Real-time Terrain Mapping

Tony Bernardin[1], Eric Cowgil[2], Ryan Gold[2], Bernd Hamann[3], Oliver Kreylos[3], and Alfred Schmitt[1]

1 **Institut für Betriebs- und Dialogsysteme, Universität Karlsruhe**
`{ug1g,aschmitt}@rz.uni-karlsruhe.de`
2 **Department of Geology, University of California, Davis**
`{cowgill,gold}@geology.ucdavis.edu`
3 **Institute for Data Analysis and Visualization, University of California, Davis**
`{hamann,kreylos}@cs.ucdavis.edu`

Abstract

We present an interactive, real-time mapping system for digital elevation maps (DEMs), which allows Earth scientists to map and therefore understand the deformation of the continental crust at length scales of 10 m to 1000 km. Our system visualizes the surface of the Earth as a 3D surface generated from a DEM, with a color texture generated from a registered multispectral image and vector-based mapping elements draped over it. We use a quadtree-based multiresolution method to be able to render high-resolution terrain mapping data sets of large spatial regions in real time. The main strength of our system is the combination of interactive rendering and interactive mapping directly onto the 3D surface, with the ability to navigate the terrain and to change viewpoints arbitrarily during mapping. User studies and comparisons with commercially available mapping software show that our system improves mapping accuracy and efficiency, and also enables qualitatively different observations that are not possible to make with existing systems.

1998 ACM Subject Classification I.3.7 Three-Dimensional Graphics and Realism, J.2 Physical Sciences and Engineering

Keywords and phrases Earth, Space, and Environmental Sciences Visualization, Interaction, Terrain Visualization, Multiresolution Visualization

Digital Object Identifier 10.4230/DFU.SciViz.2010.275

1 Introduction

Understanding how continents deform is a fundamental problem in Earth science [2]. Although the plate tectonic paradigm provides an accurate description of the behavior of oceanic crust, the theory fails to fully explain strain distribution within continents. There are currently two end-member views of the problem [3]. In one, continental deformation is spatially distributed over thousands of kilometers and thus, fundamentally differs from the plate-like behavior of the oceanic crust. In a second view, continents are mosaics of strong, rigid blocks bounded by weak faults, and thus mimic the behavior of oceanic plates. Distinguishing between these two views centers on determining the geometric and mechanical evolution of first-order ($\approx$ 1000 km long) intracontinental structural systems [3]. Do these systems of faults and folds remain stable in space and time for tens of millions of years (oceanic-plate like), or do they migrate, eventually producing spatially distributed deformation zones (diffuse pattern)?

Addressing this problem centers on determining how these 1000 km-long structural systems form and evolve over geologically intermediate time scales of a few tens of thousands to a few million years. At its core, developing this understanding requires mapping these structural systems. Specifically, mapping means identifying and measuring the 3D orientation of surfaces such as faults and folded layers of rock, along with the various topographic features by which these structures may be identified, such as contorted drainage networks or displaced ridges. Short-term deformation of a few seconds to a few thousands of years is straightforward to characterize using active, historical or paleo earthquakes as well as geodetic techniques such as VLBI, GPS, or InSAR. Likewise, long-term deformation that accumulates over tens of millions of years can be measured using thermochronology and geochronology. But understanding the intermediate record has proven difficult because it has been difficult to map.

At such intermediate time scales, active deformation of the Earth's surface produces detailed (10 m–100 m) topographic features by which active structures may be identified and mapped. However, such mapping has been hampered by the lack of both data and tools that permit efficient analysis of those data over the 1000 km × 1000 km areas that span regions of active continental deformation. In the last few years the first of these problems has been addressed and geologists now have access to non-classified intermediate (10 m–20 m) and high (1 m–10 m) pixel resolution DEMs and multispectral satellite or photographic imagery. The sudden availability of these new datasets has amplified the need within the earth sciences community for straightforward tools that provide both efficient visualization of gigabyte to terabyte datasets and geological mapping within an interactive, 3D visualization environment. The problem of interactive, 3D visualization of large datasets has been previously addressed using multi-resolution and level of detail techniques [8, 7]. We expand that environment to allow users to map on the 3D surface and compare our new application with recently developed alternative approaches based on 3D photogrammetric techniques.

Although active structures typically deform the Earth's surface, this surface is also under constant attack by geomorphic processes that either destroy it via erosion or bury it via deposition. The competition between the rates of surface deformation and destruction results in a characteristic length scale for preserved deformed geomorphic markers, such that markers that are a few thousand to a few million years old will typically have spatial dimensions of a few tens to a few thousands of meters. This spatial range requires geologists to make numerous detailed observations to fully map active structures, and therefore places a fundamental limit on the data resolution needed to study these markers. In particular, multispectral imagery and digital terrain data must have resolutions not coarser than 15 m and 30 m, respectively. However, because the first-order structural systems are typically on the order of 1000 km long, geologists must also make these detailed observations over very large areas. In addition, many of the areas of active continental deformation lie in Africa, Asia and the Middle East, where data are incomplete and/or of variable quality.

