# Terrain Synthesis Using Curve Networks

Maryam Ariyan* David Mould†

Carleton University

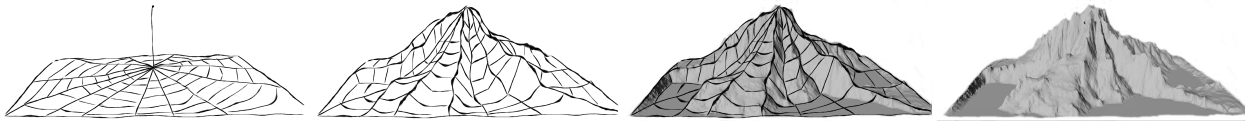

Figure 1: Curve networks used to construct a mountain peak.

**ABSTRACT**

We present a procedural technique for the controllable synthesis of detailed terrains. We generate terrains based on a sparse curve network representation, where interconnected curves are distributed in the plane and can be procedurally assigned height. We employ path planning to procedurally generate irregular curves around user-designated peaks. Optionally, the user can specify base signals for the curves. Then we assign height to the curves using random walks with controlled probability distributions, a process which can produce signals with a variety of shapes. The curve network partitions space into individual patches. We interpolate patch heights using mean value coordinates, after which we have a complete terrain heightfield. Our algorithm enables users to obtain prominent features with lightweight interaction. Increasing the density of curves and roughness of curve profiles adds detail to the synthetic terrains. The curves in a network are organized into a hierarchy, where the major curves are created first and the curves constructed at later stages are affected by earlier curves. Our approach is capable of producing a variety of landscapes with prominent ridges and distinct shapes.

**Index Terms:** 1.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling; 1.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism;

## 1 INTRODUCTION

Terrains are visible in artistic paintings, computer games, and CGI movies. Terrains can be created with manual modeling tools such as Maya [1] or Blender [2]; such tools provide complete control over the output model but demand a high degree of user involvement and expertise.

Conversely, procedural techniques employ algorithms to produce 3D models and textures with little or no input [6]. The challenging aspect of procedural terrain synthesis is to control the result based on the input parameters.

Sketch-based modeling (SBM) methods occupy a middle ground between fully manual and fully procedural techniques. SBM methods for terrain synthesis create terrain surfaces subject to user-drawn constraints. Sketch-based techniques demand less user involvement than manual methods and, compared to procedural methods, provide better control over terrain feature placement. However, the interpolated terrain surfaces exhibit less detail than the inherently rough-looking real-world terrains.

Our goal is to make controllable and realistic landscapes with minimal user intervention. Terrains exhibit intricate detail. There is a trade-off between user effort and the quantity of detail when modeling terrain using manual or sketch-based methods. We can reduce manual effort and get detailed models by using procedural methods but we still need some mechanism to provide high-level control over the terrain structure. We introduce curve networks as a framework for organizing the terrain construction and as a useful entry point for user-specified terrain features. We intend the curve networks to reflect directly the structure of the terrain, with prominent ridges extending outward from mountain peaks; between the ridges, the terrain is somewhat flat or dips down slightly. Focussing the modeling effort on the ridges is desirable as the ridges and associated silhouettes are also the most perceptually salient aspects of the terrain. In effect, we have reduced the dimensionality of the problem: we specify a terrain as a collection of 1D strokes rather than as a 2D surface. The same insight is behind the area of sketch-based modeling.

The idea is illustrated in Figure 1. The terrain begins as a set of isolated peaks; there might only be one peak, or there could be many. Each peak extends a network of curves into its surroundings. Height profiles are assigned to each curve; the patches surrounded by curve elements are then filled in, creating the complete terrain.

In this paper, we concentrated on the procedural aspect of the problem. We make two main contributions. First, we propose the curve network representation for terrain synthesis, including a procedural system for creating curve networks. Second, we present a scheme for synthesizing 1D signals to apply to the curves, using random walks at multiple scales to produce detailed structures; changing the probability distribution of the random walk produces different types of structures. A secondary contribution is the use of mean value coordinates for patch interpolation; because our patches are surrounded by closed polygons, we can directly evaluate interior heights. We discuss the use of and advantages of mean value coordinates later in the paper.

The remainder of this paper is organized as follows. We discuss previous work in Section 2. In Section 3, we give details of the entire terrain synthesis algorithm, including the processes for generating curve networks and for generating interesting profiles using random walks. In Section 4, we show a selection of results, both terrains and profiles. Section 5 contains our discussion of the results, comparison with earlier techniques, and comments about advantages and disadvantages of the approach. We conclude in Section 6 with a summary and some plans for future work.

*maryam@ariyan.net

†mould@scs.carleton.ca

Copyright held by authors. Permission granted to CHCCS/SCDHM to publish in print and digital form, and ACM to publish electronically.

## 2 Previous Work

Broadly speaking, methods for terrain synthesis can be divided into *procedural* methods and *sketch-based* methods. Example-based techniques generally fall into the procedural category, although sketch-based terrain editing techniques such as that of Tasse et al. [21] can be considered example-based.

Terrain is commonly represented as a *height field*: a surface with a unique height value per 2D point. Height fields cannot represent features such as caves and overhangs, but are popular despite this shortcoming owing to the simplicity and convenience of the representation. A recent effort to move beyond height-field terrains is due to Peytavie et al. [17], who combine implicit and discrete volumetric representations to produce terrains with complex topology.

