Modeling Real-World Terrain with Exponentially Distributed Noise

Ian Parberry
University of North Texas

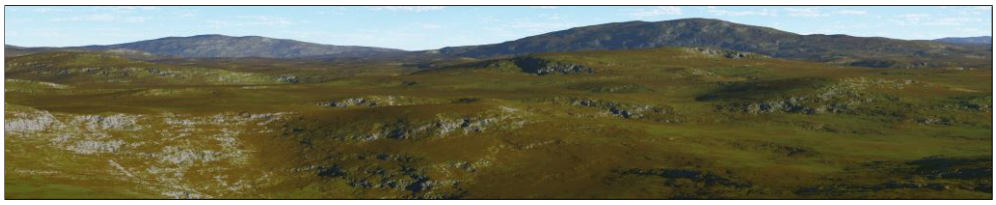

Figure 1. Terrain rendered from Perlin noise with an exponential gradient distribution.

Abstract

A statistical analysis of elevation data from a 160,000 square kilometer region at horizontal intervals from 5 meters up to 164 kilometers finds that terrain gradients appear to be exponentially distributed. Simple modifications to the Perlin noise algorithm and the amortized noise algorithm change the gradient distribution in each octave to an exponential distribution, resulting in varied and interesting procedurally generated terrain.

1. Introduction

Using techniques borrowed from the geosciences, we show how to synthesize varied and interesting artificial terrain for the entertainment industry, which is not so much interested in veracity as it is in triggering the casual viewer's willing suspension of disbelief. Geographers have long been aware of the fractal nature of terrain, for example, Tobler's First Law of Geography [Tobler 1970] states that "*All things are related, but nearby things are more related than distant things*", and Mark and Smith [2004] notes the probable fractal nature of terrain when the horizontal resolution is varied.

The geoscience community has studied the frequency distribution of gradients at a small number of resolutions over relatively small areas; for example, O'Neill and Mark [1987] and Iwahashi et al. [2003] examined the frequency distribution of gradient at a single resolution in areas chosen for their interesting geomorphology.

The latter noted that in an area of Japan noted for slump, slide, and creep type landslides, the gradients appeared to conform to a Weibull distribution. Pain [2005] noted that gradient frequency distribution changes with resolution, and gives qualitative evidence based on resolutions of 25 m, 50 m, and 100 m compared to the results of a ground survey.

We investigate how terrain gradients are related in the large by measuring gradients at at 16 horizontal resolutions from 5 meters up to 164 kilometers within an area of 160,000 square kilometers. The US state of Utah was chosen because it is generic but varied in the sense that it has both mountainous and flat terrain, and because of the availability of autocorrelated DEM files. We describe how the resulting gradient distribution can be incorporated into implementations of Perlin noise [Perlin 1985] and amortized noise [Parberry 2014].

2. Gradient Analysis

We performed a statistical analysis of gradients computed from autocorrelated elevation data in the Digital Elevation Model (DEM) format at 5 m resolution from the Utah Automated Geographic Reference Center[1], which were created using stereoscopic imagery from a 2006 fly-over filtered using LiDAR processing techniques. Although the 5-meter autocorrelated DEM provides higher resolution and horizontal accuracy than the USGS NED dataset, there are acknowledged anomalies present within the data. We have assumed for the purposes of this paper that these anomalies are not statistically significant.

Our data set consisted of 400 text files each containing the elevation of points on a 4000×4000 grid at 5 meter resolution, giving a total of 6.4 billion elevations over the square of side 400 km within the southern part of the US state of Utah shown in Fig. 4. Each elevation value is given to one decimal place in meters. These files occupy a total of 51GB on disk.

The elevation data was analyzed in a series of *octaves*. Octave 1 is the original data at 5 m resolution. Octave 2 is sampled at every alternate point (see Figure 2), giving a 10 m resolution. For $1 \leq i \leq 16$, octave i sampled every 2^{i-1}th point, giving a resolution of $\delta_i = 5 \times 2^{i-1}$ meters. The gradient between two grid points with elevation h_1 and h_2 meters in octave i is therefore $|h_1 - h_2|/\delta_i$.

