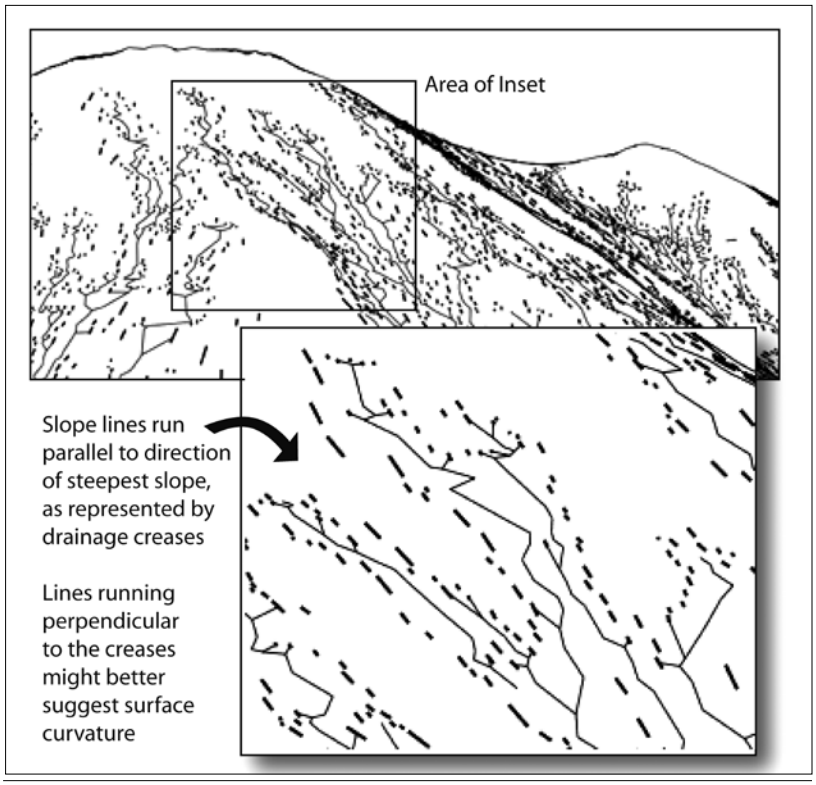

Figure 14. A problem and a solution for slope line rendering.

Algorithm *Build Curved Slope Lines*
To build curved slope lines:
1. For each crease:
    1.1. Set n to a desired topological distance and selectedSide to LEFT.
    1.2. Starting at one end of the crease, for each vertex (in order along the crease):
        1.2.1. Create a new curved slope line list.
        1.2.2. Establish a 2D vector in Easting (E) and Northing (N) extending in triangles away from the selectedSide of the vertex.
        1.2.3. For each triangle whose projection onto the E, N plane is intersected by the 2D vector:
            1.2.3.1. Find the E and N of the triangle at its midpoint along the vector.
            1.2.3.2. Find the elevation (Z) at the midpoint using the 3D equation of the intersected triangle.
            1.2.3.3. Save the triple E,N,Z to the curved slope line list for this vertex.
        1.2.4. If selectedSide = LEFT, set it to RIGHT. Otherwise, set it to LEFT.
2. Render the list of triples for each vertex as a curve.

Algorithm *Build Curved Slope Lines* renders elevations along 2D profiles, extending a preset topological distance away from crease vertices (Figure 15). Along a single crease, profiles will alternate from left to right to break up symmetry. Each profile is checked for intersection with VTIN triangles projected onto the Easting, Northing plane. If a profile intersects a triangle, an elevation is interpolated at the 2D coordinates of the triangle's midpoint along the vector using the equation of the triangle's plane. The interpolated points are saved to a list unique to each profile and rendered as 3D curves. To ensure that the rendered lines do not intersect one another, the implementation will limit each facet to a maximum of one representation. Silhouette lines will be given first priority, then crease lines, and finally curved slope lines on a first-come, first-served basis.