Influential work on procedural terrain synthesis is due to Musgrave et al. [15], who proposed multifractal fractional Brownian motion (fBm). In their method, multiple frequencies of Perlin noise [16] are summed to create a terrain:

$$h(x) = \sum_{i=1}^{n} N(x \cdot 2^{H \cdot i})/L^i \quad (1)$$

where $N(x)$ is the Perlin noise function, $H$ is the fractal increment parameter, and $L$ is called lacunarity. Varying $H$ and $L$ spatially or by height produces plausible landscapes. We use a similar formulation in our own work, but substitute a customized random walk for Perlin noise.

More recent procedural work builds on example-based texture synthesis [7, 8]. Zhou et al. [22] and Brosz et al. [4] rearranged patches from a terrain database to create novel terrains. These methods produce high-quality results, but depend on a database of terrain features.

Distance fields and distance transforms are a potentially useful primitive for terrain and texture synthesis. Rusnell et al. [20] used distance in a weighted graph to create terrains directly; we borrow many of their ideas for this paper. In an effort related to terrain synthesis, Peytavie et al. [18] used the distance transform to synthesize piles of rocks.

Sketch-based modeling is another approach for generating terrains. The first dedicated system for sketch-based terrain is due to Gain et al. [11], who present a sketching framework for terrain synthesis that allows the user to sketch ridge contours and silhouettes of landscapes from different 3D views. The system automatically completes the terrain from the user-sketched hints. Hnaidi et al. [13] generate a variety of terrain features such as ridgelines, riverbeds, hills, and cracks. Dos Passos and Igarashi [5] presented LandSketch, an example-based technique that fits terrain landscapes to silhouette strokes as seen from a first person viewpoint. Tasse et al. [21] use the first-person point of view in their terrain editing technique. They also explore methods to estimate depth information from t-junctions visible in the sketched contours. All these techniques make use of Poisson blending to populate the space between user-sketched features. Other interpolation mechanisms are possible, such as radial basis functions, used by Brazil et al. [3]. However, Poisson blending seems to be the most popular approach at present.

Poisson interpolation involves solving a global system of linear equations, with one equation per pixel; although the system is sparse, it nonetheless scales poorly for large terrains. Farbman et al. [9] suggest that Poisson interpolation is slower than the direct interpolation enabled by mean value coordinates (MVC) [10]. In MVC, the value of each point in the interior of a polygon is calculated based on a weighted sum of the values of the polygon vertices. In our method, we create ridges and assign heights procedurally, then use MVC to determine height values for the rest of the terrain surface.

Our own approach is primarily procedural, but we can consider it to be the back end for a restricted sketch-based modeling system. We have not much investigated possible sketch-based elements of the system to date, however, having concentrated on the procedural aspect. We describe our system in detail next.

## 3 Algorithm

We present an algorithm which creates mountains based on the placement and steepness of mountain peaks, and then procedurally adds ridge detail to the undefined terrain between the peaks. The algorithm consists of three main steps. First, we construct a curve network, consisting of *primary curves* extending outward from the peaks and *secondary curves* linking pairs of adjacent primary curves. Second, we assign heights to points on the curve network; we propose a random-walk-based procedural method for creating rough signals. Third, we fill in the patches bounded by curves, using mean value coordinate interpolation to obtain height values. The above process is illustrated in the teaser, Figure 1: a single peak is shown, from which an initially planar curve network extends. Heights are assigned to the curves, and the patches filled so as to create a complete height field. Of course, it is possible to use more than one peak; a sample curve network with multiple peaks is shown in Figure 2. Even in the case of multiple peaks, most of the detail in the terrain is present on the ridges, as the figure demonstrates: the silhouette only involves 2-3 peaks, but has a jagged shape owing to the random walk. We discuss each stage of the algorithm in more detail in the following subsections.

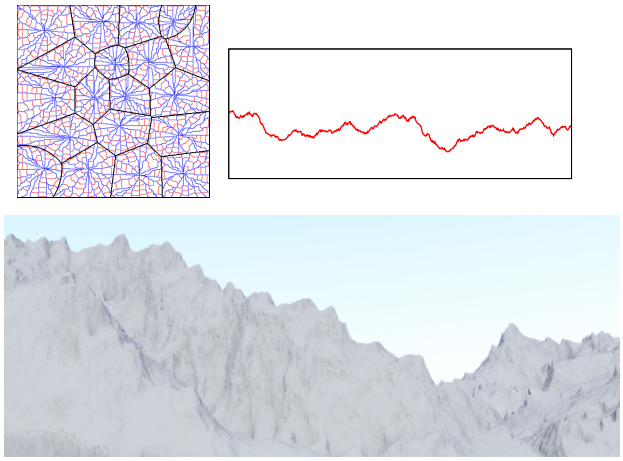

Figure 2: Above: curve network; sample primary profile. Below: resulting terrain.

### 3.1 Curve network generation

Our intent is to create a curve network consisting of primary curves spreading out from peaks and secondary curves approximately perpendicular to primary curves. The primary curves can represent structural details, especially ridges; the secondary curves will add further roughness and can help shape the terrain, for example by sloping downwards away from ridges. We use Dijkstra's algorithm to solve the single-source shortest path problem through a 4-connected graph where each node is a cell at the intended resolution of the terrain. Graph edges have random weights, producing irregular paths.