To avoid loss of statistical significance with each successive octave, each grid size was sampled four times, once as described, and once each offset by half the separation distance horizontally, vertically, and diagonally. The number of sampled grids therefore increases exponentially with octave, offsetting the exponential decrease in the number of points in each grid (see Figure 3). Suppose, for example, the first octave has an $n \times n$ grid of n^2 points, where n is odd (the case where n is even is similar and is

[1] Available from http://gis.utah.gov/data/

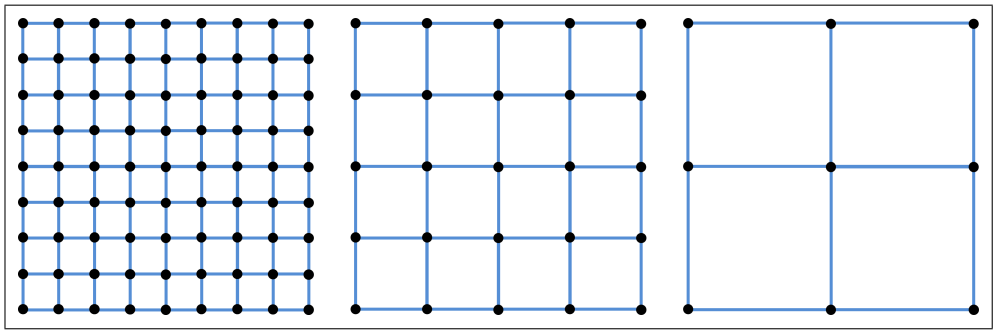

Figure 2. Example of sampling grids for three successive octaves, viewed from above looking down.

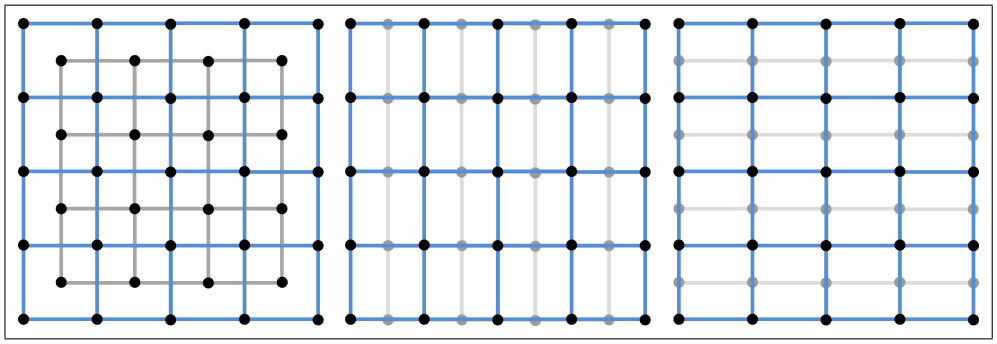

Figure 3. To avoid loss of statistical significance with each successive octave, each grid is sampled four times in total, offset by half the grid separation distance.

left as an exercise for the reader). The second octave has four grids, one of dimension $\lceil n/2 \rceil \times \lceil n/2 \rceil$ (the blue grid in Figure 3), one of dimension $\lfloor n/2 \rfloor \times \lfloor n/2 \rfloor$ (the gray grid in Figure 3, left), one of dimension $\lceil n/2 \rceil \times \lfloor n/2 \rfloor$ (the gray grid in Figure 3, center), and one of dimension $\lfloor n/2 \rfloor \times \lceil n/2 \rceil$ (the gray grid in Figure 3, right), giving a total of $\lceil n/2 \rceil^2 + 2\lceil n/2 \rceil \lfloor n/2 \rfloor + \lfloor n/2 \rfloor^2 = n^2 - \mathrm{O}(n)$ sample points for the second octave, etc.

The analysis was performed by a program written by the author in C++. It took 41 minutes to read and parse the data into an array of `floats` occupying 25.6GB of memory on an Intel® Core™ i7-3930K CPU @ 3.2GHz with 32GB of RAM and a solid state hard drive. The subsequent analysis took approximately 16 minutes. Gradient distributions for octaves 1–9 (which are sampled at horizontal resolutions from 5 m to 1.28 km respectively) are shown in Figure 5. All appear to be an exponential distribution.

