Terrain Synthesis from Digital Elevation Models

Howard Zhou, Jie Sun, Greg Turk, *IEEE Member* and James M. Rehg, *IEEE Member*

***Abstract*— In this paper we present an example-based system for terrain synthesis. In our approach, patches from sample terrain (represented by a height field) are used to generate new terrain. The synthesis is guided by a user-sketched feature map that specifies where terrain features occur in the resulting synthetic terrain. Our system emphasizes large-scale curvilinear features (ridges and valleys) because such features are the dominant visual elements in most terrain. Both the example height field and user's sketch map are analyzed using a technique from the field of geomorphology. The system finds patches from the example data that match the features found in the user's sketch. Patches are joined together using graph cuts and Poisson editing. The order in which patches are placed in the synthesized terrain is determined by breadth-first traversal of a feature tree and this generates improved results over standard raster-scan placement orders. Our technique supports user-controlled terrain synthesis in a wide variety of styles, based upon the visual richness of real-world terrain data.**

***Index Terms*— Terrain synthesis, Digital Elevation Models, terrain analysis, texture synthesis.**

Fig. 1. Synthesized terrain using DEM of Flathead National Forest Mountain Range (top image) and DEM of the Grand Canyon (bottom image).

I. Introduction

THERE are numerous applications that make use of synthetic terrain, and very often the terrain is the dominant visual element in the scene. Such applications include landscape design, flight simulators, emergency response training, battleground simulations, feature film special effects and computer games. Over the years, graphics researchers have made considerable progress towards developing efficient methods for generating synthetic terrain. Previous terrain synthesis work has focused on using fractal models and physical erosion models to create realistic-appearing terrain.

With the rapid growth of computing power and development in terrain visualization techniques, the demand for more realistic terrain has increased considerably. In addition, users of terrain modelling applications want more control over the creation of new terrain. However, current terrain synthesis methods have several limitations. First, these methods provide users with little or no control over the placement of desired terrain features. Second, using the control parameters in these methods, it is difficult to generate terrain with a desired style, such as a terrain with the geological features of the Grand Canyon.

Fig. 2(a) and 2(b) show examples of fractal and erosion terrain produced by some popular commercial software, displayed as intensity-coded elevation maps. The styles of these terrains are quite unlike the natural terrain illustrated in Fig. 2(c) and 2(d).

We present a novel example-based terrain synthesis method which addresses the need for intuitive user control over both terrain feature placement and terrain style. Our method draws upon the techniques of patch matching and patch placement from example-based texture synthesis. In our approach, the user supplies a sketched terrain feature map (called the *sketch map*) and real terrain data (called the *example height field*) which contain desired terrain styles. Example height fields are in the form of Digital Elevation Models (DEMs), which are available online from the U.S. Geological Survey. Our system then automatically generates a new height field that preserves the visual style of the real terrain data and meets the feature constraints of the sketch map. Synthetic results from our approach are shown in Fig. 1.

The sketch map provides the user with an easy and intuitive way to control the synthesis process. Each map specifies the locations of important terrain features, such as the bifurcation point at the center of Fig. 10(a). Notice that these sketches are quite coarse. In fact, the width of the brush and the pixel intensities are of little importance, as long as they follow the simple principle that darker indicates lower elevation and brighter indicates higher elevation. Our goal in this work is the generation of visually-compelling terrain. We do not address the separate issue of whether a synthesized terrain is geologically accurate.

The starting point for our algorithm is the identification of important terrain features in both the sketch map and the example height field. Our system concentrates on large curvilinear features such as rivers, valleys and mountain ridges, since these are usually the most important visual elements in large-scale terrain (as illustrated in Fig. 2(c) and 2(d)). Because the underlying terrain structures in the sketch map and the ones

All authors are with the GVU Center, Georgia Institute of Technology, 85 5th Street NW, Atlanta, GA 30308. E-mail: {howardz, sun, turk, rehg}@cc.gatech.edu

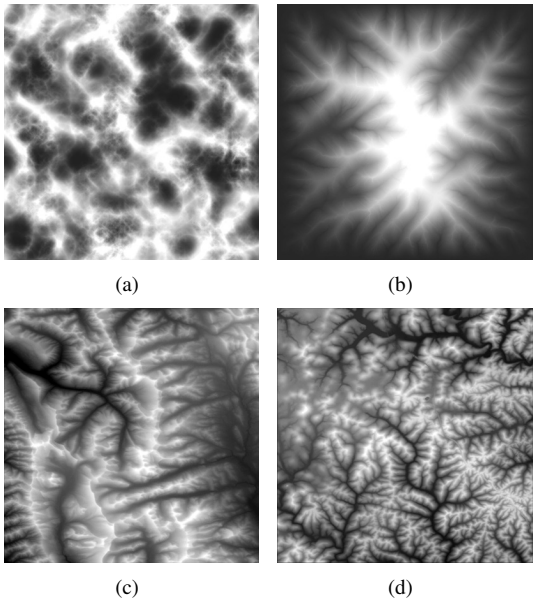


Fig. 2. Examples of terrain generated by current methods. (a) Fractal Terrain (Ridged Multi Perlin) generated by Terragen at 1025×1025 resolution. (b) Erosion Terrain generated by Bryce 5 at 1024×1024 resolution. In contrast are two real elevation maps: (c) 1/3 arc second(10m) DEM of Flathead national forest mountain range, MT and (d) 1/3 arc second(10m) DEM of Mount Vernon, KY. (DEM courtesy of the U.S. Geological Survey)

in the example height field are often very different, matches over large spatial scales are unlikely to be correct. However, we observe that at small scales, common terrain features such as a bifurcation or a straight section of valley can be found in both maps. Hence, our algorithm breaks the sketch map into small patch regions and searches through the example height field for structural feature matches.

Extraction of structural information from the sketch map is straightforward due to its simplicity. On the other hand, the extraction of high-level structural information such as ridges and valleys from a real terrain height field can be difficult. Here we draw upon work in the field of geomorphology. Specifically, we use the Profile recognition and Polygon breaking Algorithm (*PPA*) developed by Chang et al. [1] to extract structural information such as ridge and valley axes from a height field.

Once the important structural terrain features have been extracted, our algorithm proceeds in the following stages. First, curvilinear terrain features are used as constraints for the matching and alignment of patch regions in the sketch map and the example height field. Second, these features are used as constraints for matching along the overlapping boundary between neighboring patch regions. A patch is selected based on a weighted combination of scores which assess the goodness-of-fit between a patch and its neighbors and the amount of deformation of the patch that is required. The order of patch placement is determined by a breadth-first traversal of a feature tree which is constructed from the user sketch. Each patch is placed into the output map using a combination of graph cut seam finding and Poisson seam removal to minimize visual discontinuities.

This paper makes the following contributions:

- Feature-based approach to matching and placement of large curvilinear terrain features, which makes it possible to efficiently search large terrain databases and preserve important visual elements in the synthesis process.
- Tree-ordered patch placement algorithm, as an alternative to standard raster-scan ordering, which results in a more faithful reproduction of terrain structure in matching the user input.
- Method for sketch-based specification of synthesized terrain features, giving the user intuitive control over the synthesis result.
- The ability to synthesize transitions between different terrain types and incorporate multiple DEMs into a single synthesized output.

We have drawn significant inspiration for our work from previous texture synthesis methods, which we review in the following section.

II. Related Work

There are two main approaches to generating synthetic terrain: fractal landscape modelling and physical erosion simulation. Fractal landscape modelling dates back to the pioneering work of Mandelbrot [2]. Since then, a variety of stochastic subdivision techniques have been introduced. Fournier et al. [3] introduced the random midpoint displacement technique to create fractal surfaces. Voss [4] added successive random displacement to fractional Brownian surfaces. Miller [5] proposed a square-square subdivision scheme for generating fractal terrain and a parallel processing algorithm for rendering height fields. Lewis [6] proposed generalized stochastic subdivision. Szeliski and Terzopoulos [7] addressed the problem of user control by combining deterministic splines and stochastic fractals into constrained fractals. Recent fractal-based approaches are reviewed in [8] and [9].

Physical erosion simulation is an alternative approach to synthesizing terrain details based on models of landscape formulation and stream erosion from the geomorphology community. It is often used as a refinement step after a rough height field is generated. Kelley et al. [10] first introduced a method to approximate natural terrain by simulating the erosion of stream networks. Later, Musgrave et al. [11] then combined the fractal modelling and erosion simulation approaches into a single framework. Recent physical erosion techniques, exemplified by [12], [13], [14], [15], and [16], have focused on improving both the physical modelling aspect and computational efficiency. With appropriately-tuned parameters, these techniques can generate realistic-appearing terrain.

Both fractal and physical erosion techniques add terrain details through procedural refinement, which often involves non-intuitive parameter tuning. Recently, Brosz et al. [17] attempt to extract high resolution terrain details from existing DEM data and apply it to lower resolution terrain through multi-resolution analysis. In practice, their method requires both the source and target terrain to be fairly detailed and does not grant the user freedom to create arbitrary terrain.