Each peak will mark a region of surrounding nodes to which it is closest, resulting in a Voronoi-like partition of the graph. We distribute endpoints along the boundary of each region according to a user-specified spacing; then, we trace least-cost paths backwards towards the peak using the distance values previously computed. Optionally, we can use a second pass of Dijkstra's algorithm to compute new distances, if the user wants to change the

edge weights; one application of this idea is to make edges cheaper near the boundary and more expensive further away, so as to obtain paths that are approximately perpendicular to the boundary.

The above process produces the primary curves. Subsequently, we create secondary curves by computing a new pass of Dijkstra's algorithm, starting with all points on each primary curve at distance zero. Doing this produces an irregular partition of space with each primary curve having its own region; boundaries approximately bisect the space between adjacent primary curves. We then construct secondary curves by spacing out endpoints along these region boundaries and tracing back in both directions to the primary curves.

### 3.2 Profile generation

Having constructed curve networks, we next assign heights to all network points. We refer to the elevation along a curve as a *height profile*. Since primary curves are irregular paths connecting region boundaries to peaks, we can exert control over the overall terrain shape by specifying the height profile of primary curves.

For each curve, we can employ a 1D interpolation to set the height profile, given that the elevations of endpoints are known. We obtain an initial estimate of the region boundary heights by taking $h(x) = h_p - k \cdot c(x)$: the height $h$ of point $x$ with cost $c(x)$ is an elevation $k \cdot c(x)$ below peak height $h_p$, for a constant $k$. We can then modify this estimate by using a random walk to displace the heights, as described below. Alternatively, we can do something much simpler such as setting the boundary heights to zero. Once the boundary heights are known, we can use them to interpolate the heights of the primary curves; in turn, once the primary curves have known profiles, we can interpolate the profiles of the secondary curves.

We interpolate heights along the path of a curve element by referring to heights of each curve's endpoints. The simplest approach is linear interpolation. Other kinds of interpolation are possible, e.g., splines. However, we intend to construct more stochastic height profiles so as to reflect the roughness of rocky, mountainous terrains. To get a rough interpolation, we employ random walks to compute profiles.

Recall that Musgrave et al. used multiple instances of Perlin noise to obtain height fields, as shown in equation 1. We use the same formulation, but substitute a random walk for Perlin noise:

$$h(x) = \sum_{i=1}^{n} R_i(x \cdot 2^{H \cdot i})/L^i. \tag{2}$$

In the preceding, $H$ is the fractal increment and $L$ is lacunarity as before; we sum $n$ octaves to obtain the signal. The difference between equations 1 and 2 is the use of a random walk $R(.)$ in the latter, compared to Perlin noise $N(.)$ in the former.

A random walk is an incrementally constructed signal: the sum of a sequence of random samples. A random position after $x$ steps can be computed as follows:

$$R(x) = p_0 + \sum_{i=1}^{x} \langle p \rangle, \tag{3}$$

where $\langle p \rangle$ is a random sample drawn from probability distribution $p$. Note that the above is only defined for integer $x$; for real $x$, compute $R(\lfloor x \rfloor)$ and $R(\lceil x \rceil)$ and interpolate between them. Also note that, as written, $R(x)$ is not stable on repeated evaluations; in practice, we compute a single random walk for each profile and store it.

The advantage of using a random walk is that we can obtain different shapes by sampling from different probability distributions. Consider the "drunkard's walk", illustrated in Figure 3, where there is a 50% likelihood of a positive increment and a 50% chance of a negative increment. The resulting signal fluctuates quickly. A Gaussian PDF centred at zero, conversely, has a high likelihood of a small or zero increment, and hence changes slowly. Figure 4 shows four random walks generated from a Gaussian PDF.

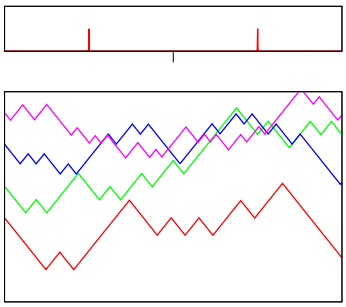

Figure 3: Above: probability distribution (as a histogram); below: four random walks from this distribution.

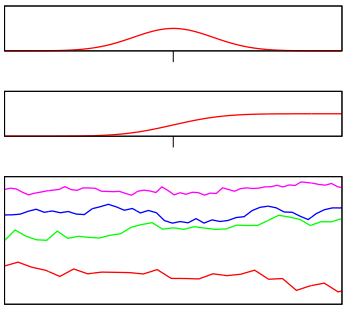

Figure 4: Above: Gaussian PDF (histogram); middle: cumulative distribution; bottom: four random walks generated from the PDF

In general it is not feasible to directly sample from the probability density function. We use the cumulative distribution function (CDF), which describes the probability $C(x)$ that a sample from the distribution is less than or equal to $x$. Figure 4 illustrates the CDF of the Gaussian PDF. Unlike the PDF, the CDF is directly invertible and hence we can use it to sample from the input PDF. We compute the CDF by summing a discretized record of the PDF; this allows us to work directly with arbitrary PDFs, including those drawn by hand by a user. We consider some potentially useful PDFs in the next section.