Geologists have struggled with the dilemma of making detailed, remote observations over large areas for some time. One compromise is to use low-resolution imagery, and a second is to conduct detailed investigations of small (10 km × 10 km) areas at a few, widely spaced localities. Both methods give a strongly filtered view of the active deformation field. As a result, the geomorphic record of neotectonic deformation, and thus our understanding of how major structural systems evolve at intermediate time scales, remains largely unexplored. In response, we have developed RIMS, a Real-time, Interactive, Mapping System. RIMS allows geologists to visualize, and map in 3D space, structures that are active at intermediate time scales, both in detail but also over thousand-kilometer wide zones of active continental deformation.

2 Related Work

Prior to the development of RIMS, there were two basic methods for analysis of high resolution, multispectral imagery and digital terrain data available to geologists. One option was perspective views of texture data such as imagery draped over a DEM. A large number of widely available tools allow interactive manipulation of such displays [6, 9, 10, 11, 12, 13]. Most of them do not yet appear to make use of multiresolution techniques [14]. More importantly, it is not possible to map directly on the 3D scene. The second approach has been to perform 3D feature extraction using a StereoGraphics Z-Screen and a photogrammetry package [15, 16, 17].

2.1 Terrain Level-of-Detail (LOD) Algorithms

Multi-resolution visualization of large-scale 3D terrain models is an active area of research. [1] have recently presented a technique using triangulated irregular network (TIN) patches as drawing primitives in place of triangles. TIN patches represent terrain highly accurately and are optimized in a preprocessing step for efficient storage and rendering and are batched to the graphics hardware by a view-dependent algorithm. In addition, a texture tile hierarchy is constructed to allow for multi-resolution imagery to be projected onto the terrain geometry. [4] and [8] work with regular gridded data, considerably reducing the preprocessing requirement, and present data management and view-dependent algorithms focusing on the real-time generation of the triangulated mesh representation of the terrain. The former method focuses on attaining the best fidelity in the generated approximation using complex algorithms, whereas the latter method concentrates on the simplicity and ease of implementation of the data management and simplification techniques. All the mentioned techniques feature out-of-core data management enabling them to process the large data sets associated with terrain visualization.

The terrain visualization component of our system is based on the algorithm of [7]. Quadtrees are a well-understood data structuring concept we use to uniformly represent and manage the different components of our system (geometry, texture and mappings).

2.2 Vector Mapping and Display Systems

One of us (Cowgill) has recently developed a method for building digital stereo models (DSMs) from 15 m-pixel resolution ASTER stereoscopic imagery using only the ground control point information provided with the Level 1A data. Individual DSMs comprise 4200×4200 pixel ($\approx 60\,\text{km} \times 60\,\text{km}$) images that can be mapped in 3D using StereoAnalyst (SA) [15] at scales up to 1:20,000 ($\approx 6^2$ screen pixels to 1 image pixel). This system is limited to a plan or bird's eye view and does not permit a user to view the surface along a vertical cross-section, which is a perspective that geologists rely upon heavily for analysis. In addition, the DSMs typically have lateral variations in X and Y parallax, resulting in eye strain after a few hours of analysis with SA. Finally, the lack of an ASTER-specific sensor model and external ground control information restricts the DSM to a single scene, thus only a $60\,\text{km} \times 60\,\text{km}$ area can be mapped at once using SA.

To address the visualization problem of draping 2D vector data over a multi-resolution 3D terrain representation, [18] propose algorithms for rendering geometric lines adapted to surface tessellation. Their method handles sophisticated restricted quadtree triangulations where the representation is not fixed for a given node. Our approach employs fixed regular triangulations per node and allows for the more straightforward method presented in Sect. 4.1.2.

As an alternative to the polyline-as-geometry approach mentioned above, [5] present techniques to render vector data using textures generated on-demand. In hopes of being less sensitive to vector data quantity, we would like to add such a polyline-as-texture approach to our system in the future.

3 Terrain Visualization

To be useful for netotectonic studies, a 3D mapping system must provide interactive textured height field rendering of very large terrain data sets (above $35\,\text{k} \times 35\,\text{k}$ samples) with full roaming and viewpoint manipulation, and at the same time must provide interactive mapping of attributed points, polylines and polygons directly on the 3D terrain model. When zoomed in to large magnifications, the system must display height field and texture at full resolution. The system must be able to import and export georeferenced mapping data from and to standard GIS applications. Ideally, the system should be able to manipulate the viewpoint at any time while placing mapping elements.

3.1 Terrain Representation

Unprocessed terrain data sets are commonly maintained as two-dimensional single or multi-band images. In our case, one gray-scale DEM represents the height measurements for a given longitude/latitude sampling, and other three-channel images contain the corresponding color-information – typically either false-colors generated from the height or spectral-band interpretations, or actual aerial photographs. The first main task of our system is to facilitate visual exploration of such data sets by constructing and rendering appropriate 3D surface representations.