To provide the user with more intuitive control over the synthesized terrain, image-based alternatives were proposed by

Lewis [18] and Perlin and Velho [19]. In these works, terrain was viewed as a type of texture and user control was provided through direct manipulation of the texture. The result was then interpreted as a height field to create a variety of terrain types. However, because it is difficult for a user to draw a realistic natural height field by hand, these methods typically suffer from a lack of realistic detail.

Commercial Software We are not aware of any existing commercial software for terrain synthesis that employs an approach that is similar to ours. We briefly review four major commercial systems, Mojoworld, Terragen, World Machine, and Bryce, which are representative of the current state-of-the-art. Mojoworld and Terragen both have fractal synthesis engines which generate terrain procedurally. The resulting models are very compact because they are generated on-the-fly. World Machine, on the other hand, offers geological erosion on top of its procedural shape and noise generator to synthesize realistic-appealing terrain. In all these systems, control over the synthesis result is obtained by changing global parameters in the generation process. It is, however, difficult to set these parameters so as to generate distinctive, realistic terrain types. Moreover, these systems do not support the user-specified placement of major terrain features. Bryce, in addition to having both fractal and erosion synthesis engine, also accepts as input a direct specification of the height field by the user using a painting approach (essentially the method of [18] with the addition of fractal noise). Fig. 2(c) and 2(d) are example terrains generated by such commercial systems. None of these products, however, make it possible to synthesize terrain in a specific style, such as style of the Grand Canyon.

Texture Synthesis Image-based texture synthesis is the process of creating an arbitrarily large patch of texture by drawing pixels from a given example image. During the past few decades, a steady improvement in the quality of synthesized textures, both 2D and volumetric, has been achieved through an evolution from pixel-based methods [20] to non-parametric neighborhood-based methods [21], [22], [23]. The most recent patch-based techniques, exemplified by [24], [25], [26], [27], [28], [29], [30], and [31], have two common stages: 1) search in a sample texture for neighborhoods most similar to a context region; 2) merge a patch or a pixel with the (partially) synthesized output texture. Dynamic programming [24] and graph cuts [26] have been used to optimize the patch merging stage. We employ a related search-and-merge strategy that addresses the unique characteristics of terrain data.

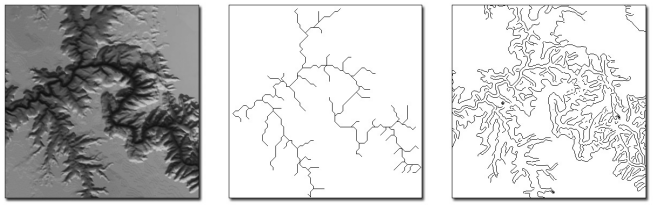
Recently, Zhang et al. [32] introduced feature-based warping and blending techniques to synthesize progressively-variant texture on arbitrary surfaces. In their work the feature texton masks were manually extracted. Wu and Yu [27] use edges extracted from the input texture as high-level features to guide patch-based texture synthesis. We build on this earlier work in two ways. First, we employ curvilinear features to support user-sketching of desired terrain features and efficient search for matching patches in large terrain datasets. Second, we introduce a feature analysis technique which can reliably extract global terrain characteristics such as ridges and valleys from large DEMs in a wide range of styles. We demonstrate that standard edge-finding methods are inappropriate for this task.

The image analogies framework introduced by Hertzmann et al. [25] can be used to synthesize terrain images through a texture-by-numbers approach. This work does not directly synthesize novel terrain height fields. Its application to height field synthesis is hampered by the difficulty of guaranteeing that the local neighborhood matching approach would preserve extended structures such as ridges and valleys that characterize terrain style. The GPU-based texture synthesis method presented in [30] includes drag-and-drop features and synthesis magnification. These techniques were used to relocate mountains on a terrain height map. In contrast to this work, our focus is to automatically generate a complete terrain image in a particular style by matching extended terrain features with a desired user sketch. We therefore support fully-automatic feature extraction and matching.

While our example-based approach to terrain synthesis is inspired by the recent success of patch-based methods, terrain synthesis is not simply texture synthesis on height fields. The terrain generation problem can be distinguished from conventional texture synthesis in three main ways: First, a wide variety of terrain types can be characterized by a combination of global features (such as ridges and valleys) which can be reliably extracted from input terrain maps. In contrast, no such easily identifiable global features exist for general image textures. Second, terrain synthesis must be globally controllable in order to be useful for a wide range of applications, and large terrain data sets (often a gigapixel or more) must be searched in order to meet the users' constraints. This stands in contrast to the canonical texture synthesis problem of "growing" a small input patch of texture into a larger output image with the same local intensity structure. We demonstrate that feature-based terrain matching can address both of these concerns. Third, many recent cut-and-merge techniques for texture synthesis exploit the fact that image textures contain many natural boundaries that provide good seams along which to merge texture elements. In contrast, there are no natural seams in terrain data. Moreover, any mismatch between the height fields in two adjacent patches is immediately visible. To remove height differences by blurring results in highly-visible artifacts. We show how to combine recent graph-cut and Poisson seam removal techniques to address the problem of merging terrain patches.

III. Feature Extraction

In this section, we describe our method for extracting terrain features from the example height field and from the user's sketch map. We use the term *terrain features* to mean large-scale long curvilinear features such as a river, a valley or a mountain ridge. Such terrain features characterize the overall layout of the terrain. Fig. 3(a) shows an example height field, consisting of a portion of the Grand Canyon, which is depicted as a shaded relief map. This example illustrates both the existence of large scale curvilinear structures (the main river bed) as well as a rich array of secondary structures (the side canyons and other features). Our goal is to match the primary structures to a desired sketch shape while preserving the rich detail that is present in this data.



(a) (b) (c)

Fig. 3. Comparison of feature extraction methods. (a) Grand Canyon height field displayed as a shaded relief map. (b) Features extracted by the Canny edge detector. (c) Features extracted by PPA.

It may seem that terrain features could be identified through edge detection. Closer scrutiny, however, reveals several problems in applying edge detection to complex terrain data. First, terrain features are characterized by the local extrema of the height field, while edge detection methods are based on locally-maximal derivatives of the image. As a consequence, the application of edge detection to terrain data results in spurious features due to local height variations, as shown in the upper right corner of Fig. 3(b).[1] In addition, terrain data has characteristic branching structures which are not always handled correctly by standard edge detection algorithms (e.g. lower right corner of Fig. 3(b)).

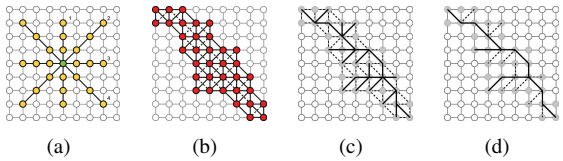
We draw upon work in the geomorphology literature to identify large-scale terrain features. In particular, we have adapted the Profile recognition and Polygon breaking Algorithm (PPA), developed by Chang et al. [1], to our task. We briefly summarize the method in this section, and refer the reader to their paper for additional details.

PPA can extract either ridges or valleys. Here we will only describe ridge finding, with the necessary modification to finding valleys being understood. First, each grid point (pixel in the height field) is visited to determine if it is a candidate for being on a ridge. The grid points in eight outward paths are examined (the yellow points in Fig. 4(a)), and the current point is marked as a candidate if the height dips below the central point by more than a threshold[2] along any of these paths. All such candidates are then connected by a segment to all other adjacent candidates (the red circles in Fig. 4(b)). When one segment crosses another, the lower elevation segment is cancelled (shown as dotted segments in Fig. 4(b)). The polygons are then broken into dendritic line patterns by repeatedly eliminating the least important segment (the remaining segment with the lowest height, shown as dotted segments in Fig. 4(c)) for each closed polygon. This process terminates when there are no more closed polygons. After the polygon breaking process, shorter branches are eliminated (dotted segments in the Fig. 4(d)).

The output of PPA for a sample terrain is shown in Fig. 3(c). Notice that the valley axes coincide with human perception of the important curvilinear features in the height field. Our system uses PPA to identify features both in the example height field and in the user's sketch map. The gray-scale values in the sketch map are treated as elevation values. The output of PPA is a collection of line segments that are connected and that form long chains along ridges or valleys. These chains of segments are the basis for feature matching between the user's sketch and regions in the given height field. Once the chains of segments have been identified, they are analyzed to form two classes of features: isolated features and curvilinear (path) features. *Isolated features* are branch points and end points. *Curvilinear features* are long chains of segments that connect isolated features. The feature matching process described next make use of these two feature categories.

[1]Searching for edges at multiple scales cannot easily solve this problem, as perceptually important terrain structures are not guaranteed to persist over scale.

[2]We typically set the threshold to one half of the elevation range of the input height field.