Recall that we do not use a single random walk, but rather multiple random walks at different scales, according to equation 2. We consistently used $H = 1$ and $L = 2$ throughout the results we show. An example of the multiscale random walk is shown in Figure 5; here we used a user-drawn distribution with a large mode near zero and another mode far on the negative end. We automatically debias the distribution so that its long-range mean is zero, and the resulting random walk has a gentle but varied upward slope punctuated by sudden steep downward increments. We consider this shape to have a pleasing, natural-seeming balance of predictability and unpredictability. The sum of four octaves produces an overall structure with detail at different scales and apparent features.

It is not in general possible to directly interpolate fixed endpoints with a random walk: we can start at one endpoint, which would be $p_0$ in equation 3, but we cannot predict what the value will be at the other endpoint. To make both endpoints line up, we compute a random walk of the desired length, and find the difference between

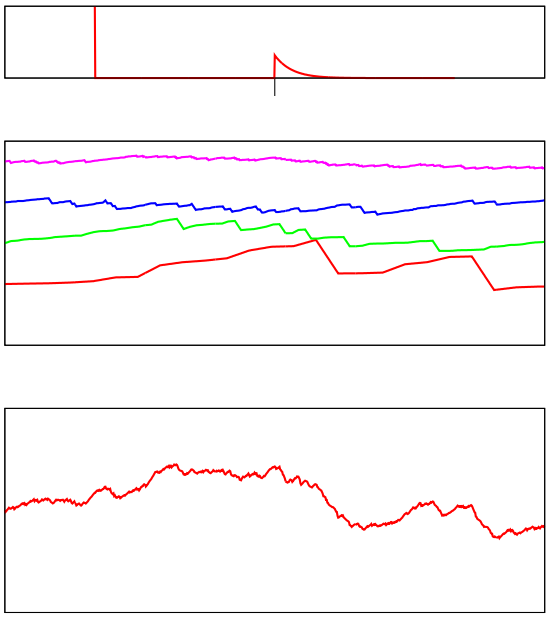

Figure 5: Top: user-drawn probability distribution; middle: four random walks with decreasing amplitudes and increasing frequencies; bottom: sum of the four octaves, to be used as a height profile.

the desired endpoint and the random walk's ending value. Then, we correct for the mismatch by linearly interpolating the displacement along the length of the random walk.

### 3.3 Interpolation of patch interiors

A curve network partitions space into separate *patches* bounded by the curves; curves have known height values after assigning profiles. The next step is to find heights for points inside the patches. For this step, we use mean value coordinates (MVC) [10]. MVC have several desirable properties: they allows interpolation over arbitrary polygons, even non-convex polygons; interior points depend only on points on the boundary; and both random access and arbitrary resolution are possible, i.e., any interior point can be evaluated independently from the others without reference to any underlying discretization of space.

For each patch, we use boundary tracing [12] to find an ordered list of the border points. This set of points then form the polygon for MVC interpolation. We used the algorithm given by Hormann and Floater [14] to evaluate heights at the interior points as a weighted sum of the heights along the boundary. Once the heights of all interior points of all patches have been computed, we have a complete height field, ready to be rendered.

## 4 Results

Next we present synthetic terrains that are produced using our approach. First we discuss the role of different elements of our algorithm in getting varied shapes of terrain. Then we present synthetic terrains alongside real-world terrain images to show the versatility of our approach in generating realistic terrains. We then discuss the role of the random walk and show some examples of specific probability distributions and the resulting profiles.

We implemented our algorithm in C++ running on a 64-bit Intel dual Core i7 CPU 2.40 GHz (and 1.9 GHz) with 8 GB RAM. The output from our program is a heightfield; we used Terragen [19] to render terrains. The renderings contain small-scale displacement noise with no textures on top of the surfaces.

Our algorithm is composed of three main stages: curve network construction, height profile assignment on curve elements, and patch height interpolation. We exert control over the shape of terrains by constructing different types of curve networks and by assigning profiles to curves. In this section, we demonstrate different terrains that can be created procedurally using our method. Figure 6 we show a collection of terrain examples. Collectively, the terrains show the range of features we can straightforwardly produce. Particularly take note of the irregular silhouettes, created using few peaks; the rough terrain surfaces, with detail oriented approximately vertically (away from the peaks); and the heterogeneity of the visible structures, created by the combination of curve networks and the application of various probability distributions along the curves. Figure 7 shows comparisons with photographs of real-world terrains; similar structures can be seen in the corresponding synthetic terrains.

One of the ingredients for creating the terrains is the probability distribution that governs the random walk for the curve profiles. We show a few examples of probability distributions and sample resulting random walks in Figure 8. Each probability distribution was sketched manually; the graphs show both the distributions (as histograms of possible increments) and an example random walk.

The topmost example shows a simple PDF: it is akin to the drunkard's walk, but with broader distributions over the positive and negative outcomes. The resulting walk is similar to the original drunkard's walk but with a wider range of slopes apparent.

The second example is an arbitrary user-drawn distribution. The resulting walk has a range of slopes, with visible structural elements resembling notches and plateaus. Because the original sketch was biased, with more likelihood of obtaining a positive increment than a negative one, we automatically debias it by incrementally reducing the histogram entries on the heavier side until the two are in balance.