3.1.1 Quadtrees

For rendering purposes, a terrain data set's surface is reconstructed using triangle drawing primitives. When rendered directly, the large and high-resolution terrain data sets required by the application would far exceed current graphics hardware capabilities, prohibiting real-time exploration. To address this problem, we use multi-resolution representations based on a quadtree subdivision. A quadtree's lowest-level nodes correspond to a tiling of a given data set at its native resolution. Higher-level nodes contain successively coarser representations, subsampled by a factor of two between levels. All nodes in the same tree have a fixed size of some power of two in each dimension. By sending only appropriate nodes to the graphics hardware (see Sect. 3.1.2), we can render very large data sets in real time while maintaining sufficient detail for mapping purposes. When treating each tree node as an atomic entity, the quadtree representation enables efficient frame processing, e. g., hierarchical view culling (see Sect. 3.1.3, selecting appropriate detail level (see Sect. 3.1.2), and computing mapping element representations (see Sect. 4.1.2).

In our system, terrain models are represented by a set of correlated quadtrees, the first one containing the terrain's height values, and one or more additional ones containing texture data. These trees are generated from the unprocessed data in a preprocessing step (see Sect. 3.2). Considering that the original height and texture data are tightly correlated, the height and texture trees will present a node-to-node matching if generated with appropriate node sizes. For example, using a 30 m resolution DEM with a 15 m resolution texture draped over it would require a texture quad twice the size of a height quad, e. g., 128 and 64, respectively. This tight coupling prompted us to merge the two trees leaving a single terrain tree that needs to be maintained and processed each frame: each of its nodes contains references to corresponding height and texture data (see Fig. 1 left). We expect this structure to also facilitate future out-of-core management/caching and allow for quick overlaying of compatible preprocessed textures.

3.1.2 Level-of-Detail Calculation

To effectively exploit the multi-resolution terrain representation, we need a means of evaluating the *appropriateness* of a node based on a set of frame conditions. We consider a continuous LOD value characterizing a node as perfectly fitting the conditions if it is zero. Negative values progressively indicate the node's resolution to be too coarse and positive values indicitate it being too fine. Additionally, we choose a *split-threshold* below which nodes will be considered for subdivision and a *merge-threshold* above which merging is suggested (see Sect. 3.1.3). We make the following considerations for our LOD evaluation:

Focus and Context

When mapping, users operate locally on the terrain data, effectively defining the region of interest. We consider each evaluated node's distance from the manipulation cursor as a first LOD value: at the cursor location, the finest detail is desired and the farther away from the focus point, the coarser we are allowed to represent the terrain (see Fig. 1, top-right).

View-Distance Dependency

To explore the data, users are constantly manipulating view parameters that in turn affect the projection of terrain tree nodes into screen space. To account for such view-specific characteristics we consider a node's projected pixel coverage using [8] estimation of perspective projection dependent on the distance between the node and the viewing camera. The node's world-space edge-size is projected and compared to a chosen optimum for a second LOD value: for a given view, the finest detail level discernable on screen (as specified by the optimum) is chosen (see Fig. 1, middle-right).

In the end, the final LOD value is computed by combining the two previous ones: the *focus-LOD* sets the coarsest bound, meaning that although the view would allow for more detail to be displayed on the screen, for a node away from the focus point this does not currently interest the user. On the other hand, the *view-LOD* specifies the finest bound, in that even for the node directly under the focus point we need only render as much detail as will be discernable in the screen projection (see Fig. 1, bottom-right).

3.1.3 Tree Maintenance

Ideally, the *active* tree nodes chosen for a terrain approximation would be those with a LOD value evaluated between the split and merge thresholds. Refining recursively by starting with the root node and subdividing nodes whose LOD values lie below the split-threshold would in fact result in a set of leaf nodes defining a gap-less, overlay-free tiling of the terrain area. However, cracks might appear between neighboring nodes of different resolutions due to hanging triangle vertices (see Fig. 2). To address this problem, we modify the LOD criterion such that direct neighbors in the active set differ at most by one level of resolution. If this property holds for an entire tree, cracks can be removed by simple *stitching* at the edges of affected nodes.

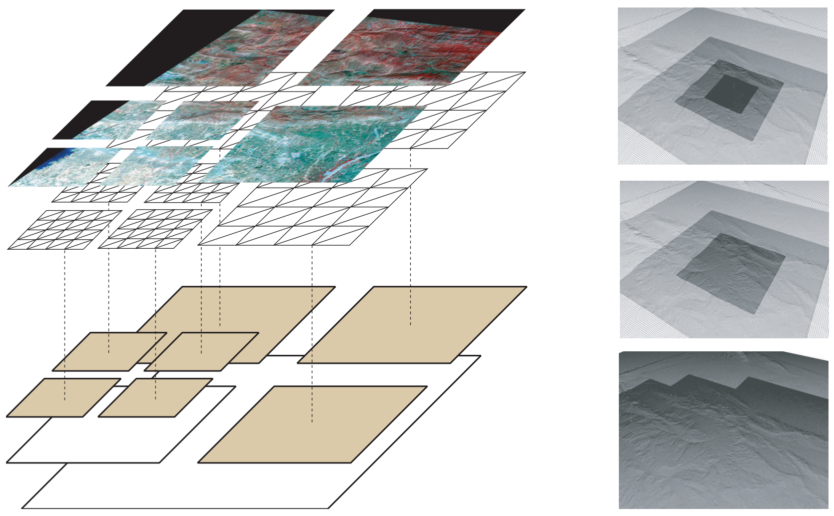

Figure 1 Left: Terrain tree with height and texture data. *Active* tiles belonging to the current approximation are colored, and their geometry and texture data quads are shown. Right: Level-of-detail computation. Top: LOD is calculated based solely on the focus point (note the overly detailed square in the middle). Middle: LOD is determined only using the view parameters. Bottom: Focus and view LODs are combined