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

Fig. 4. Chang's Profile recognition and Polygon breaking Algorithm (PPA) (a) Profile recognition (b) Target Connection (c) Polygon breaking (d) Branch reduction

A key property of the PPA algorithm is that the extracted features form a tree (a forest in general) since all closed polygons are broken in the analysis process. This property allows us to use tree traversal to order the placement of patches during synthesis, as described next.

IV. Feature-based Patch Matching and Placement

Our synthesis process creates a new height field by extracting patches from the example height field and placing them in an output height field in a manner that is dictated by the user's sketch. Typical patch sizes are 80×80 pixels (this is determined by the spatial scale of the example terrain data and the detail of the result desired by the user). Patch selection and placement is performed in two stages: *feature patch matching and placement* and *non-feature patch placement*.

The first stage, feature matching and placement, locates both isolated features: branch points and end points, and non-isolated features: curvilinear features (*paths*) in the sketch map. See Fig. 5 for examples of each of these feature types. Patches are found that match these features, and these patches are placed in the output height field.

If we treat the isolated features extracted by the PPA algorithm as nodes and curvilinear features (paths) as edges of a graph, then this graph is guaranteed to be acyclic. Our algorithm follows a breath-first-search order to match and align the patch regions. It first picks an isolated feature (usually a branch if it is present) as the root, it then traverses down through the graph one edge at a time until every edge reachable from the root is covered. This process is illustrated in Figs. 6(c)–6(e). Note that this is a departure from traditional texture synthesis methods that follow a rigid patch placement order (e.g. left-to-right and top-down).

Finally, all the as-yet unfilled regions in the output height field are filled using regions from the input that do not contain any strong features. The final result is depicted in Fig. 6(f).

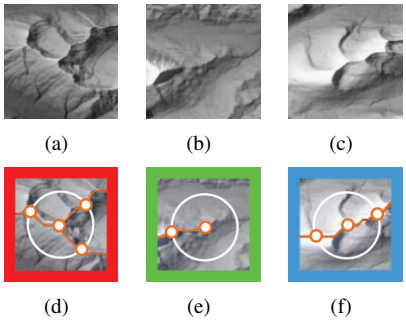


Fig. 5. Examples of terrain patches for each feature type: (a),(b),(c) branch point, end point, and path patch respectively. (d),(e),(f) corresponding patches after feature extraction.

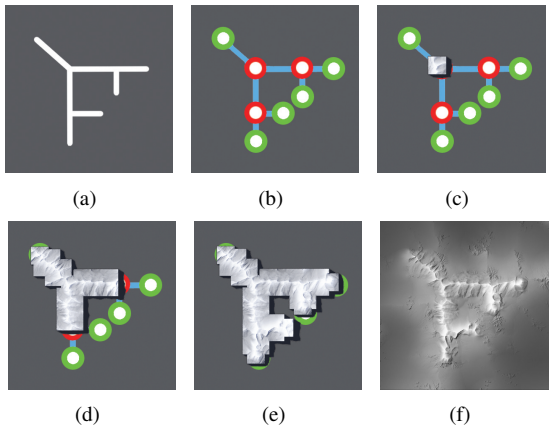


Fig. 6. Illustration of patch placement order. (a) Sample sketch map. (b) Tree structure returned by PPA analysis. Branch point features and end point features are connected by curvilinear path features. (c) The root patch is placed first. (d) Breadth-first traversal guides the placement of additional patches. (e) Once tree traversal is complete, begin placing non-feature patches. (f) Final result.

In both stages, several candidate patches are considered for placement, and the patch that is selected for use is the one that gives the least cost in terms of its match with the user's sketch and its overlap with already placed patches. (Patch placement invokes a graph cut algorithm to select which pixels should be deposited in the output height field, and this will be described in Sec. V-A). In the following sections, we describe these matching and placement operations in more detail.

A. Feature Patch Matching and Placement

As described above, isolated features in the user's sketch are either branch points or end points in the segment graph. Fig. 6(b) contains three branch points and five end points. The system analyzes the user's sketch map and identifies all such isolated features. One by one, each such isolated feature is examined and a list of candidate matches is formed from isolated features in the example height field. We will consider branch points and end points in turn.

Branch Points For branch point matching, the degree (or valence) of the branch points must match and the angles of the outgoing segment chains must be similar. For determining the quality of angle matching, only d possible alignments have to be considered for a degree d branch point. For instance, as shown in Fig 5(d), the degree 3 branch patch would have 3 alignments $(1,2,3) \rightarrow (1,2,3),(2,3,1),(3,1,2)$ (The mirror image of this patch is treated as a new patch and its alignment is a separate process.) Because there are typically not many such branch points in a given example height field, testing against all candidate matches is extremely fast.

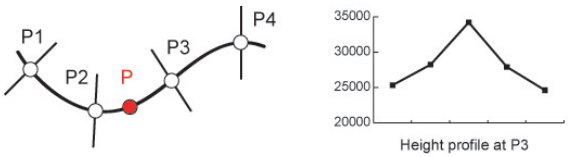
To perform warping of a candidate patch to fit the user's sketch, first a set of control points $\{P_i\}$ must be identified for the patch. These control points consist of the location of the branch point itself, plus those places where each outgoing path intersects a circle that is inscribed in the patch (see Fig. 5(d)). For instance, a patch with a three-way branch will have a total of four control points. Corresponding control points $\{P'_i\}$ are also defined for the patch from the sketch map. Now we desire a continuous coordinate transformation that maps the control points $\{P_i\}$ to $\{P'_i\}$ and that gives the minimal amount of distortion. We use a thin-plate spline (TPS) interpolant [33] for this purpose because it works well with few constraints and it introduces minimal distortion as measured by the integral bending norm. We use two separate TPS functions to form a coordinate transformation that maps any position in the original height field to its interpolated location in the warped height field. The best k branch point patches with the lowest deformation energy are the candidates for further matching.

End Points Endpoint matching is straightforward because all end points have similar curvilinear features: a relatively short segment chain going out of the end point, which can be aligned easily. Thus we select all end point patches (Fig. 5(e)) as candidates for further matching without applying a warping in this case.

Path Features All of our curvilinear features are chains of line segments (from PPA) that stretch between two isolated features. Our system finds matching patches along such a curvilinear feature while traversing these chains of line segments in the user's sketch. The system travels along these chains in steps that are one-half a patch in size, laying down a patch with each step. As with branching patches, the candidate patches along a curvilinear feature are deformed to better fit the user's sketch. There are always three control points P_0, P_1, P_2 in the candidate patch (Fig. 5(f)), and three corresponding points P'_0, P'_1, P'_2 in the user's sketch that determine the warp. The outer control points P_1 and P_2 are located where the path crosses the inscribed circle of the patch. The central point P_0 is the midpoint in the chain within a patch, which is analogous to the central control point in a branch patch.

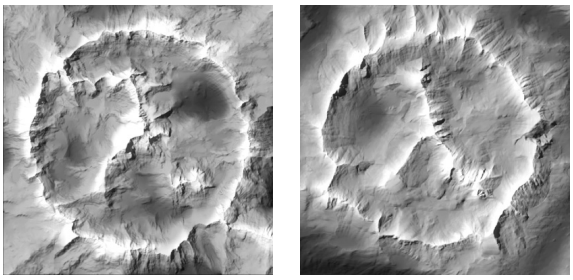
All of the candidate patches are ranked according to a combination of matching criteria. The best candidate patch is merged in the output height field. In our current implementation, we use the following matching criteria:

- c_d: Deformation energy from the TPS warping (We refer the readers to [33]) the candidate patch terrain structure matches the sketch map constraints. Although TPS warping can warp the patch into desired configuration in most cases (Except for degenerate cases such as when three control points are collinear), large deformation results in noticeable distortion and is penalized here.
- c_g: Graph cut score (See Appendix A). The graph cut seam cost is an indication of how well the candidate patch



(a) (b)

Fig. 7. (a) Height profiles perpendicular to the path are stored at uniformly sampled points along the path. The height profile at the point P which joins two paths is linearly interpolated from the profiles stored at $P2$ and $P3$. (b) An example of the 5-point height profile that is stored at $P3$.



(a) (b)

Fig. 8. (a) Synthesis result using raster-scan patch placement order (b) Synthesis result using tree traversal

matches the already merged patches in the overlap regions (Graph cut automatically handles the case where one patch overlaps with multiple patches and the overlapping can be disjoint). A higher graph cut score means the seam is more noticeable and harder to remove by Poisson editing.

- c_f: Feature dissimilarity cost. The dissimilarity cost measures how closely features along the joining paths are matched[3]. We uniformly sample the curvilinear path and store a coarse height profile perpendicular to the path (our current implementation uses 7-point profile) at each sampled point (Fig. 7). When the path is joining the other path at point P, the height profile at P is linearly interpolated from the profiles stored at its two nearest neighbors, $P2$ and $P3$. Height profiles from joining paths are compared using sum of squared differences.
- c_i's : Other user specified constraint(s) such as height constraint or path smoothness constraint.