The third example shows a trimodal distribution with a large mode near zero and small modes at large slopes. The resulting walk has a range of slopes, with visible structural elements resembling notches and plateaus. Because the original sketch was biased, with more likelihood of obtaining a positive increment than a negative one, we automatically debias it by incrementally reducing the histogram entries on the heavier side until the two are in balance. The last example shows a variant of the same trimodal distribution with larger tails: the resulting random walk is flat in places but is overall much more jagged than the previous profile.

## 5 Discussion

Here, we will discuss some attributes of our method and give comparisons to terrains created with previous methods.

### 5.1 Comparisons

The previous work most similar to ours is that of Rusnell et al. [20], who proposed Dijkstra-based distance calculations for height field synthesis. They created an entire height field with a single Dijkstra pass, obtaining unified terrains but with limitations on the shape of features: in their terrains, heights monotonically decrease with distance to the peaks. Our approach separates the ridge placement from the profile assignment, and our customized random walks offer more detail and variety than the silhouettes seen in the terrain from Rusnell et al. Figure 9 shows a side-by-side comparison.

Gain et al. created the first sketch-based terrain synthesis system, and we compare against it in Figure 10. We created the same dome shape (centre) and somewhat irregular silhouettes (right, middle left) but our islands have vastly more detail. Gain et al. suggest some detail by applying a texture, but the terrain surface is smooth, a consequence of their Poisson blending between sparsely sketched

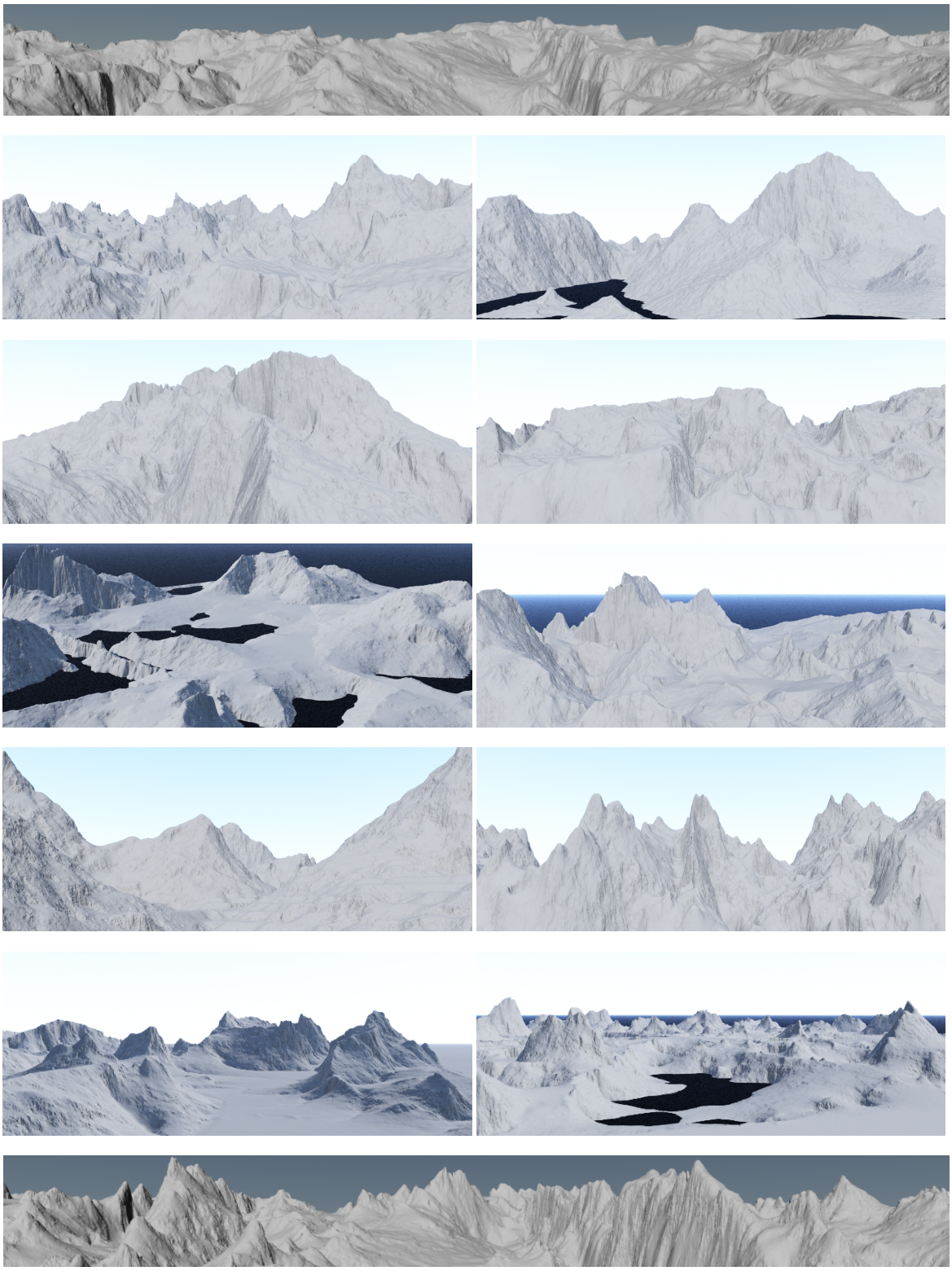

Figure 6: Several synthetic terrains made using our system.