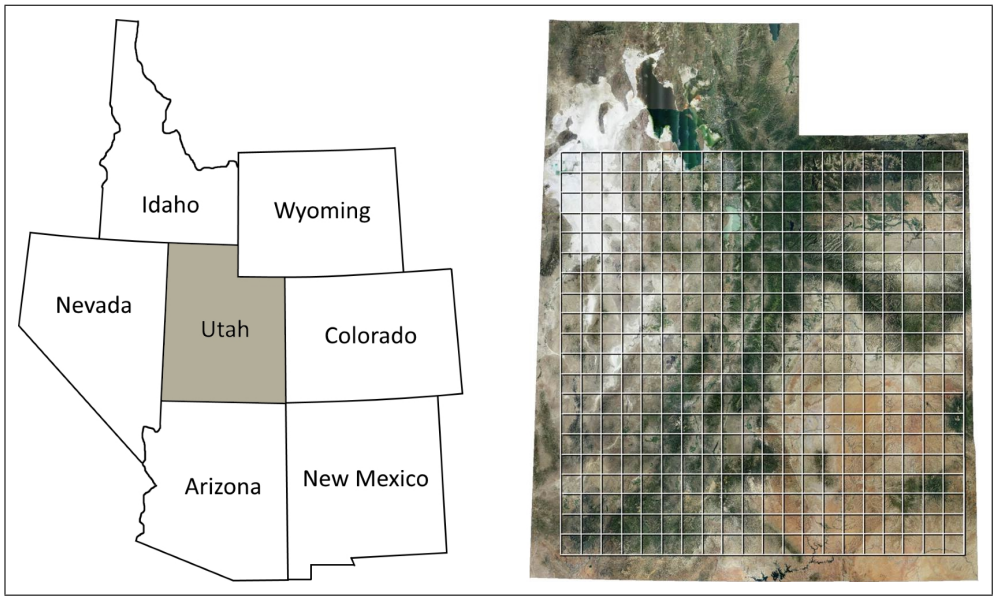

Figure 4. Left: The state of Utah. Right: The 160,000 km^2 area chosen for the gradient study. Each of the 400 squares is covered by a DEM file containing height data for a 2000×2000 grid of points spaced 5 meters apart.

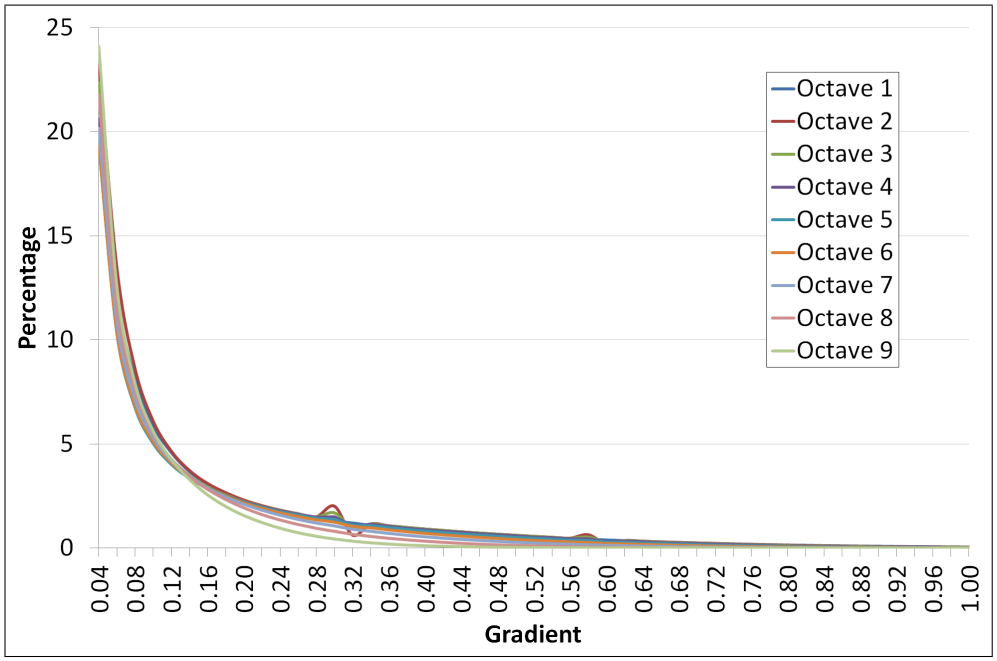

Figure 5. Gradient distribution for octaves 1–9 (distances 5 m–1.28 km).