Splitting and Merging

Since users exploring terrain roam interactively, view parameters change little from one frame to the next. A high inter-frame coherence can be expected and an approach similar to the one followed by [4] is worthwhile. Instead of generating the appropriate representation by recursive subdivision of the root (with the necessary balancing performed on the fly), the previous representation is *tweaked* to conform to the new frame conditions. Our current implementation dedicates a first traversal pass to tree maintenance using split and merge operations, but the task could easily be left to the care of an independent thread.

The tree maintenace pass initially evaluates the LOD values of all previously active nodes. Their new LOD values are then compared to the thresholds in order to decide whether nodes should be kept as they are, split into their children, or merged with their siblings. Both the split and merge operations assume a valid one-level difference terrain tree and result in a similarly valid one. The split operation does so by recursively forcing the considered node's neighborhood to subdivide appropriately, whereas the merge operation only succeeds if the siblings and neighborhood allow for the parent to become a leaf. This scheme favors showing detail over hiding it. Additionally, before either operation can be completed, the stitching attributes for the inserted node(s) and the neighborhood have to be corrected, i. e., coarser nodes have to be adapted to neighboring finer ones. We use four bit-flags, one for each edge, specifying if the corresponding edge connects to a more detailed one. We describe how these bit-flags relate to actual mesh approximations in Sect. 3.3.

View Frustum Culling

By taking advantage of the quadtree structure, we can further reduce maintenance costs (and later rendering time) using hierarchical view frustum culling. The tree maintenance pass traverses the terrain tree depth-first from the root until it encounters a node that is outside the current view frustum or a leaf node in the current approximation (as determined by the

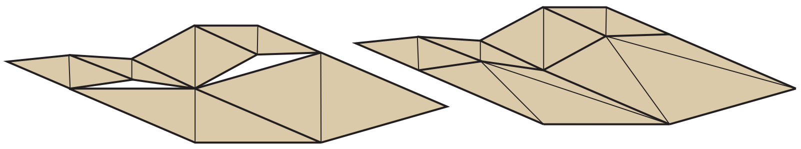

Figure 2 Stitching between nodes. Left: Cracks can appear at the edge of neighboring nodes of different resolutions. Right: Stitching adapts lower-resolution nodes to higher-resolution ones for smooth transitions.

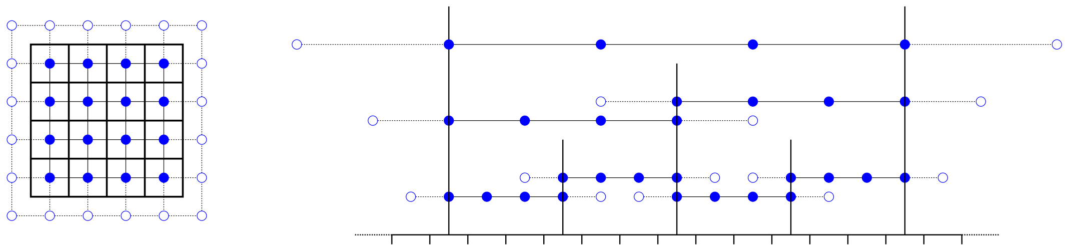

Figure 3 Quadbase preprocessing. Left: Alignment of vertex positions in a height quad (blue dots) and texels in a texture quad (bold squares). Right: Alignment between input height data, height quads in the same level, and height quads between levels. Black tick marks denote pixel borders of input data, blue dots denote vertex positions in height quads, hollow dots denote "ghost vertices" around height quads.

split and merge criteria described above). Computing the visibility of a node is done by intersecting its bounding sphere with the view frustum. This check is very inexpensive, and enables efficient culling of entire subtrees from the maintenance traversal if an upper-level node is invisible. We maintain a visibility flag in each node to forward the results obtained here to the rendering phase (see Sect. 3.3).

3.2 Quadbase Preprocessing

Available terrain data sets usually describe a continuous area at a given resolution, whereas our program requires a multi-resolution hierarchical tiling of that area. We generate the needed tiles off-line with our preprocessing tool and store them in a binary *quadbase* file. The preprocessor first constructs a skeletal quadtree with the property that its leaf nodes tile the input data set at its native resolution, the root node entirely covers the input data set's domain, and only those nodes intersecting the domain are retained. The skeletal tree is then traversed in a bottom-up, breadth-first fashion. At each level, each node crops out the data associated with it and appends it to the quadbase file. After all nodes in a level are processed, the input data is resampled to the resolution appropriate for the next higher level. To associate the image data with mesh geometry we place vertices at the centers of pixels (see Fig. 3, left). Therefore, care must be taken to duplicate quad edge pixels where vertices are shared for rendering. Moreover, quads produced to store height information should additionally store border pixels to facilitate generation of vertex normals later on (see Fig. 3, right).