Figure 7: Real terrains. Above: Navarino; middle: Karakoram; below: Aleutians.

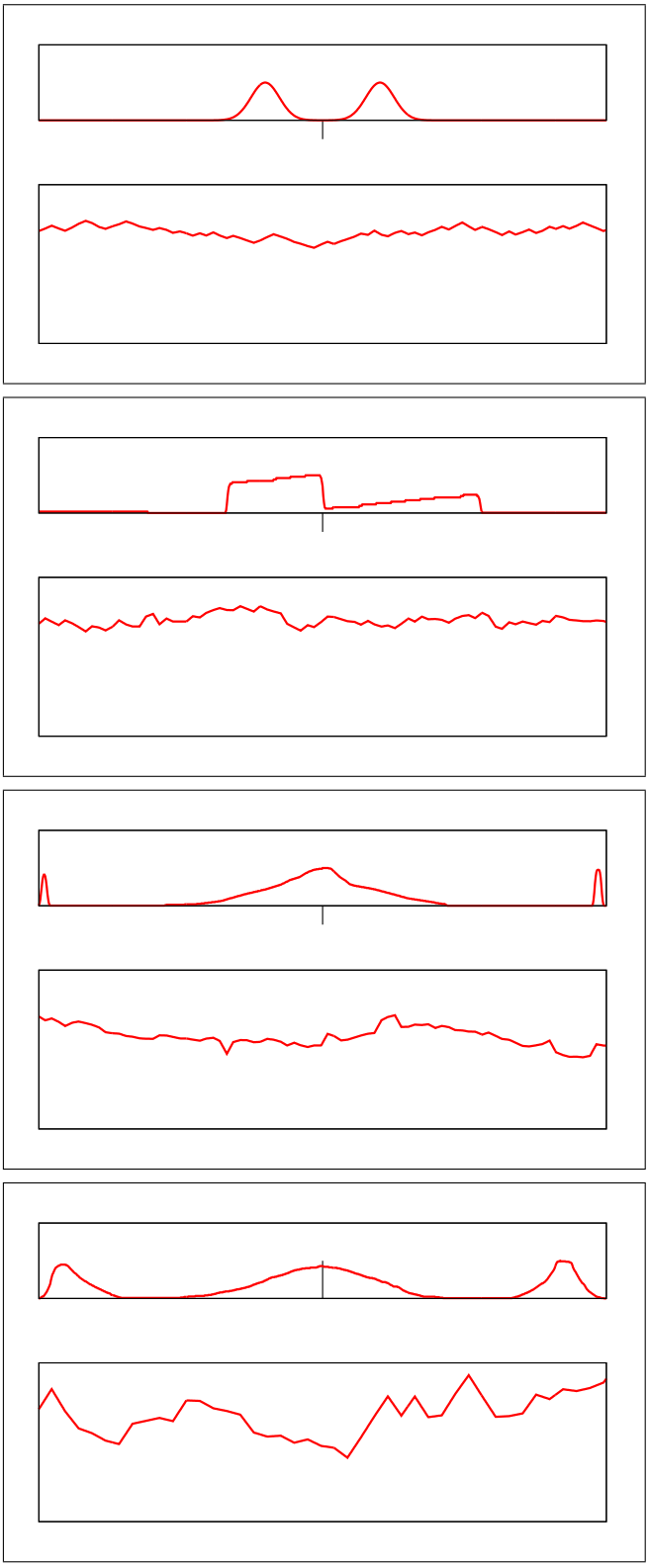

Figure 8: Four example probability distributions and random walks. Each example contains a user-drawn probability distribution (above) and a sample random walk from the distribution (below).

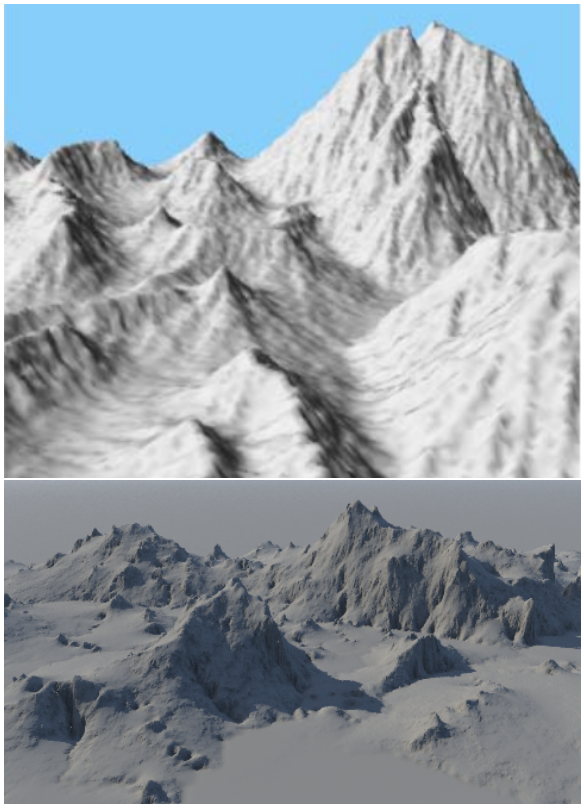

Figure 9: Above: Rusnell et al.'s result. Below: our result.

strokes. Our silhouettes are more intricate than the silhouettes seen in their result, since our procedural profiles have multiple octaves of random noise, while theirs come from pen strokes: it would take an inordinate amount of human attention to add this much detail.