Figure 6. Terrain generated from Perlin noise. Notice that there are large gradients almost everywhere, resulting in a terrain lacking in landmarks.

3. The Terrain Generator

Ken Perlin's [1985] continually varying smooth random noise function is frequently used for terrain generation. Table 1 (left) gives pseudocode for generating a single octave of Perlin noise. Multiple octaves are generated with successively smaller granularity and elevation and summed to give a final value. Figure 6 shows terrain from Perlin noise rendered by Terragen 3 (`http://planetside.co.uk/products/terragen3`). Notice that since all gradient vectors have unit magnitude, the terrain looks similar in all directions.

Examining Table 1 (left) in more detail, the input is a 2D vector $\vec{p}$ representing the coordinates of a point on the horizontal plane and the output is a floating point value of magnitude less than unity representing the elevation of the terrain at point $\vec{p}$. Function $\text{s\_curve}(x) = x^2(3-2x)$ computes a cubic spline of $0 \le x \le 1$, and $\text{lerp}(t,a,b) = a + t(b-a)$ linearly interpolates between a and b by fraction $0 \le t \le 1$. h is a hash function that maps from 2D vectors to unit length 2D vectors. Perlin's algorithm approximates a hash function by pre-computing a table of $B = 256$ randomly chosen unit vectors, whereas modern versions of Perlin's algorithm (for example, [Parberry 2014]) substitute a more robust hash function. The operation "$\cdot$" is vector dot product.

Following typical noise function notation, the vertices $\vec{p}_{01}$, $\vec{p}_{10}$, and $\vec{p}_{11}$ are the four integer grid points surrounding fractional location $\vec{p}$ [not four different values of $\vec{p}$]. More precisely, for $\vec{p} = [x,y]$, let $\vec{p}_{ij} = [\lfloor x \rfloor + i, \lfloor y \rfloor + j]$.

	Original	Modified
0.	Input $\vec{p} = [x, y]$	Input $\vec{p} = [x, y]$
1.	$s_x = \text{s\_curve}(x)$	$s_x = \text{s\_curve}(x)$
2.	$s_y = \text{s\_curve}(y)$	$s_y = \text{s\_curve}(y)$
3.	$u = \vec{p} \cdot h(\vec{p}_{00})$	$u = h'(\vec{p}_{00})\vec{p} \cdot h(\vec{p}_{00})$
4.	$v = \vec{p} \cdot h(\vec{p}_{01})$	$v = h'(\vec{p}_{01})\vec{p} \cdot h(\vec{p}_{01})$
5.	$a = \text{lerp}(s_x, u, v)$	$a = \text{lerp}(s_x, u, v)$
6.	$u = \vec{p} \cdot h(\vec{p}_{10})$	$u = h'(\vec{p}_{10})\vec{p} \cdot h(\vec{p}_{10})$
7.	$v = \vec{p} \cdot h(\vec{p}_{11})$	$v = h'(\vec{p}_{11})\vec{p} \cdot h(\vec{p}_{11})$
8.	$b = \text{lerp}(s_x, u, v)$	$b = \text{lerp}(s_x, u, v)$
9.	Output $\text{lerp}(s_y, a, b)$	Output $\text{lerp}(s_y, a, b)$

Table 1. The original 2D Perlin noise algorithm (left) and the exponentially distributed version (right). Note that the changes require an extra hash function evaluation and scalar multiplication on lines 3, 4, 6, and 7.

Table 1 (right) shows the modified version of Perlin noise. Let h' be a hash function that maps onto floating point numbers between zero and unity with an exponential distribution. This can be implemented using a magnitude table `m` initialized with with exponentially decreasing values $\texttt{m[i]} = \mu^{\texttt{i}}$ for some $0 < \mu \leq 1$ (details can be found in the code included in the Supplemental Materials). The smallest value in the gradient magnitude table `m`, which is $\texttt{m[B-1]} = \mu^{\texttt{B}-1}$, should be one that can actually be represented as a floating point number. The smallest normalized floating point value is $2^{-(2^f-2)}$, where f is the number of bits in the exponent ($f = 7$ for a `float`, $f = 10$ for a `double`). We therefore want $\mu^{\texttt{B}-1} \leq 2^{2^f-2}$, that is, $\mu \leq 2^{(2^f-2)/(\texttt{B}-1)}$. Using the standard implementation of Perlin noise with `floats` and $\texttt{B} = 256$ means that we should ensure that $\mu \leq 2^{126/256} < 1.1637$.