The author is currently developing a 3D Bezier curve implementation for creases. Buchin et al. (2004) described a procedure for creating curves as an application of a vector field visualization but their approach requires a complex iterative pro-

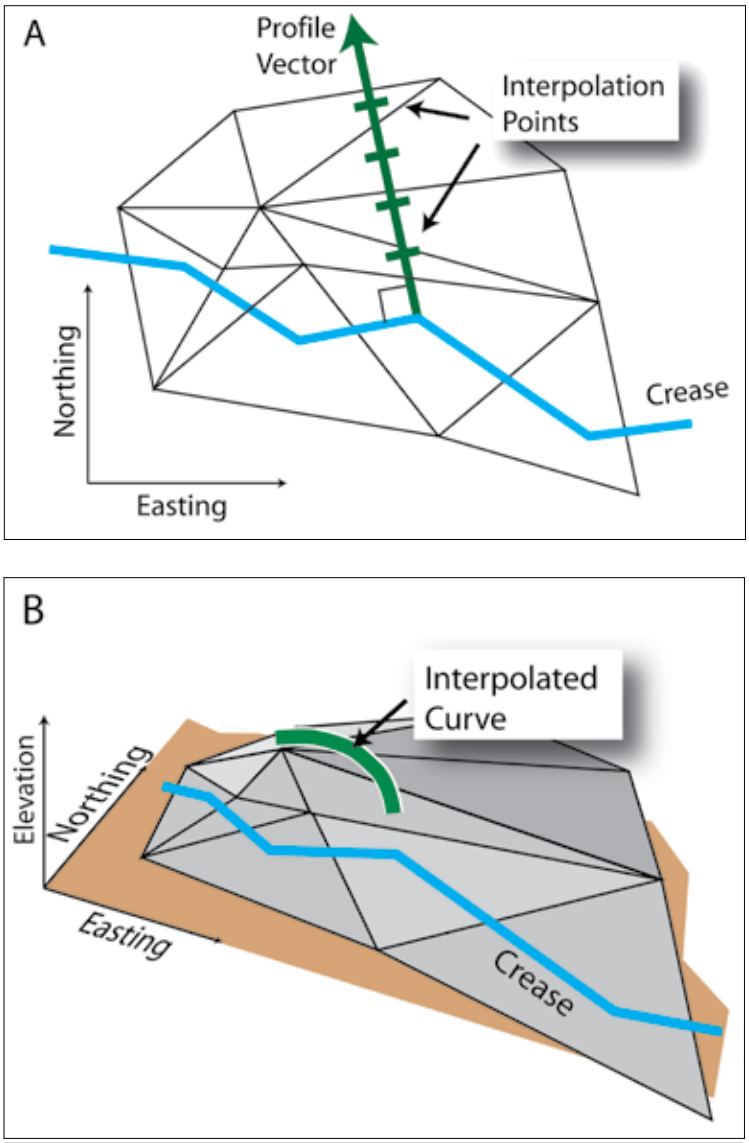

Figure 15. Interpolating elevations for slope lines along vectors. In A, a profile is constructed as a vector normal to the crease segment with an interpolation point for each intersected triangle. In B, the interpolated points have been connected and rendered as a 3D curve.

cedure for line intersection detection. Creases can be drawn more simply as Bezier curves within the drainage framework by using the points extracted from the accumulation model as control points. Bezier curves are particularly attractive forms for this application because such curves are guaranteed to remain entirely within the convex hull described by their control points (Foley and Van Dam 1984, p. 521). Thus, drainage lines for separate basins should not intersect one another.

PenAndInk identifies, but does not render, lines between points along drainage basin ridges. Rendered as straight-line segments, such lines are good candidates for illustrating exposed rock faces. Imhof (2007, pp. 235-260) provides numerous examples of rock-face drawing in which both hydrologic channels and ridges are rendered as straight-line segments, some connected in branching patterns and others in which the linework is largely disconnected. The next implementation will explore the rendering of rock-face landforms and evaluate the contribution of both connected and disconnected straight-line ridge and drainage networks for such illustrations.

To support development and debugging, the prototype described here allows the user to recalculate slope lines, creases, and silhouettes at any time through a button push. In a finished application, silhouette construction would need to be recomputed on the fly as the viewpoint changes. If its position shifts beyond an established threshold, the implementation would also need to calculate a new VTIN and underlying surface rendering. Furthermore, because the shift in position would change the resolution of the surface upon which the slope lines and creases are built, they should be recalculated as well. Future work will begin to implement automatic resampling and rendering on changes in the user's viewpoint. It will also improve the drainage model by adding a procedure for false internal pit elimination. Once the implementation has become somewhat stabilized, it will be converted to a Java applet for testing and collaboration over the Internet.