Descriptive information for both the quadtree, e. g., quad resolution and number of quads, and the contained data, e. g., upper-left corner longitude/latitude and data resolution, is stored in an additional quadbase header file.

3.3 Rendering

Mesh Representation

Whereas the preprocessed texture quads can be used directly as sources for texture objects, the height quads have to be converted into triangulated patches of 3D vertices. The vertex positions and texture coordinates are generated by creating a planar regular grid where (x, y)-points are elevated using the appropriate pixel value of the height quad. Vertex texture coordinates are calculated by linearly mapping (x, y)-coordinates into the associated texture quad's texture rectangle, which is identical for each texture node. The only considerable computation comes from running a filter on a height pixel's neighborhood to obtain vertex normals for rendering. After being computed, the position, texture coordinates and normals are stored in memory compactly as an interleaved vertex array. We have chosen to omit this step from the preprocessing to keep the input data as general and independent of internal geometric representation as possible. This approach also reduces I/O volume, making us less dependent on slow reads from disk.

We employ a simple caching scheme for node geometry and texture data, to circumvent having to wait for disk I/O when a previously active node becomes activated again. This caching scheme can be enhanced for full data size-independent out-of-core rendering for very large terrain data and limited main memory.

Rendering pass

The image corresponding to the current terrain representation is produced in a second pass through the terrain tree. Similar to tree maintenance, a separate thread could be assigned this task, refreshing at the graphics hardware's rate instead of the I/O-bound update thread's rate. A depth-first traversal from the root finds the active nodes of the current approximation and renders them, exploiting the hierarchical view culling maintained in the visibility bit-flag (see Sect. 3.1.3). Additionally, we could use the quadtree structure to always draw the nodes in front-to-back order and take advantage of the graphics hardware's depth buffer culling.

In Sect. 3.1.3, we mention the need for neighboring rendered quads to align without cracks, even when they do not represent the same level of detail. This affects the triangulations that have to be generated: with one level of difference maximally possible between neighbors, we can identify fifteen different stitching cases. For each case, we pre-compute a static index array defining appropriate triangle strips over the vertex grid. To render a node, its vertex data can then efficiently be sent to the graphics hardware with the appropriate index array for the node's stitching flag computed by the tree maintenance traversal (see Sect. 3.1.3).

4 Mapping

The real-time rendering provided by RIMS constitutes a highly valuable tool for terrain data exploration. However, textured 3D representations are already available in common commercial software (albeit not using multi-resolution approaches yet) and many advanced techniques have been published. More important for our purposes is the use of the 3D terrain model to directly and efficiently specify and edit georeferenced mapping elements. The following section presents our program's mapping capabilities.

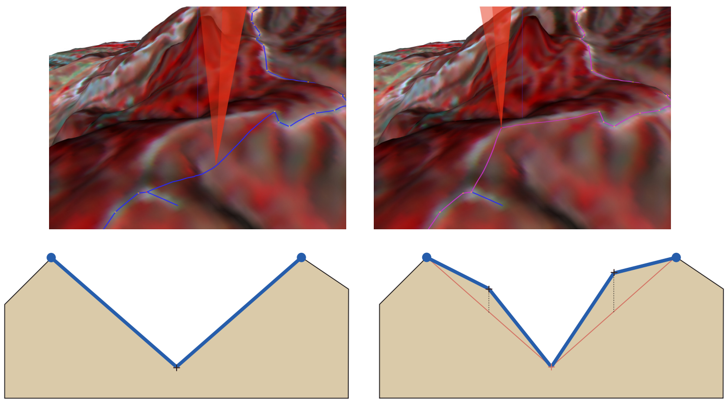

Figure 4 Top: Refining a polyline by inserting a new control point. Bottom: 3D polyline representation. Left: Line strips connecting the segment control points follow the terrain topology. Right: Mesh representation has changed showing more detail, thus hiding the old line strips (red). A new line strip has to be generated connecting the same control points.

Specifying 2.5D Mappings

Typically, mapping data is two-dimensional, e. g., a polyline would be specified as a list of (longitude, latitude) control points. Our mapping tools conceptually operate on a 2D plane by keeping this representation and dynamically assigning appropriate height values to all control points. This approach allows for mappings to be defined independently of the current 3D terrain approximation which, in our case, is constantly changing. Interfacing with common GIS packages can then also be realized easily: our system supports the ASCII ARC/INFO interchange file format for imports and exports.

Most commonly, geologists highlight features using a connected sequences of line segments, i. e., polylines. Our system supports mapping with this primitive: controlling a cursor bound to the terrain surface as a spatial reference, users can perform various actions such as creating, selecting, moving and deleting control points (see Fig. 4, top).

4.1 Polyline Rendering

To display polylines we take a line-as-geometry approach similar to [18]. Combining such an approach with the multi-resolution 3D terrain representation requires “lifting” polylines to the 3D terrain model appropriatedly to avoid clipping with the terrain geometry (see Fig. 4, bottom). In the following, we describe processing the polyline approximation in detail.