The total cost is then a linear combination of all of the matching costs $c = \alpha_d c_d + \alpha_g c_g + \alpha_f c_f + \sum_{i=1}^{n} \alpha_i c_i$. Note that the α's serve not only as weights but also as normalization coefficients for the costs. In our current system, we set $\alpha_d = 1000$, $\alpha_g = 1$, and $\alpha_f = 3$ for most test cases. Changing these coefficients changes the emphasis on the matching criteria and results in different synthesis results. After curvilinear feature matching and breadth-first-search order placement, the output height field looks like Fig. 6(e).

[3]The graph cut cost gives an overall measure of the overlap quality, but it weights every pixel in the overlap region equally. It is difficult to integrate the feature dissimilarity measure into the graph cut algorithm; therefore, a separate feature dissimilarity cost term is introduced into our system to place emphasis on features.

Fig. 8 shows a comparison between raster-scan and tree-ordered patch placement using the sketch map of Fig. 10(a) and the example height map of Fig. 10(b). The inner structure of the λ symbol is garbled in the raster scan result (Fig. 8(a)). This is because the row based scan locks onto the boundary of the circle immediately, creating constraints on future matching that prevent good matches in the interior. As Fig. 8(b) shows, both the λ and the circle can be reproduced if the most difficult element (the center branch point) is matched first.

B. Non-Feature Patch Placement

Once the isolated features and curvilinear features have been placed, the empty areas in the output height field are those places without strong features. A feasible way to fill these areas is to copy patches that match the pixels that have already been placed. To do this, our system "grows out" from the already filled-in areas. Specifically, Square patch positions in coarsely-spaced increments (e.g. every 100 pixels horizontally and vertically) are placed down in descending order according to the area of the overlapping region with the already synthesized height field.

The system looks for a high-quality match between such a patch and the already synthesized height field based on the sum of squared difference on the overlapping region and selects k candidates. It then finds the best match according to the combination of the SSD score and the graph cut cost in merging. The best patch is placed, and the system continues traversing the output height field looking for overlaps. The process terminates when all pixels in the output map have been filled.

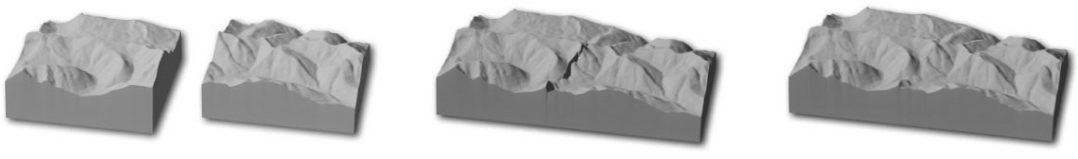
The SSD-based search to find the patch that best matches the already synthesized output height field can be accelerated using Fast Fourier Transforms [34], [35]. Using the FFT-based approach, the matching cost can be computed in $O(n\log(n))$ time, where n is the number of pixels in the source height field. This is in comparison to $O(n^2)$ time for a naive SSD implementation.

V. Patch Merging

Our system combines two techniques to assure smooth transitions between patches that are placed in the output height field. The first of these is the graph cut technique [26], [36], which finds good seams between already-placed pixels and the pixels from a region that is in the process of being placed. The second procedure solves a discrete Poisson equation [37] to create more gentle transitions between the existing pixel elevations and the pixels from a newly placed patch.

A. Graphcut Optimal Seam Finder

The graph cut algorithm finds a seam in the overlapping region between patches that determines which pixels will be kept in the final image. This is accomplished by solving a max-flow/min-cut graph problem that minimizes the cost of mismatched elevations across the cut. Edges in the graph represent connections between adjacent pixels, and they are given weights based on elevation differences. For some patches, we



(a) (b) (c)

Fig. 9. Illustration of patch placement and seam removal. (a) Matching patches are identified via graphcut. (b) Patch placement results in a seam. (c) Poisson seam remover yields final output.

TABLE I
INPUT DEM LOCATIONS, RESOLUTION AND SIZES.

Origin	Resolution	Size
Mount Jackson, CO	10m	1620×1620
The Grand Canyon	30m	4080×2760
Mount Vernon, KY	10m	1200×1200
Flathead Range,MT	10m	4097×4097
Puget Sound, WA	10m	4097×4097

want to insist that particular groups of pixels from a given patch should be included, such as the central pixels in a branch feature patch. This is accomplished by setting some of the edge weights infinitely high.

As mentioned earlier, the cost of the final seam from the graph cut algorithm is used by the system to select among candidate patches for representing a given feature. Details on the use of the graph cut algorithm for seam finding can be found in [26].

B. Poisson Seam Remover

After the system has placed a new patch into the output map using graph cut, a discontinuity may still be visible along the seam (Fig. 9(b)). We further improve elevation matching across a seam by adjusting the heights according to artificially set gradient values at the seams. We do this by solving a Poisson equation, similar to the manner in which Pérez et al. perform pixel matching across seams in a color image [37].

Our elevation adjustment stage first translates the elevation values in the overlap region (typically one third of the patch size) into gradient values. Then the gradient values across the seam are artificially set to zero. Finally, a Poisson equation is solved to find the best-fit set of elevations to these adjusted gradient values. The result of this process is a new set of elevations that is much more smooth at the seam (Fig. 9(c)). Our system performs such elevation adjustments locally, invoking the method one time per placed patch. Performing elevation adjustment locally avoids the challenge of solving huge matrix equations. For more details, we refer the reader to Appendix B.

VI. TERRAIN SYNTHESIS RESULTS

We used DEMs from U.S. Geological Survey terrain data for the results shown in Figures 1, 10, 11, 12, 13, and 15. The origin, resolution, and size of the data are listed in Table I.

Our example height fields range from 1200×1200 to 4097×4097 samples at a height resolution of 16 bits. These large terrain maps pose significant search problems for a traditional texture synthesis approach. At our patch size of 80×80 pixels, there are approximately 16 million possible matches in a given 4000×4000 map. However, as a result of our feature-based analysis, we can filter this total set down to approximately 600 match evaluations for a given candidate position to be filled, which dramatically reduces the computational cost. Our method can synthesize a representative terrain in approximately 5 to 6 minutes on an Intel P4 2.0 Ghz processor with 2GB memory. All of the results are rendered using Planetside's Terragen terrain rendering system with procedural textures (determined by height and slope) overlayed on top of the terrain geometry.

Fig. 10 and Fig. 11 illustrate the variation in output of that can be obtained from the same sketch using our terrain synthesis approach. Fig. 10(d) shows the synthesis result (1000×1000) for the Mount Jackson terrain, and the sketch map for this example is shown in Fig. 10(a). Although this result was created from many patches, both the interior λ and the outer circle are formed by unbroken mountain ridges. Also note the characteristic curtain-like folds in the sides of the mountains. Fig. 10(h) shows the synthesis results (2000×2000) using the Grand Canyon terrain. The sketch map in Fig. 10(e) used for this example is the inverse of Fig. 10(a) so that valleys are selected instead of mountains. Here the λ and the circle are formed from joined pieces of the Colorado River, yet the seam locations cannot be detected. Although guided by the user's sketch, this terrain retains the rich water-carved features of the original data. Fig. 11(d) shows the synthesis result (1000×1000) for the Mount Vernon terrain and Fig. 11(h) shows the synthesis results (2000×2000) using the DEM from Flathead National Forest mountain region. Fig. 12(d) shows the synthesis result (1000×800) for the Mount Vernon terrain with a user sketched Chinese character for "water". These results illustrate the ability of our system to work with a wide variety of terrain types with very different characteristics, while maintaining the salient features of the input sketch map. Fig. 15 provides close-up views of the Grand Canyon and Flathead range synthesis results. Fig. 13(a) shows a rendering of the Grand Canyon height map. We extract the canyon feature from this map using PPA to obtain a sketch map. By synthesizing the sketch in the style of Puget Sound DEM, we obtain a "mountain" version (4000×2000) of the Grand Canyon, illustrated in Fig. 13(b).

Synthesis with Multiple Terrain Styles A larger example, illustrated in Fig. 14, shows the generation of a 3D map of Middle Earth using several different terrain styles. A simple sketch map (Fig. 14(b)) was created by the user from an artist rendering of the map of Middle Earth (Fig. 14(a)). The user

assigned different terrain styles to each part of the sketch map (Table II). In this example, the user first placed down Mount St.Helen at the location of Mount Doom in Mordor (a localized feature). Our system then synthesized the rest of the terrain (3470×2996) automatically and merge it seamlessly with Mount Doom. Note that the user sketch includes both a variety of mountain ranges and a river valley (the Dead Marshes, near the North-West corner of Mordor). The coast, lakes, and rivers were created using a simple mask, which was extracted from the original map. The synthesized height map (Fig. 14(c)) was rendered using Planetside's Terragen terrain rendering system with both procedural and masked textures to give it an artistic feel. This example illustrates the ability of our system to combine multiple terrain styles into one synthesized terrain and to blend seamlessly among them.