Conclusions

Much of the beauty of pen-and-ink renderings of landscapes is expressed through an economy of strokes. By establishing a basic set of algorithms for drawing silhouette, crease, and slope lines, this research has established a platform for the refinement of current graphic elements and the introduction of new elements that will further contribute to aesthetically pleasing and informative terrain representations. The author has begun to expand the work through the development of line-rendering procedures for specific

types of features, allowing alternate interpretations for rock faces, glaciated terrain, and other types of landforms. A comprehensive landscape illustration package arising from such work should be able to highlight interesting surface features in ways that shaded perspective models do not. Advances toward this goal will come with the development of additional methods for extracting terrain features from 3D models and with the continued refinement of line-rendering algorithms.

It is hoped that this research and others like it will focus attention on the work of Lobeck, Imhof, and other masters of line-drawing techniques and help to develop adequate methods for their imitation in digital form. Although such research is only beginning to explore and automate their styles, it may provide an interesting nexus for the integration of artistic, geomorphological, and geometric principles for perspective renderings of landscapes.

REFERENCES

Buchin, K., M.C. Sousa, J. Dollner, F. Samavati, and M. Walther. 2004. Illustrating terrains using direction of slope and lighting. In: *4th ICA Mountain Cartography Workshop*, Vall de Núria, Catalunya, Spain, September 30-October 2 2004. [http://www.mountaincartography.org/ publications/papers/papers_nuria_04/buchin.pdf.]

Cornell University Geospatial Information Repository (CUGIR). 2005, *New York State Digital Elevation Models—Metadata*. [http://cugir.mannlib.cornell.edu/browse_lis/dem_list.html.]

Flemer, J.A. 1895. Phototopography as practiced in Italy under the auspices of the Royal Military Geographical Institute, and as practiced in Canada under the auspices of the Department of the Interior. Also a short historical review of other photographic surveys and publications on the subject. In: *Report of the Superintendant of the U.S. Coast and Geodetic Survey for the Fiscal Year Ending June 30, 1893, Part 2, Appendix No. 3. Washington, D.C.: Government Printing O*ffice, pp. 37-116. [http://docs.lib.noaa.gov/rescue/cgs/003_pdf/CSC-0103.PDF.]

Foley, J.D., and A. Van Dam. 1984. *Fundamentals of interactive computer graphics*. Reading, Massachusetts: Addison-Wesley Publishing Co.

Heller, M. 1990. Triangulation algorithms for adaptive terrain modeling. In: *Proceedings, 4th International. Symposium on Spatial Data Handling*, Zurich, Switzerland, July, 1990. pp. 163-74.

Imhof, E. 2007. *Cartographic relief presentation*. Redlands, California: ESRI Press.

Isenberg, T., B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte. 2003. A developer's guide to silhouette algorithms for polygonal models. *IEEE Computer Graphics and Applications* 23(4): 28-37.

Kennelly, P.J., and A.J. Kimerling. 2006. Non-photorealistic rendering and terrain representation. *Cartographic Perspectives* 54: 35-54.

Lesage, P.L., and M. Visvalingam. 2002. Towards sketch-based exploration of terrain. *Computers & Graphics* 26: 309-28.

Lobeck, A.K. 1927. *Popular guide to the geology and physiography of Allegany State Park*. Albany, New York: The University of the State of New York.

Lobeck, A.K. 1958. *Block diagrams and other graphic methods used in geology and geography*. Amherst, Massachusetts: Emerson Trussell Book Company.

Mackay, D.S., and L.E. Band. 1998. Extraction and representation of nested catchment areas from digital elevation models in lake-dominated topography. *Water Resources Research* 34(4): 897-901.

Mower, J.E. 1994. Data-parallel procedures for drainage basin analysis. *Computers & Geosciences* 20(9): 1365-78.

Mower, J.E. Creating and delivering augmented scenes. *International Journal of Geographic InformationScience*. in press.

O'Callaghan, J.F., and D.M. Mark. 1984. The extraction of drainage networks from digital elevation data. *Computer Vision Graphics and Image Processing* 28(3): 323-44.

Raisz, E. 1948. *General cartography*. New York, New York: McGraw-Hill Book Company, Inc.

Saito, T., and T. Takahashi. 1990. Comprehensible rendering of 3-D shapes. *Computer Graphics* 24(4): 197-206.

Tanaka, K. 1932. The orthographic relief method of representing hill features on a topographic map. *The Geographical Journal* 79(3): 213-9.