4.1.1 General Handling

Geometric lines, our display primitives, can only accurately follow flat surfaces, like those defined by the triangles of the 3D terrain representation. Thus, each 2D polyline segment – specified by a pair of 2D control points – has to be represented by a sequence of 3D line segments, one for each triangle intersected by the 2D polyline segment. Re-computing the appropriate 3D vertices for each frame would dramatically reduce the amount of segments that can be visualized interactively. To address this limitation, we exploit the locality of polyline manipulations (moving an inner control point, for example, modifies at most two segments) and the strong frame-to-frame coherence (triangulations will only change for few quads in each frame) by storing 3D representations for all polylines, and *tweaking* previously valid representations when polyline segments are edited, or the terrain approximation changes.

Table 1 Data set sizes (in pixels for DEM and texture), preprocessing times (in seconds) and frame rates (in frames per second) for the three test data sets.

Data Set	DEM Size	Tex Size	Build	Min. fps	Avg. fps	Max. fps
Aksai	1850 × 900	3700 × 1800	6 s	41.2	141.6	285.7
Mosul	2558 × 2447	5115 × 4901	22 s	60.6	130.0	400.0
Puget Sound	8193 × 8193	16384 × 16384	750 s	30.1	94.2	285.7

A polyline is represented as a list of subsegments, such that each subsegment is contained in a single currently active quadtree node. When a polyline is created or manipulated, the sequence of subsegments is computed by clipping the 2D polyline against the domains of all active nodes it intersects. Each active node also stores a list of subsegments associates with it, such that when a node splits or is merged with its neighbors, the affected polyline subsegments can be determined efficiently, and replaced with new ones appropriate for the changed set of active nodes.

4.1.2 Subsegment Computation

The dominating computational cost of visualizing a polyline lies in the generation of line strips for each of its subsegments, i. e., for each polyline part contained in an active quad of the current terrain approximation; thus, a fast technique is required to maintain high frame rates. In our case, this is facilitated by the regular triangulations within each quad. Moreover, computing the subsegment vertices is effectively only to a 2D problem: since all vertices of the 3D terrain approximation are extruded from the (x, y)-plane along the same direction, we “flatten” them back onto the plane containing the 2D polylines. Intersection points can then be computed and subsequently extruded appropriately. Thus, a very simple algorithm similar to those used to rasterize lines to a regular pixel grid can be used with few modifications.

5 Results

To evaluate RIMS’ performance, we simulated mapping usage on three test data sets of different sizes. The data set sizes (as DEM size and texture size) and the preprocessing times necessary to create the hierarchical quadbases from the input data sets are given in Table 1, as well as the minimum, average and maximum frame rates achieved during mapping.

To evaluate the utility of RIMS, we conducted two comparison tests between the RIMS and StereoAnalyst (SA) [15] mapping methods. The first test (see Fig.5 and 6) compares the maximum level of geological detail that can be extracted from the data to identify the mapping system with the highest sensitivity to detail. Geologists seek the most sensitive system because it allows them to extract the largest amount of information and thus develop the most sophisticated geological analysis. For this test, a user spent as much time as needed to extract the maximum number of features over the same area. The second test (see Fig. 7) compares the number and quality of geologic observations that can be collected in the same finite period of time to identify the most efficient mapping system. Geologists prefer highly efficient systems that allow them to process their data as quickly as possible. For this test, a user spent two hours mapping the same area. In both tests, the study areas were mapped first with SA, then with RIMS. This approach was admittedly biased, because the users had the benefit of already having mapped the scene once at the start of their RIMS sessions. However, both users are significantly more familiar with the SA navigation/mapping environment;

thus, their lack of familiarity with the RIMS controls likely offsets any advantage their prior SA mapping provided.

The tests indicated that RIMS provides a number of user benefits, including reduced eye strain, faster zoom and pan speeds, and slight advantages in the navigation. More importantly, the tests also revealed five key differences that make RIMS more useful for geological applications. Relative to SA, RIMS provided greater 1) understanding of the mapped structural geometry and thus pattern of active deformation; 2) confidence in feature identification and location; 3) numbers of mapped features (i. e., a larger number of mapping elements); 4) mapping accuracy (i. e., a larger number of vertices per mapping element); and 5) ability to locate and identify small features. Specific examples of each difference are provided in the following sections, highlighting the utility of RIMS.