Figures 1 and 7 show terrain from exponentially distributed Perlin noise (also rendered by Terragen 3). Notice that since the gradient vectors have exponentially distributed magnitudes, the terrain has mostly flat and gently sloping terrain, with occasional areas of higher gradient that create interesting geographical artifacts such as cliffs and escarpments. Contrast this to Figure 6.

Perlin noise repeats with period $n \cdot B$, where n is the number of interpolated points between integer values. This means that any supposedly "infinite terrain" generated from Perlin noise will in fact repeat. Amortized noise [Parberry 2014] is a fast method for generate potentially infinite non-repeating 2D noise. It uses unit gradients $\vec{g}([x, y]) = [\cos(h(x, y)), \sin(h(x, y))]$, where h is a 2D hash function. It is again relatively easy to modify the code to draw the gradient magnitudes from an exponential distribution using the *inverse transform sampling method*, which says that an exponential distribution can be achieved by applying the inverse of the cumulative proba-

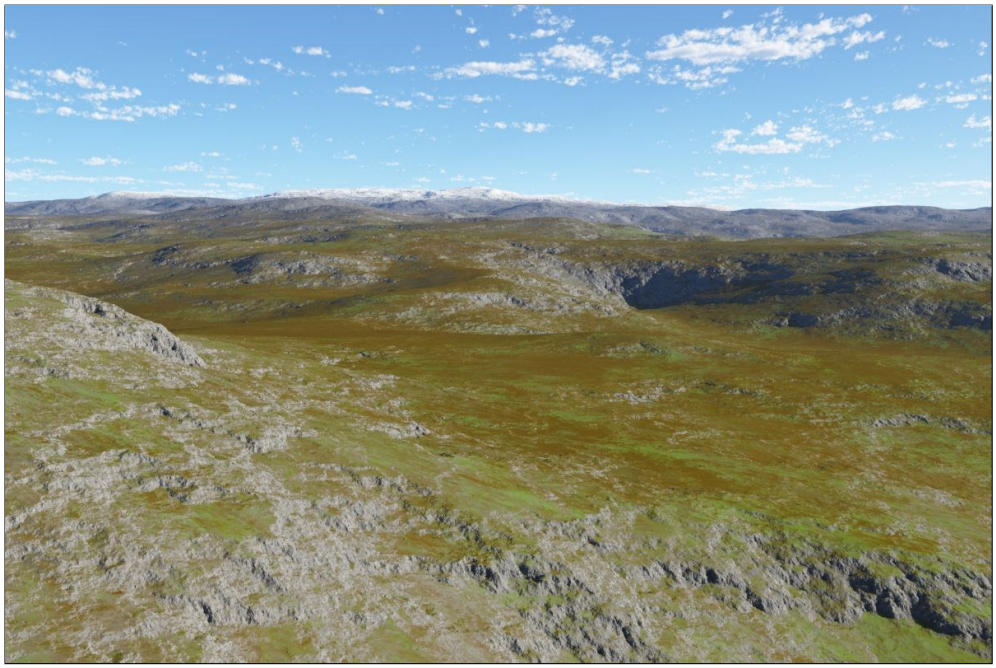

Figure 7. Terrain generated from exponentially distributed Perlin noise. Notice that it only occasionally contains gradients as large as those of Figure 6, which create recognizable landmarks in the form of cliffs and escarpments.

bility density function (that is, the logarithm) to values sampled uniformly at random. Implementation details can again be found in the code included in the Supplemental Materials.

4. Conclusion and Further Work

Some interesting open questions remain. We conjecture that gradients are exponentially distributed over large enough areas in all parts of the world. Nonetheless, a cursory examination of Figure 5 reveals some interesting irregularities at gradients 0.28–0.34. These may not be significant, but on the other hand they may be characteristic of the area under study. It is interesting to ask whether local variations in gradient distribution can be used to generate interesting terrain.

