Preprint of a paper published in

Computers & Graphics

DOI: 10.1016/j.cag.2021.06.012

Example-based terrain synthesis with pit removal

Joshua J. Scott[a], Neil A. Dodgson[a,*]

[a] *Victoria University of Wellington, Wellington, New Zealand*

ARTICLE INFO

Article history:
Received 19 March 2021
Revised 25 June 2021

Keywords: example-based, optimization, terrain

ABSTRACT

Artificial terrain synthesis is challenging because natural terrain contains many types of features generated by a diverse range of natural processes over vastly different timescales. Example-based terrain synthesis has been used to overcome the challenge by using real-world terrain as the basis for synthesis. This produces good local behaviour but can have poor global behaviour. We introduce fluid-flow solutions that can be overlaid on multi-resolution example-based terrain synthesis to improve global realism. We apply these to an existing pixel-based terrain synthesis method and to a novel terrain synthesis method, *terrain optimisation*, based on texture optimisation. Our fluid-flow solutions improve results for artificial terrains.

1. Introduction

The creation of realistic artificial terrain has been a challenge for computer graphics for decades. Real terrain is generated by a range of complex physical processes over millions of years and it is a substantial challenge to reproduce plausible virtual terrain. Realism refers to the terrain's adherence to the physical rules of the world and it is an important aspect of terrain synthesis. We are particularly interested in generating terrain that could be judged to have been formed by real-world processes.

We concentrate on example-based terrain synthesis, where the generated terrain is drawn from real-world examples. Using real-world data tends to give good local behaviour, because the local features are true terrain. However, it gives poor global behaviour, because the methods stitch together pieces of terrain from different parts of the exemplar. To improve the overall realism, we add a global process, "pit removal", that is well-understood as a pre-process for Geographic Information Systems (GIS) algorithms. We demonstrate that pit removal is not appropriate as a *post*-process for terrain generation and that it must instead be applied at every level in a multi-resolution terrain generation process. Our results show that pit removal improves results significantly when the guiding input does not explicitly address drainage patterns.

As part of this work, we also investigated a range of ways to adapt the method of texture optimisation to terrain generation. We report the best of our investigations here (the less successful methods are described in Scott's thesis [1]). The best algorithm, which incorporates pit removal, produces results that are different in character to other example-based approaches. Our experimental participants judged it better than other methods in certain cases.

We make the following contributions:

1. a novel example-based multi-resolution terrain synthesis algorithm, *terrain optimisation*, based on texture optimisation (Section 4);
2. application of three pit-removal algorithms (Section 3) to two example-based terrain multi-resolution generation algorithms: our novel terrain optimisation algorithm (Section 4.4) and Gain et al.'s [2] pixel-based algorithm (Section 5);

In addition, we summarise relevant prior work (Section 2), present example output, and summarise the results (Section 6) of a comparative study, that is reported fully elsewhere [1, 3], comparing two state-of-the-art example based methods against our two algorithms that incorporate pit-removal.

*Corresponding author
e-mail: neil.dodgson@vuw.ac.nz (Neil Dodgson)

2. Related work

There are three broad categories of terrain synthesis algorithms: *procedural* modelling, *physically-based* simulation, and *example-based* methods. These are covered in detail in Galin et al's 2019 review [4] and Scott's thesis [1].

2.1. Procedural terrain synthesis

Research in terrain synthesis initially focused on developing *procedural* algorithms that approximate the shape of terrain, inspired by Mandelbrot's early research that models natural phenomena using fractals [5, 6, 7]. Other procedural approaches followed, including the use of alternative fractal generation techniques [8, 9, 10, 11], distance-based functions [12, 13, 14], and sketching methods [15, 16, 17, 18]. These algorithms were designed to be fast and were especially popular due to the lack of computing power at the time. While the later procedural methods allowed a greater degree of control, overall the terrain produced is not realistic as there is little to no consideration of the natural phenomena that shape terrain in the real world. Sketching approaches, in particular, are so unconstrained that they rely almost entirely on the user's knowledge of physical geography to synthesise any kind of realistic structure.