Please view the movie (DivX encoded) from the TVCG web site that accompanies this paper. Additional results may be found at `http://www.cc.gatech.edu/~howardz/terrain`.

TABLE II
MIDDLE EARTH REGIONS

Middle Earth region	Elevation map
Mordor	Puget Sound, WA
Gondor, Rhun, Ered	Rocky Mountain, CO
Moria	Mount Jackson, CO
Dead Marshes	Mount Vernon, KY
Jron	Flathead Range, MT

VII. CONCLUSION

We have demonstrated that example-based texture synthesis methods can be successfully adapted to the domain of terrain synthesis. The result is a new level of visual realism in user-controllable synthesized terrain. Our approach leverages the fact that useful terrain features can be extracted from height fields using analysis techniques from the geomorphology community.

We have introduced a tree-ordered patch placement algorithm which is based on a breadth-first traversal of a feature tree. Our results demonstrate that this placement method is superior to standard raster-scan placement orders.

We have demonstrated the ability to synthesize terrain in widely-differing styles while retaining control over the positioning of major terrain features. Our system is based on an intuitive sketch-based interface for specifying desired terrain characteristics. We believe ours is the first system to make full use of an example-based approach in the domain of terrain synthesis.

Our method suffers from the same limitations as all example-based methods. In particular, the quality of the final synthesis depends upon the richness of the available terrain data. If the terrain features desired by the user cannot be found in the example height field, then it will not be possible to produce the desired result. For instance, our method will perform poorly if the terrain data is from a desert region where few significant curvilinear features are present. The method can also perform poorly when the curvilinear feature pattern is extremely complicated. An additional issue is the need to specify the patch size, which depends in turn upon the resolution and scale of the example terrain.

We plan to extend our current method in several ways. First it would be interesting to give the user greater control over the synthesized terrain by incorporating additional constraints into the sketch map. For example, we plan to provide the ability to specify a desired elevation at a specific position. This could be accomplished by constraining the Poisson solver in conjunction with the matcher. We are also investigating the enforcement of C^1 continuity in addition to our current enforcement of C^0 continuity. We intend to explore the joint synthesis of elevation and texture maps for rendering and the interactive control of terrain synthesis.

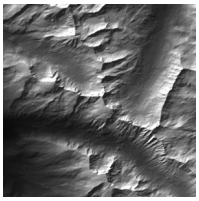
ACKNOWLEDGMENT

The authors would like to thank the many people at the GVU Center of Georgia Tech who have helped and encouraged them. This work was supported by National Science Foundation grant ITR-0205507 and NSF Graduate Research Fellowship (to Howard Zhou). The authors would also like to thank the reviewers whose excellent comments lead to the current paper.

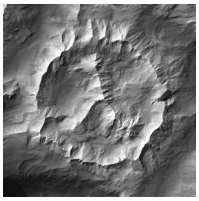
REFERENCES

[1] Y.-C. Chang, G.-S. Song, and S.-K. Hsu, "Automatic extraction of ridge and valley axes using the profile recognition and polygon-breaking algorithm," in *Computer & Geosciences*, vol. 24, no. 1, 1998, pp. 83–93.
[2] B. B. Mandelbrot, *The Fractal Geometry of Nature*. New York: WH Freeman and Co., 1982.
[3] A. Fournier, D. Fussel, and L. Carpenter, "Computer rendering of stochastic models," in *Communications of the ACM*, vol. 25, no. 6, 1982, pp. 371–384.
[4] R. F. Voss, "Random fractal forgeries," in *Fundamental Algorithems for Computer Graphics*, R. A. Earnshaw, Ed., Berlin, 1985.
[5] G. S. P. Miller, "The definition and rendering of terrain maps," *SIGGRAPH 1986*, vol. 20, no. 4, pp. 39–48, 1986.
[6] J. P. Lewis, "Generalized stochastic subdivision," *ACM Trans. Graphics*, vol. 6, no. 3, pp. 167–190, 1987.
[7] R. Szeliski and D. Terzopoulos, "From splines to fractals," in *SIGGRAPH 1989*, 1989, pp. 51–60.
[8] D. S. Ebert, F. K. Musgrave, D. Peachy, K. Perlin, and S. Worley, *Texturing and modeling: a procedural approach*. Morgan Kaufmann, 2002.
[9] C. Dachsbacher, "Interactive terrain rendering: Towards realism with procedural models and graphics hardware," http://www.opus.ub.uni-erlangen.de/opus/volltexte/2006/354/, 2006.
[10] A. D. Kelley, M. C. Malin, and G. M. Nielson, "Terrain simulation using a model of stream erosion," in *SIGGRAPH 1988*, 1988, pp. 263–268.
[11] F. K. Musgrave, C. E. Kolb, and R. S. Mace, "The synthesis and rendering of eroded fractal terrains," *SIGGRAPH 1989*, vol. 23, no. 3, pp. 41–50, 1989.
[12] P. Roudier and B. P. M. Perrin, "Landscapes synthesis achieved through erosion and deposition process simulation," *Computer Graphics Forum*, vol. 12, no. 3, p. 375, August 1993.
[13] N. Chiba, K. Muraoka, and K. Fujita, "An erosion model based on velocity fields for the visual simulation of mountain scenery." *Journal of Visualization and Computer Animation*, vol. 9, no. 4, pp. 185–194, 1998.
[14] K. Nagashima, "Computer generation of eroded valley and mountain terrains," *The Visual Computer*, vol. 13, no. 9-10, pp. 456–464, 1997.
[15] B. Benes and R. Forsbach, "Layered data representation for visual simulation of terrain erosion," in *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics*. Washington, DC, USA: IEEE Computer Society, 2001, p. 80.
[16] B. Neidhold, M. Wacker, and O. Deussen, "Interactive physically based fluid and erosion simulation," *Eurographics Workshop on Natural Phenomena*, 2005.

(a) User sketch



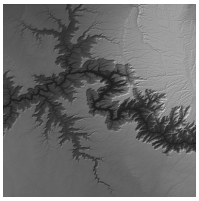
(b) Mount Jackson



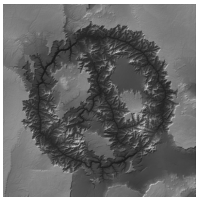
(c) Synthesis Result

(d) Rendered terrain synthesized from an elevation map of Mount Jackson, CO.

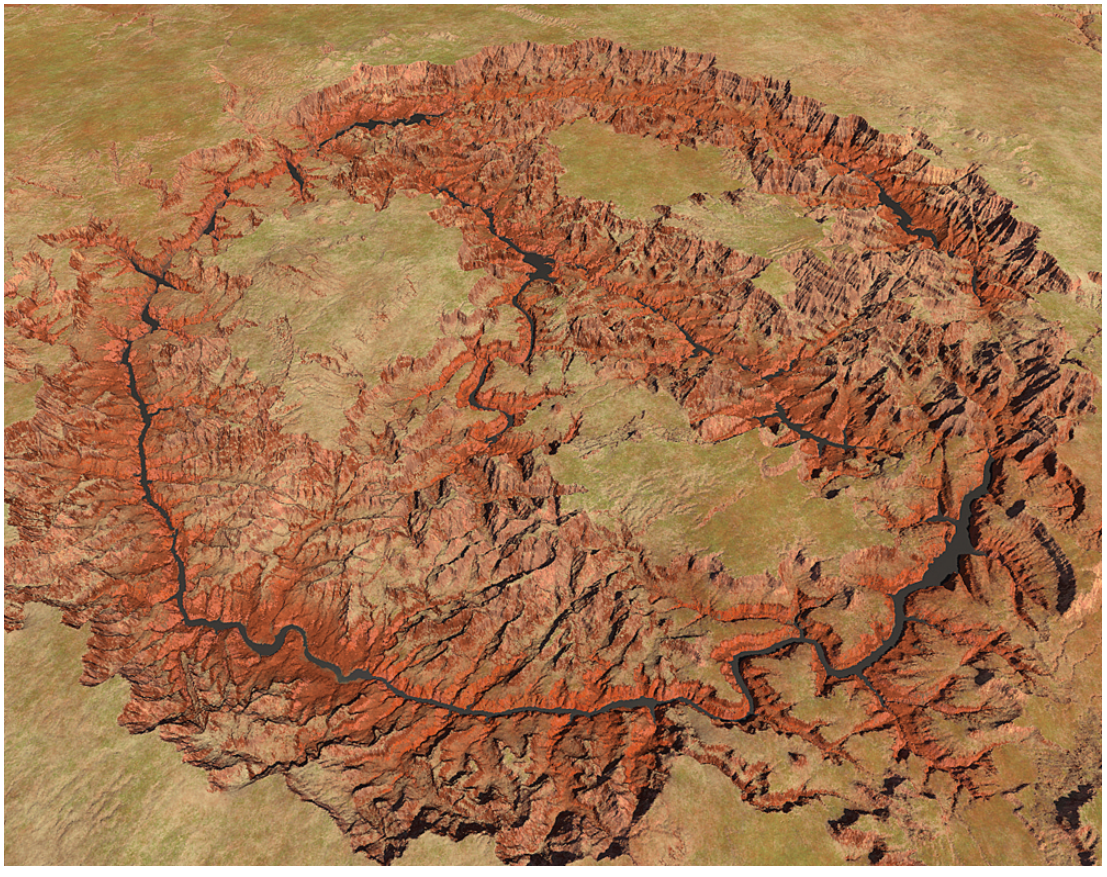
(e) User sketch



(f) Grand Canyon

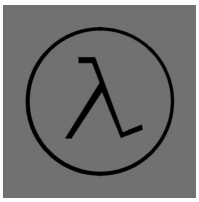


(g) Synthesis Result

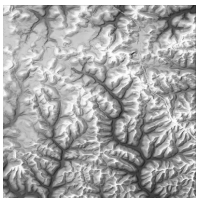


(h) Rendered terrain synthesized from an elevation map of the Grand Canyon.