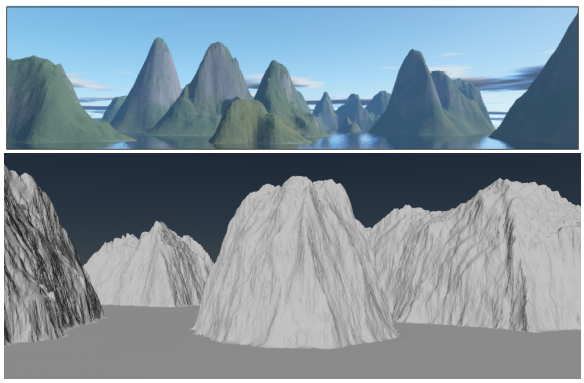

Figure 10: Above: Gain et al.'s result. Below: our result.

Hnaidi et al. offer a more recent and sophisticated terrain sketching system, also based on the Poisson equation. They use Perlin noise to add detail to an otherwise smooth terrain, but their results do not have the level of detail or heterogeneity that ours exhibit. A direct comparison can be seen in Figure 11.

We omitted a comparison with Tasse et al. since they are doing terrain editing, not synthesis: their work is not intended to create novel terrains, unlike ours. Therefore, they did not seek to add any detail: their output terrains can be quite detailed without any extra effort, as they preserve most of the details in the input terrains.

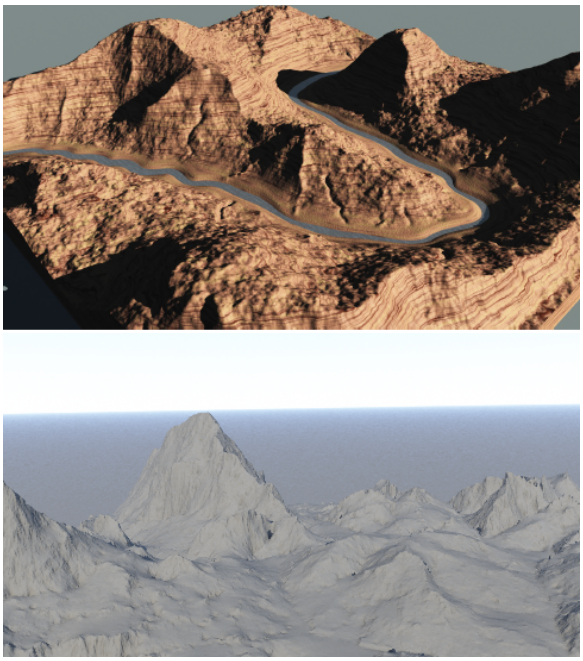

Figure 11: Above: Hnaidi et al.'s result. Below: our result.

Because of the dissimilarity in our goals, we did not believe that a direct comparison would be very informative.

### 5.2 Timing

A typical run of the system at a resolution of $400 \times 400$ takes four to eight minutes. The majority of this time (about three quarters) is due to the patch interpolation; the rest is in the curve generation. Profile generation takes less than one second. The variability is due to variation in patch sizing: smaller patches (with fewer boundary points per patch) are faster to compute on a per-pixel basis.

Timing is with respect to a severely non-optimized implementation; we are confident that simply by streamlining the code we would be able to eke out at least an order of magnitude speedup, probably more. Even in its current form, the system is usable, albeit stretching the limits of a user's patience for an interactive system.

Because the method is procedural, there is minimal human effort: only a few seconds are needed to choose peak positions and random walk properties. Even the peak placement could be automated, although we have opted not to spend much attention investigating this idea. Of course, to obtain an appealing synthetic image, scene composition and lighting require time and attention.

Unlike other procedural methods, with our approach, the user is able to direct the terrain more closely if desired. The profile of any curve can be set manually instead of procedurally. We have not explored this possibility, but it seems that the curve network itself could be constructed manually if desired. The process we describe can be calibrated to the user's desired level of involvement, from virtually none to specifying many curve properties, and a plausible terrain can be completed given any level of user input.

### 5.3 Advantages and Limitations

There are several advantages to our method. It is procedural, but the user can step in to supply information at various levels of detail if desired. The use of random walks affords us flexibility in generating different shapes of profiles. We did not explore it in this paper, but we can make use of the $H$ and $L$ parameters to create multifractal terrains as Musgrave et al. did [15]. However, $H$ and

$L$ are difficult to tune to produce the desired appearance, and spatially varying them is an even greater challenge; conversely, we can alter the profile settings spatially or on a curve-by-curve basis, as desired.

Our use of mean value coordinates rather than Poisson blending is intended to be fast. The speed advantages of MVC do not manifest in our implementation, but in general, we believe MVC to be a better option: it has better scalability, it allows local evaluation, and it allows evaluation at arbitrary resolution. Our curve networks create the bounded patches necessary for MVC.

Our method has some limitations. It is restricted to heightfield generation. In its present implementation, it is slow. Creating profiles by sketching probability distributions is indirect; the process is inscrutable to novice users.