1. The most important difference revealed by the tests is that RIMS allowed both users to obtain a more sophisticated understanding of the structural geometry of their areas. For example, in Fig. 5, arrow A' on the right-hand side of the figure points to a structure that was obvious in the RIMS environment. The lack of a structure at arrow A on the left-hand side of the figure indicates that the user was not able to see and interpret this feature using SA. A RIMS screen shot (see Fig. 6, right half) clearly shows the structure mapped at A', and also demonstrates that it appears as an uninterpretable bump in a plan-view stereo pair that replicates the view from SA (see Fig. 6, left half). Likewise, additional structures were discovered at B' and C' using RIMS while the corresponding points B and C indicate that the user missed these features when using SA. In summary, the plan (bird's eye) view and grayscale imagery of SA made it difficult to identify the topographic and textural variations that indicated the existence of these subtle features.
2. RIMS provided both users with higher confidence in their vector mapping, as indicated by the type of lines selected to represent mapped features. Geologists express their confidence in their ability to accurately locate a mapped feature by using solid, dashed, or dotted lines (in order of decreasing confidence). Fig. 7 shows that the RIMS project contains 20 boundaries mapped using solid lines, 2 using dashed, and 2 with dotted. In contrast, the SA project has only 2 boundaries defined with solid lines, 21 with dashed lines, and 1 with a dotted line.
3. Both users were able to identify a larger number of features using RIMS than SA. The RIMS output shown in Fig. 5 has 289 mapped features whereas only 172 features were extracted using SA. Likewise, Fig. 7 indicates that 14 major structures were defined using RIMS, in contrast to 8 structures on the SA map.
4. RIMS allows users to more accurately locate features and then map them using more vertices per feature because it does not demand constant manual parallax adjustments. Because the polylines have more vertices in the RIMS outputs, they better track short wavelength variations in the feature geometry and thus more accurately follow subtle changes in the boundaries between geologic units. In contrast, the maps generated from SA show a prevalence of long straight line segments. Differences in detail are especially evident in Fig. 7 at comparison points A–A', B–B' and C–C' in SA and RIMS, respectively.
5. Finally, RIMS is more effective for locating small geologic features. For example, a series of river terraces located at point D' in the RIMS output were not located at point A using SA (see Fig. 7). Likewise, points E–E' indicate a small outcrop that was not seen in SA at E but that was mappable using RIMS at E'. Although these features are small, their identification has important implications regarding the geometry of active deformation in the mapped area.

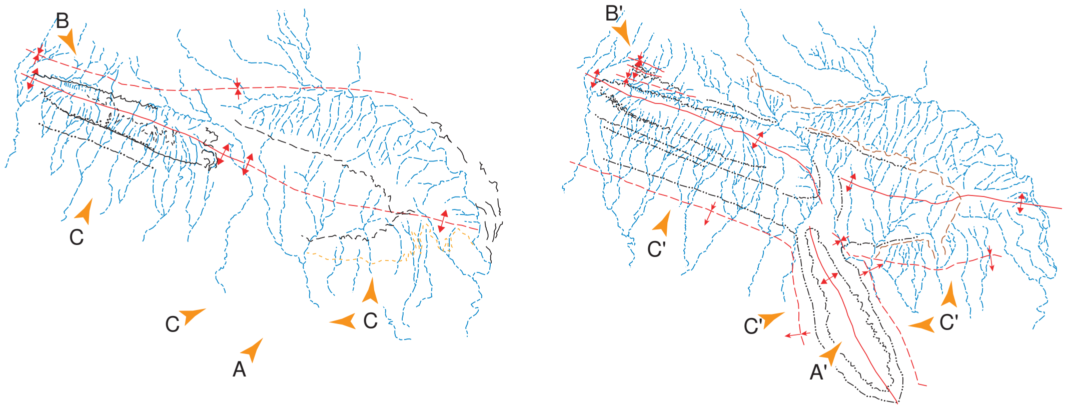

Figure 5 Results of sensitivity test. Gold arrows highlight points where the maps differ significantly, as discussed in the text. Red lines are fold hingelines and are solid where confidently located and dashed where their position is less clear. Blue lines denote drainages. Broken black lines indicate contacts between two different geologic units, dotted black lines are marker beds. Dashed yellow line denotes the edge of a geomorphic surface. Brown lines indicate drainage divides. Left: Map generated using StereoAnalyst. Right: Map generated using RIMS.

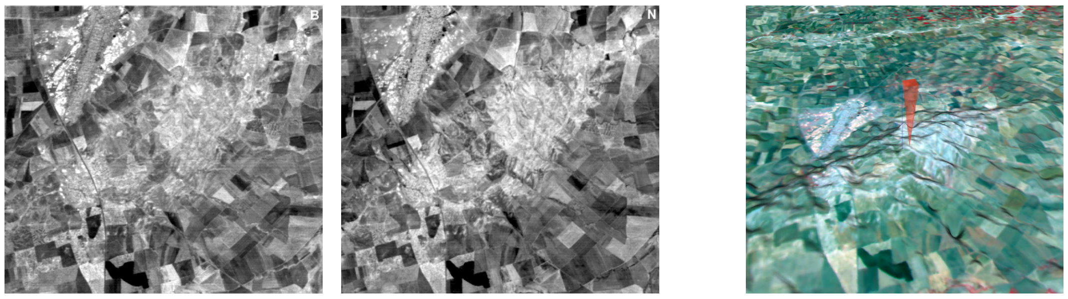

Figure 6 Subtle ridge appearing at location A–A' in Fig. 5. Left and center: Cross-eye stereo pair reconstructing the plan view provided by StereoAnalyst. Right: Screen shot from RIMS.