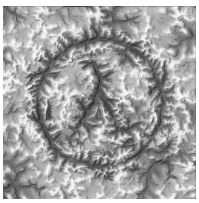
Fig. 10. Multiple terrain synthesis results with sketched half-life symbol.



(a) User sketch



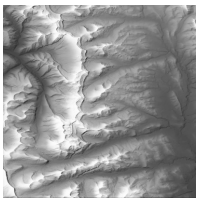
(b) Mount Vernon



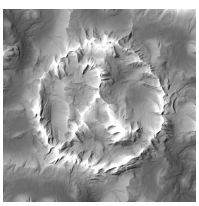
(c) Synthesis Result

(d) Rendered terrain synthesized from an elevation map of Mount Vernon, KY.

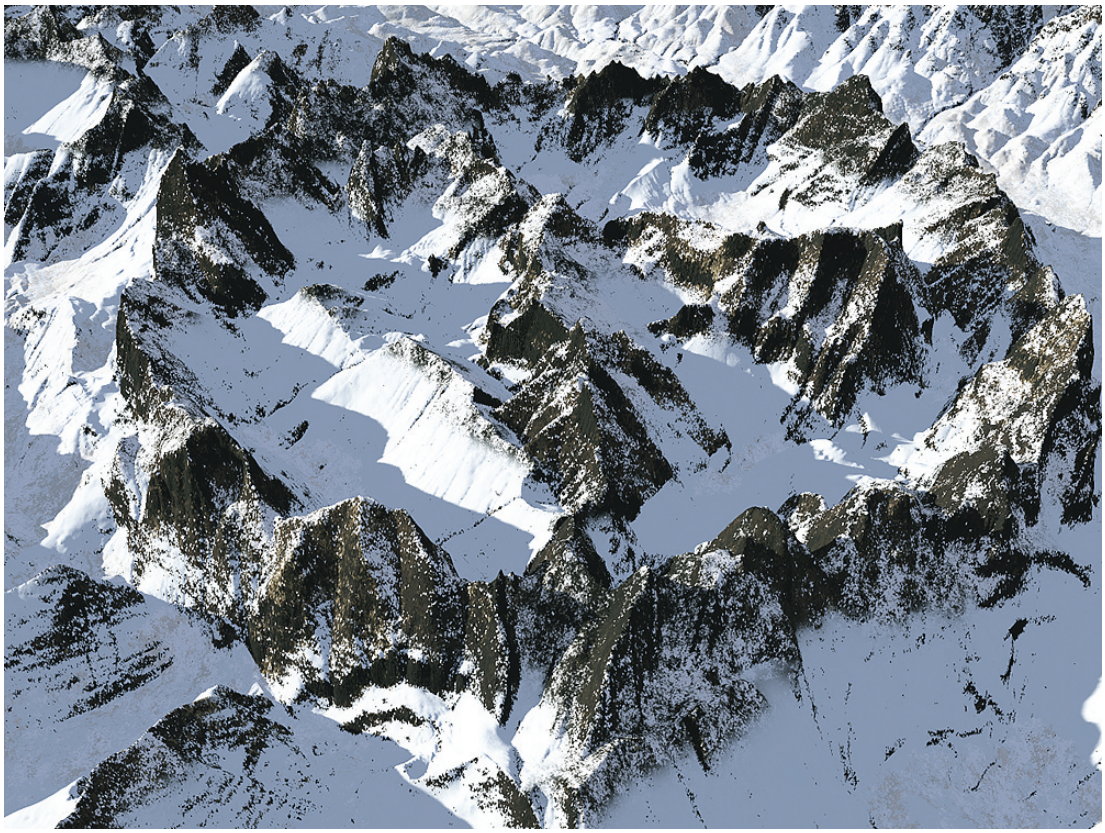
(e) User sketch



(f) Flathead Range



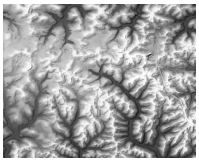
(g) Synthesis Result



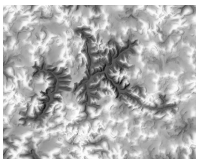
(h) Rendered terrain synthesized from an elevation map of Flathead National Forest mountain range, MT.

Fig. 11. Multiple terrain synthesis results with sketched half-life symbol.

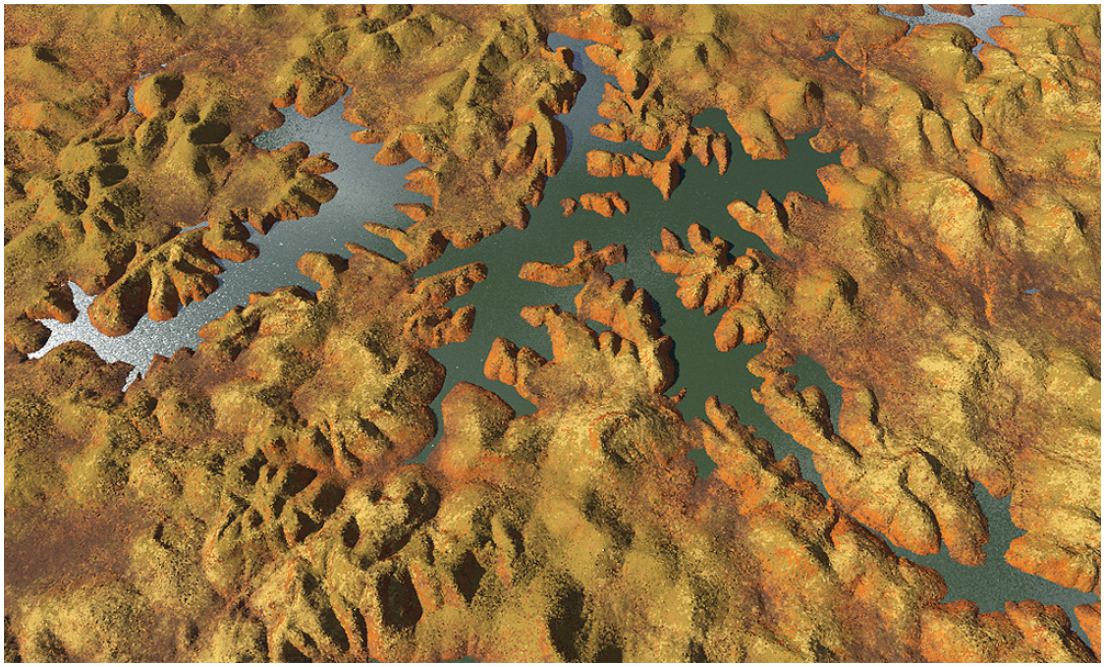
(a) User sketch



(b) Mount Vernon

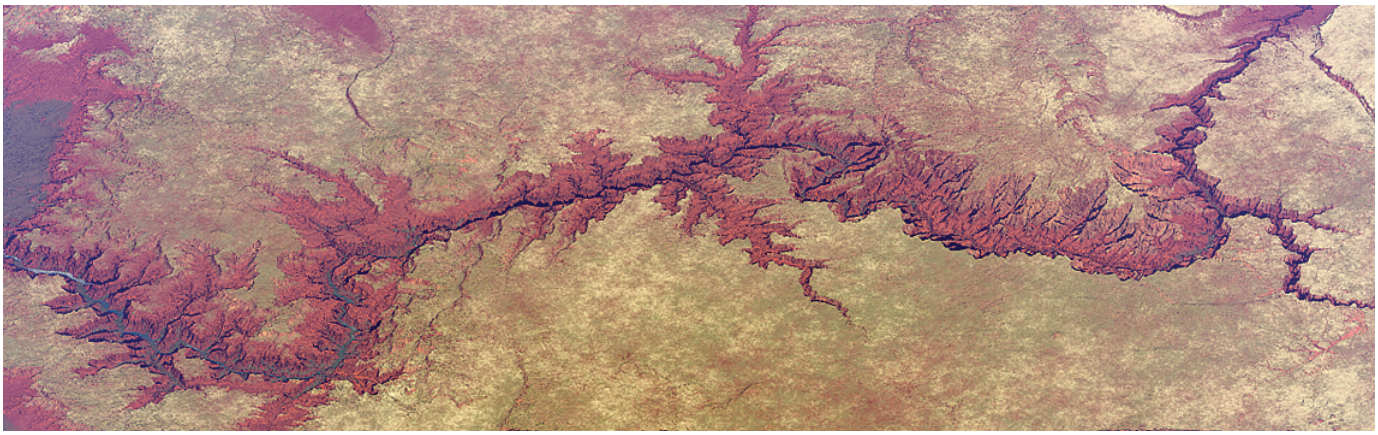


(c) Synthesis Result



(d) Rendered terrain synthesized from an elevation map of Mount Vernon, KY.