Our philosophy of primary curves representing ridges is suitable for the mountainous terrains we showed, but is not as well suited to other types of terrains or terrain features, especially flat and smooth terrains such as hills and swamps. Poisson-based terrain synthesis is already well-suited to smoother terrains; we have sought to make it easy to make rugged terrains, but have not made specific efforts towards using our system to make smooth surfaces.

## 6 Conclusion

This paper proposed curve networks for terrain synthesis, where a hierarchy of curves is created in 2D, then heights are procedurally assigned to points on curves. and a terrain surface is interpolated from the sparse curves. We used a random walk mechanism to create rough terrains, controllable through user-specified probability distributions. The division of labor between curve placement, curve profile creation, and surface interpolation is helpful. The use of mean value coordinates for interpolation allows local evaluation and we expect that it can be faster than the popular alternative of Poisson interpolation, following the direction of Farbman et al. [9].

Our method controls both large-scale and fine-scale detail of terrain models. We can control the placement of peaks and other terrain properties such as roughness and ridge shapes and we can produce structural and prominent ridge detail around input peaks and ridges. We generate realistic rough terrains by increasing the density of curves or by making rough random walk curve profiles in the gaps between the scattered input peaks. We can control the shape of ridges outwards from peaks by specifying base profiles for curves onto which we superimpose smaller-scale random walks.

Some challenges remain for future work. Multiscale path planning would allow faster terrain construction and higher-resolution terrains. We would like to incorporate example-based synthesis by drawing the profiles from exemplars or synthesizing profiles by connecting exemplar fragments, a task easier in 1D curves than 2D terrains. It would be worthwhile to have roughness within patches, rather than interpolate smoothly; a step in this direction would be to create a deeper hierarchy with smaller patches. Overall, we believe that the curve network mechanism will be a useful basis for further work in terrain synthesis.

## Acknowledgements

The authors wish to thank the anonymous reviewers and our colleagues in GIGL for their helpful suggestions . This work was supported by NSERC and by the GRAND NCE. Photos were provided by flickr users “El Guanche” (Navarino), Stefanos Nikologianis (Karakoram), and Allan Shimada (Aleutians).

## References

[1] Autodesk. Maya. `http://www.autodesk.com/products/maya/overview`. [Accessed: 2014-10-30].

[2] Blender Foundation. Blender. `http://www.blender.org/`. [Accessed: 2014-10-30].

[3] E. Brazil, I. Macedo, M. C. Sousa, L. H. de Figueiredo, and L. Velho. Sketching Variational Hermite-RBF implicits. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, pages 1–8. Eurographics Association, 2010.

[4] J. Brosz, F. F. Samavati, and M. C. Sousa. Terrain synthesis by-example. In *Advances in Computer Graphics and Computer Vision*, pages 58–77. Springer, 2007.

[5] V. A. dos Passos and T. Igarashi. LandSketch: A first person point-of-view example-based terrain modeling approach. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, pages 61–68. ACM, 2013.

[6] D. S. Ebert. *Texturing & modeling: A procedural approach*. Morgan Kaufmann, 2003.

[7] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[8] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[9] Z. Farbman, G. Hoffer, Y. Lipman, D. Cohen-Or, and D. Lischinski. Coordinates for instant image cloning. In *ACM Transactions on Graphics (TOG)*, volume 28, page 67. ACM, 2009.

[10] M. S. Floater. Mean value coordinates. *Computer aided geometric design*, 20(1):19–27, 2003.

[11] J. Gain, P. Marais, and W. Straßer. Terrain sketching. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 31–38. ACM, 2009.

[12] R. Gonzalez and R. Woods. *Digital image processing*. Pearson Prentice Hall, 2008.

[13] H. Hnaidi, E. Guérin, S. Akkouche, A. Peytavie, and E. Galin. Feature based terrain generation using diffusion equation. In *Computer Graphics Forum*, volume 29, pages 2179–2186. Wiley Online Library, 2010.

[14] K. Hormann and M. S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph.*, 25(4):1424–1441, Oct. 2006.

[15] F. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrains. In *ACM SIGGRAPH Computer Graphics*, volume 23, pages 41–50. ACM, 1989.

[16] K. Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[17] A. Peytavie, E. Galin, J. Grosjean, and S. Merillou. Arches: A framework for modeling complex terrains. In *Computer Graphics Forum*, volume 28, pages 457–467. Wiley Online Library, 2009.

[18] A. Peytavie, E. Galin, J. Grosjean, and S. Merillou. Procedural generation of rock piles using aperiodic tiling. In *Computer Graphics Forum*, volume 28, pages 1801–1809. Wiley Online Library, 2009.

[19] Planetside Software. Terragen. `http://planetside.co.uk/`. [Accessed: 2014-10-30].

[20] B. Rusnell, D. Mould, and M. Eramian. Feature-rich distance-based terrain synthesis. *The Visual Computer*, 25(5-7):573–579, 2009.

[21] F. P. Tasse, A. Emilien, M.-P. Cani, S. Hahmann, and A. Bernhardt. First person sketch-based terrain editing. In *Proceedings of the 2014 Graphics Interface Conference*, pages 217–224. Canadian Information Processing Society, 2014.

[22] H. Zhou, J. Sun, G. Turk, and J. M. Rehg. Terrain synthesis from digital elevation models. *Visualization and Computer Graphics, IEEE Transactions on*, 13(4):834–848, 2007.