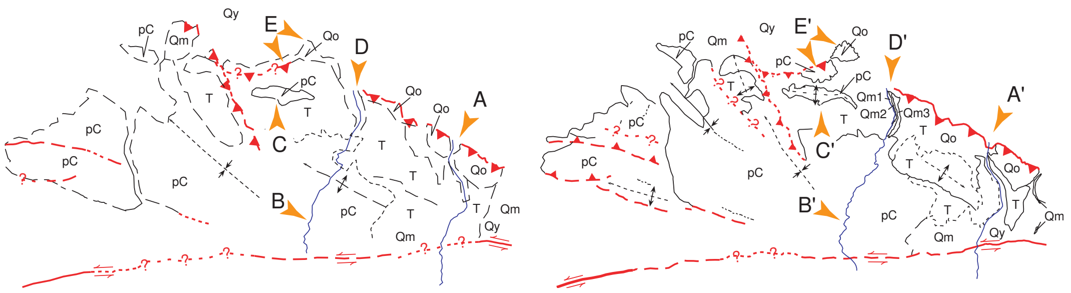

Figure 7 Results of efficiency test. Decorated red lines are various types of active faults. Black lines represent folds and contacts between two different geologic units. Red and black lines are solid where features are confidently located and dashed, dotted, or querried where position is increasingly less clear. Solid blue lines are drainages. Text labels (pC, T, Qo, Qm#, Qy) denote units of different apparent ages. Left: Map generated using StereoAnalyst. Right: Map generated using RIMS.

6 Conclusions and Future Work

In summary, while the tests described above show that the maps generated using both utilities capture many of the same major geologic features, it is clear that RIMS is both a more sensitive and a more efficient mapping utility, and thus greatly advances geologists' ability to remotely map patterns of active defomation in fine detail while also spanning continental collision zones that are thousands of kilometers wide and often inaccessible for field study. The advantages of RIMS over the previously used system are mostly due to RIMS' interactive visualization of large textured 3D terrain models, and its ability to map directly onto the 3D terrain in real-time.

Our future efforts will focus on moving the terrain maintenance out-of-core to allow for more scalability. We are also looking into on-demand textures to support a higher quantity of mappings with a more varied appearance (as seen in the results figures produced with ArcMap). In addition, mapping capabilities are to be extended providing geologist with more tools and help so as to more efficiently extract interesting features from the data sets.

Acknowledgments

This work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, through the National Partnership for Advanced Computational Infrastructure (NPACI), and a large Information Technology Research (ITR) grant. This work was partially supported by the W. M. Keck Center for Active Visualization in the Earth Sciences (Keck CAVES) and NASA grant EOS/03–0663–0306. We have benefited from conversations with Magali Billen, Louise Kellogg, and Nickolas Raterman. We thank the members of the Visualization and Graphics Research Group at the Center for Image Processing and Integrated Computing (IDAV) at the University of California, Davis.

References

1 P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. BDAM – batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum*, 22(3), 2003.

2 E. Cowgill, J Ramón Arrowsmith, A. Yin, X.-F. Wang, and Z. Chen. The Akato Tagh bend along the Altyn Tagh fault, NW Tibet 2. Active deformation and the importance of transpression and strain-hardening within the Altyn Tagh system. *Geological Society of America Bulletin*, 2004.

3 E. Cowgill, A. Yin, J Ramón Arrowsmith, Xiao-Feng Wang, and Shuanghong Zhang. The Akato Tagh bend along the Altyn Tagh fault, NW Tibet 1. Smoothing by vertical-axis rotation and the effect of topographic stresses on bend-flanking faults. *Geological Society of America Bulletin*, 2004.

4 Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. In *Proceedings of the 8th conference on Visualization '97*, pages 81–88. IEEE Computer Society Press, 1997.

5 Oliver Kersting and Jürgen Döllner. Interactive 3D visualization of vector data in GIS. In *Proceedings of the tenth ACM international symposium on Advances in geographic information systems*, pages 107–112. ACM Press, 2002.

6 J. M. Lees. Geotouch: Software for three- and four-dimensional GIS in the Earth sciences. *Computers and Geosciences*, 26:751–761, 2000.

7 Peter Lindstrom, David Koller, Larry F. Hodges, William Ribarsky, Nickolas Faust, and Gregory Turner. Level-of-detail management for real-time rendering of phototextured terrain. Technical Report 6, 1995.

8 Peter Lindstrom and Valerio Pascucci. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):239–254, 2002.

9 GRASS GIS. http://grass.baylor.edu.

10 ArcScene utility in 3D Analyst extension of ArcGIS.
http://www.esri.com/software/arcgis/arcgisxtensions/3danalyst/.

11 3D SurfaceView utility in ENVI.
http://www.rsinc.com/envi/.

12 VirtualGIS module in ERDAS IMAGINE.
http://gis.leica-geosystems.com/Products/Imagine/.

13 FLY! http://www.pcigeomatics.com/product_ind/fly.html.

14 ArcGlobe, to be released in Spring 2004.
http://www.esri.com/news/arcnews/summer03articles/introducing-arcglobe.html.

15 Stereo Analyst for ArcGIS.
http://gis.leica-geosystems.com/Products/StereoAnalyst/.

16 OrthoEngine add-on for Geomatica.
http://www.pcigeomatics.com/product_ind/add_on_oe.html.

17 SOCET SET. http://www.vitec.com/products/socetset/.

18 Zachary Wartell, Eunjung Kang, Tony Wasilewski, William Ribarsky, and Nickolas Faust. Rendering vector data over global, multi-resolution 3D terrain. In *Proceedings of the symposium on Data visualisation 2003*, pages 213–222. Eurographics Association, 2003.