Fig. 12. Terrain synthesis result with sketched Chinese character for “water”.



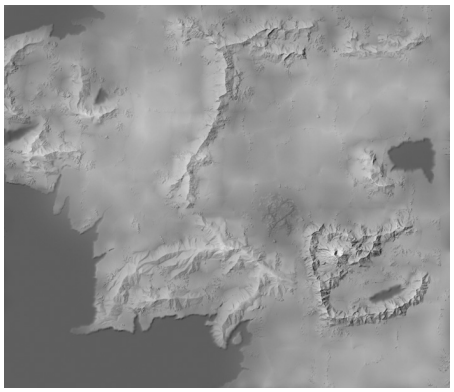
(a)

(b)

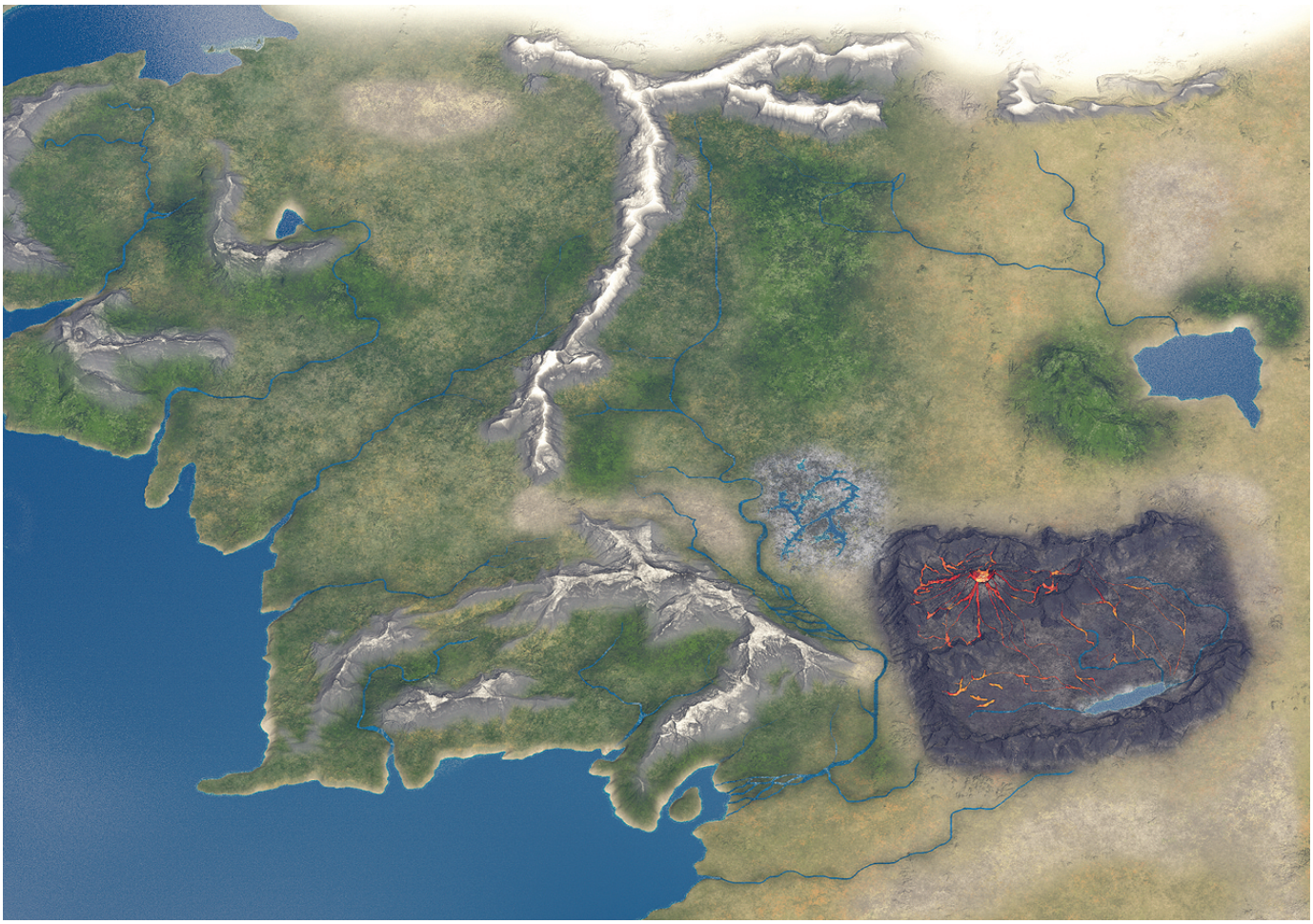
Fig. 13. The Grand Canyon turned into a mountain range. Features extracted from the Grand Canyon DEM are used as the user sketch to synthesize a mountain range following the structure of the Grand Canyon from an elevation map of the Puget Sound style mountain range.

(a) Map of Middle Earth

(b) User sketch with masked water system



(c) Synthesis result



(d) Rendered terrain, synthesized from multiple elevation maps.

Fig. 14. A 3D map of Middle Earth synthesized from multiple elevation maps.

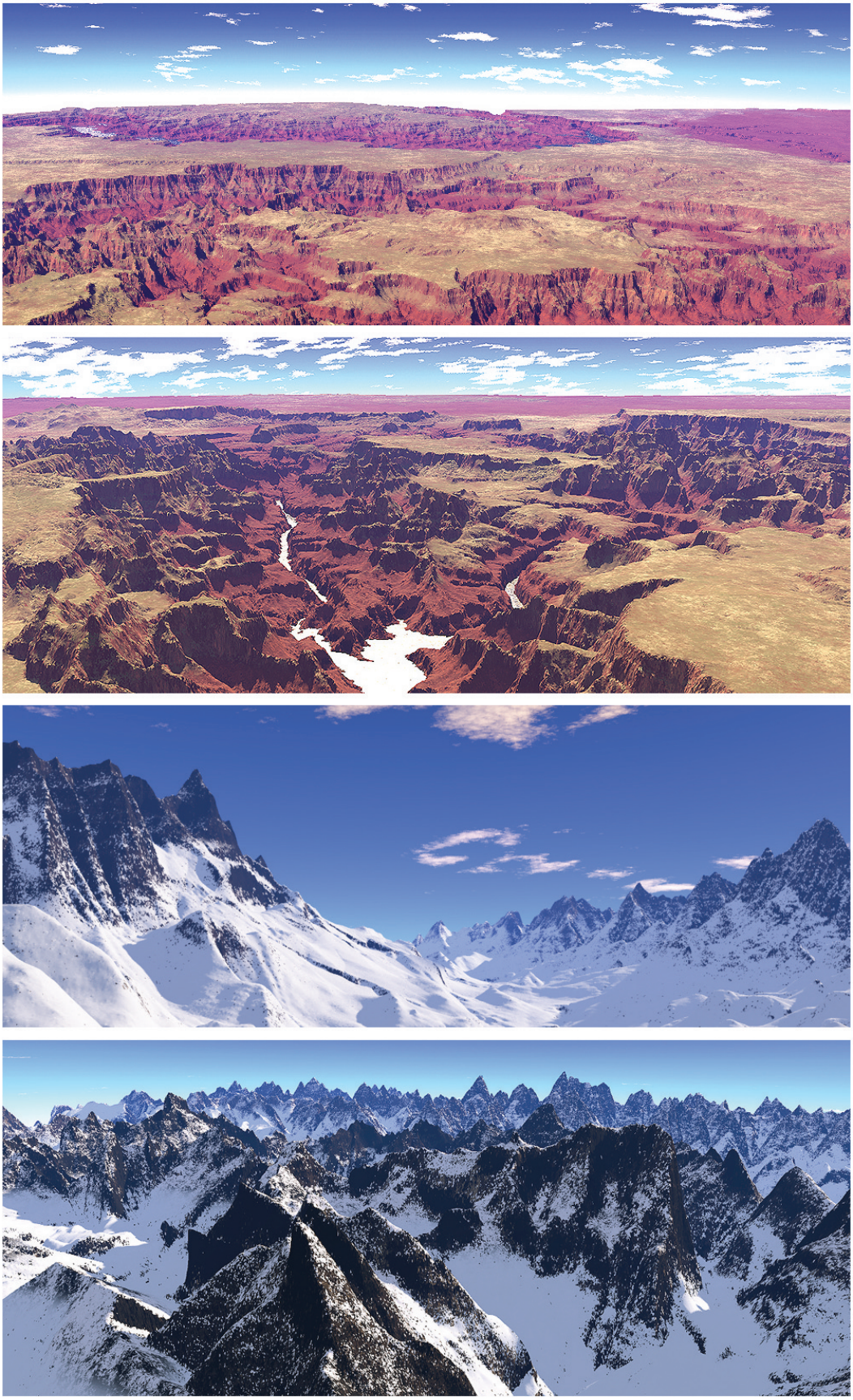


Fig. 15. Close-up views of synthesized terrain from the Grand Canyon (top images) and Flathead National Forest mountain range. (bottom images).