References

IWAHASHI, J., WATANABE, S., AND FURUYA, T. 2003. Mean slope-angle frequency distribution and size frequency distribution of landslide masses in Higashikubiki area, Japan. *Geomorphology 50*, 4, 349–364. http://www.sciencedirect.com/science/article/pii/S0169555X02002222. 1

MARK, D. M., AND SMITH, B. 2004. A science of topography: From qualitative ontology to digital representations. In *Geographic Information Science and Mountain Geomorphography*, M. P. Bishop and J. Shroder, Eds. Springer-Praxis, 75–100. 1

P. O'NEILL, M., AND MARK, D. M. 1987. On the frequency distribution of land slope. *Earth Surface Processes and Landforms 12*, 127–136. http://onlinelibrary.wiley.com/doi/10.1002/esp.3290120203/abstract. 1

PAIN, C. 2005. Size does matter: Relationships between image pixel size and landscape process scales. In *MODSIM 2005, International Congress of Modelling and Simulation*, Modelling and Simulation Society of Australia and New Zealand, 1430–1436. 2

PARBERRY, I. 2014. Amortized noise. *Journal of Computer Graphics Techniques 3*, 2, 31–47. http://jcgt.org/published/0003/02/02/. 2, 5, 6

PERLIN, K. 1985. An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, ACM SIGGRAPH, 287–296. http://doi.acm.org/10.1145/325334.325247. 2, 5

TOBLER, W. 1970. A computer movie simulating urban growth in the Detroit region. *Economic Geography 46*, 2, 234–240. http://www.jstor.org/stable/143141?seq=1#page_scan_tab_contents. 1

Index of Supplemental Materials

Further information (including a longer draft of this paper, terrain images, and fly-through videos) can be found at http://larc.unt.edu/ian/research/tobler/. Source code and data is available under the GNU All-Permissive License at https://github.com/Ian-Parberry/Tobler. There you will find five folders, each of which contains a program that will help reproduce the results of this paper. You will find full C++ source code that compiles under both Visual Studio 2012 and gcc. Each folder contains a Microsoft Visual Studio 2012 project and a Unix makefile.

`Generate with Perlin Noise`. To generate terrain using Perlin noise with an exponential gradient distribution, compile and run the program in this folder which will generate random terrain in a DEM file `output.asc`. You will also find a Terragen project file `output.asc` that can be used to render the terrain from `output.asc`. A sub-folder called `Terrain Images` contains some supplementary images.

`Generate with Amortized Noise`. This folder contains a second version of the Generate program using amortized noise. A sub-folder called `Terrain Images` contains some supplementary images.

`Exponential Distribution`. To verify the code for generating exponentially distributed random numbers, compile and run the program in this folder. An Excel spreadsheet called `exponential.xlsx` contains a copy of some data generated by this program with the corresponding distribution graphs.

`Pack`. To verify the gradient analysis in Section 2, begin by downloading the DEM files listed in `filelist20x20.txt` from the Utah Automated Geographic Reference

Center at `http://gis.utah.gov/data/`. You will need approximately 51GB of disk space to store these files. Compile and run the Pack program, which will read the DEM data and pack it into a binary file `UtahDEMData.bin` for faster processing. You will need an additional 11GB of disk space to store this file, but the 51GB of DEM files that you downloaded may be deleted after this step.

`Analyze`. After running the Pack program, move the resulting packed binary data file `UtahDEMData.bin` from the `Pack` folder to the `Analyze` folder, then compile and run the Analyze program. The results will be placed in `output.txt`. An Excel spreadsheet called `utah20x20.xlsx` contains our version of `output.txt` and the resulting graph shown in Figure 5.

Links to Doxygen-generated documentation of the source code can be found at `http://larc.unt.edu/ian/research/tobler/`.

Author Contact Information

Ian Parberry
Dept. of Computer Science and Engineering
University of North Texas
1155 Union Circle #311366
Denton, Texas 76203–5017

http://larc.unt.edu/ian

Ian Parberry, Modeling Real-World Terrain with Exponentially Distributed Noise, *Journal of Computer Graphics Techniques (JCGT)*, vol. 4, no. 2, 1–9, 2015
http://jcgt.org/published/0004/02/01/

Received:	2015-02-04		
Recommended:	2015-03-31	Corresponding Editor:	Oliver Wang
Published:	2015-05-05	Editor-in-Chief:	Morgan McGuire