[17] J. Brosz, F. F. Samavati, and M. C. Sousa, "Terrain synthesis by-example," in *GRAPP: 1st International Conference on Computer Graphics Theory and Applications*, 2006.
[18] J.-P. Lewis, "Texture synthesis for digital painting," in *SIGGRAPH 1984*, 1984, pp. 245–252.
[19] K. Perlin and L. Velho, "Live paint: painting with procedural multiscale textures," in *SIGGRAPH 1995*, 1995, pp. 153–160.
[20] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *SIGGRAPH 1995*, 1995, pp. 229–238.
[21] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Int. Conf. Computer Vision*, Corfu, Greece, 1999, pp. 1033–1038.
[22] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *SIGGRAPH 2000*, 2000, pp. 479–488.
[23] M. Ashikhmin, "Synthesizing natural textures," in *2001 symposium on Interactive 3D graphics*, 2001, pp. 217–226.
[24] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *SIGGRAPH 2001*, 2001, pp. 341–346.
[25] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *SIGGRAPH 2001*, 2001, pp. 327–340.
[26] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Trans. Graphics, SIGGRAPH 2003*, vol. 22, no. 3, pp. 277–286, 2003.
[27] Q. Wu and Y. Yu, "Feature matching and deformation for texture synthesis," *ACM Trans. Graphics, SIGGRAPH 2004*, vol. 23, no. 3, pp. 364–367, 2004.
[28] P. Bhat, S. Ingram, and G. Turk, "Geometric texture synthesis," in *eurographics symposium on Geometry Processing*, 2004.
[29] A. Lagae, O. Dumont, and P. Dutré, "Geometry synthesis by example," in *Shape Modeling International*, 2005.
[30] S. Lefebvre and H. Hoppe, "Parallel controllable texture synthesis," *ACM Transactions on Graphics, SIGGRAPH 2005*, pp. 777–786, August 2005.
[31] ——, "Appearance-space texture synthesis," *ACM Transactions on Graphics, SIGGRAPH 2006*, vol. 25, no. 3, pp. 541–548, 2006.
[32] J. Zhang, K. Zhou, L. Velho, B. Guo, and H.-Y. Shum, "Synthesis of progressively-variant textures on arbitrary surfaces," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 295–302, 2003.
[33] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Trans. PAMI*, vol. 11, no. 6, pp. 567–585, 1989.
[34] C. Soler, M.-P. Cani, and A. Angelidis, "Hierarchical pattern mapping," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 673–680, July 2002.
[35] S. L. Kilthau, M. S. Drew, and T. Möller, "Full search content independent block matching based on the fast fourier transform," in *International Conference on Image Processing 2002*, 2002, pp. 669–672.
[36] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. PAMI*, vol. 23, no. 11, pp. 1222–1239, 2001.
[37] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graphics, SIGGRAPH 2003*, vol. 22, no. 3, pp. 313–318, 2003.

APPENDIX A
GRAPHCUT SEAM FINDING

The graph cut algorithm finds the minimum cost seam (according to some matching quality measure) in the overlapping region between patches that determines which pixels will be kept in the final image. We choose the matching quality measure defined in [26], which is a measure of intensity (elevation) difference between the pairs of pixels. For example, in Fig.16, let s and t be two adjacent pixel positions in the overlap region Ω between patch A and B. Let $a(s)$ and $b(s)$ be the elevation at the position of the patches respectively. Then the matching quality measure M between the two adjacent pixels at position s and t from patches A and B is defined to be:

$$M(s,t,A,B) = |a(s) - b(s)| + |a(t) - b(t)| \qquad (1)$$

The graph shown in Fig.16 has one node per pixel in the overlap region between patches. The weight of the edge connecting the adjacent pixel nodes s and t is set to equal

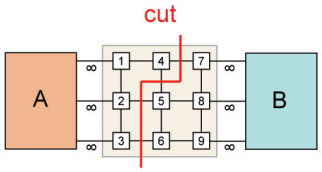


Fig. 16. Finding optimal seam in the overlapping region with Graph cut.

the matching quality cost $M(s,t,A,B)$. We use two additional nodes **A** and **B** to represent the old and new patches. The edges that connect pixels in Ω and nodes **A** and **B** are set to have infinitely high weight indicating that these pixels are constrained to come from one particular patch. In Figure 16, pixels 1, 2, and 3 have to come from the patch A, and pixels 7, 8, and 9 from B. To determine which patch each of the pixels 4, 5, and 6 will come from, we solve a max-flow/min-cut graph problem that minimizes the cost of mismatched elevations across the cut. The red line shows the minimum cut (the elevation difference between the two patches along the cut is minimum). In the overlap region, pixels 5 and 6 will be copied from the old patch B since they are still connected to node **B**. Likewise, pixel 4 will be copied from A. The cost c_g of the minimum cut $\mathscr{C}$ is defined in terms of the matching quality measure as:

$$c_g = \sum_{\substack{<s,t>\in\mathscr{C} \\ s,t\in\Omega}} M(s,t,A,B) \qquad (2)$$

For some patches, we want to insist that particular groups of pixels from a given patch should be included, such as the central area surrounding a branch point feature or along a path. This is accomplished by setting the edge weights connecting those areas to the non-overlapping region infinitely high.

APPENDIX B
POISSON SEAM REMOVAL

Even using the graphcut approach to choose where to join two patches, elevation differences may still be visible. For example, in Fig. 9(a), the overlap region Ω is the new height field from our graphcut seam finder. Though optimum, seam $\mathscr{C}$ is still visible in the new height field (Fig. 9(b)). We will describe how we remove this seam through our Poisson elevation adjustment method.

Let Ω be a closed subset of $\mathbb{R}^2$ with boundary $\partial\Omega$. Let f be the elevation value in the overlap region, hence, f is a scaler function defined over Ω. Our elevation adjustment stage first translates the elevation values in Ω into gradient vector fields $\mathbf{v} = (u,v)$ with $u = \frac{\partial f}{\partial x}$ and $v = \frac{\partial f}{\partial y}$. To remove the height difference between pixels across the seam $\mathscr{C}$, the gradient values along the seam are artificially set to zero. However, the resulting gradient vector field $\mathbf{v}'$ is most likely no longer conservative ($curl(\mathbf{v}') \neq 0$), in other words, it is no longer the gradient of any scaler function. Here, the Poisson methodology comes into play because it allows non-conservative vector fields to be used to reconstruct a plausible elevation field. As shown in [37], we can find the best-fit set of elevations f' to

the adjusted vector field $\mathbf{v}'$ by solving the Poisson equation with Dirichilet boundary conditions.

The result of this process is a new set of elevations that change very little at the seam, as shown in Fig.9(c).

Howard Zhou graduated from the California Institute of Technology in 2002 with the BS degree in Electrical Engineering and Applied Mathematics. He received the MS degree in Computer Science from the Georgia Institute of Technology in 2005, and is currently working towards the PhD degree under the supervision of Dr. James M. Rehg and Dr. Greg Turk. His research interests include computer vision and computer graphics and machine learning.

Jie Sun received the BE degree in Computer Science and Engineering from the Zhejiang University, China in 2002 and the MS degree in Computer Science from the Georgia Institute of Technology in 2005, where he is currently working towards the PhD degree under the supervision of Dr. James M. Rehg and Dr. Aaron Bobick. His research interests include computer vision, machine learning and computer graphics.

Greg Turk received the PhD degree in Computer Science in 1992 from the University of North Carolina (UNC) at Chapel Hill. He is currently an associate professor at the Georgia Institute of Technology, where he is a member of the College of Computing and the Graphics, Visualization, and Usability Center. His research interests include computer graphics, computer vision, and scientific visualization. He is a member of the IEEE.

James M. Rehg received his PhD degree in Electrical and Computer Engineering from the Carnegie Mellon University. He is an Associate Professor in the College of Computing at the Georgia Institute of Technology. He is a member of the Graphics, Visualization, and Usability Center and co-directs the Computational Perception Lab. His research interests are computer vision, computer graphics, machine learning, and robotics. He is a member of the IEEE.