2.2. Physically-based simulation for terrain synthesis

A second approach is *physically-based simulation*, using models derived from physical geography [19, 20, 21]. Some methods in this category synthesise terrain by running simulations at interactive rates [22, 23, 24, 25], and others through the use of high level evaluations [26, 27]. While these methods produced more realistic terrain than the procedural methods, the use of simplified models limits the realism of the terrain that they can produce. For instance, most approaches rely on models of hydraulic and thermal (diffusive) erosion, ignoring other influential forces that shape terrain including glaciers, earthquakes, weather patterns, and animal and human interference. Control is also an issue, as the user can often change only the parameters of the models and cannot directly specify the shape of desired terrain. As a result only a limited range of types of terrain can be synthesised and the outputs are often unrealistic and hard to control.

2.3. Example-based terrain synthesis

The third approach is *example-based*, in which terrain is synthesised from existing data. These methods were developed as more real-world elevation data became available [28]. The expectation is that the use of real-world examples will lead to output that is as realistic as the examples. The availability of data is growing constantly. For instance, 1-arc second digital elevation data is readily available for the majority of the Earth's surface [29] with some areas scanned up to 1 meter resolution [30].

Most example-based methods use techniques from texture synthesis algorithms [2, 31, 32, 33, 34, 35, 36] while others use artificial intelligence techniques [37, 38]. However, while the local topography is realistic, owing to the real-world data provided, the global structure is less realistic than in physically-based simulation methods because there is no consideration of natural processes. Our work is inspired by the challenge that these methods generally produce numerous 'pits' (areas with no outflow to external bodies of water) in a single synthesis, with those pits being of various sizes, whereas it is rare to have even a single substantial pit in a section of real-world terrain.

Our work focuses on whether we can increase the level of realism in example-based approaches in two ways: (1) by investigating a novel example-based approach, *terrain optimisation* (Section 4) and (2) by adding a simulation-based global fix (Section 3) to example-based results (Sections 4.4 and 5).

Galin et al. [4] and Scott [1] discuss the full range of example-based terrain synthesis methods. The three state-of-the-art example-based terrain synthesis methods are those of Zhou et al. [32], Tasse et al. [33], and Gain et al. [2]. All build on *texture* synthesis. Wei et al. [39] classify texture synthesis into three main approaches: patch-based, pixel-based, and texture optimisation. The following summarises the three approaches and how they are applied to terrain synthesis; more detail is provided by Scott [1].

2.3.1. Patch-based terrain synthesis

Patch-based *texture* synthesis [40, 41, 42] creates a new texture by incrementally adding patches from the example and blending or carving the overlapping regions to fit it into the synthesis. Brosz et al. [31] used example height map data to synthesize fine detail for patch-based *terrain* synthesis. The user provides an example height map with fine detail features and a coarse height map with no fine detail as the target. Small scale features are extracted from the example data by matching patches from the target height map to the example height map and taking the relative difference in elevation as the fine detail. This detail is then transferred to the target height map using Efros and Leung's image quilting [40]. Zhou et al. [32] extended this idea, using an improved method of image quilting by Wu and Yu [42]. The user supplies example data and a rough sketch of the ridges and valleys as the target output. Patches are extracted from the example height map along the ridges and valleys and stitched together using Poisson image stitching [43] against the corresponding ridges and valleys in the target. Patches are then placed to fill out the rest of the image in a raster scan order until there are no pixels unfilled. This work was further extended by Tasse et al. [33], who provided a more efficient algorithm for ridge identification, an alternative patch stitching method using Shepard interpolation [44] and a GPU implementation for finding the best candidate patches. Their algorithm ran much faster and produced fewer artifacts between stitched patches [1]. We use Tasse et al. as one of our comparators (Section 6).

2.3.2. Pixel-based terrain synthesis

Pixel-based *texture* synthesis creates a texture pixel by pixel [45, 46, 47, 48, 49]. A value for a synthesised pixel is selected from the example texture that minimises the difference between its neighbourhood in the example texture and the neighbourhood in the current synthesis. Pixel-based methods use Markov random field (MRF) theory which models the texture as a realisation of a local and stationary process.

Dachsbacher's pixel-based *terrain* synthesis [50] uses non-parametric sampling, which is adapted from Efros and Leung's pixel-based texture synthesis [45]. For a given input that is partially filled with data, the algorithm grows outward from the set data, pixel-by-pixel. A more successful adaptation, of pixel-based texture synthesis methods to terrain, was made by Gain et al. [2], which is based primarily on the pixel-based methods of Lefebvre and Hoppe [51] and Han et al. [49]. The user defines point, curve, type, and copy-paste constraints that control the height and shape of the terrain in certain areas and the method performs a parallel hierarchical texture synthesis with example height map data to produce the final result. Implementation on the GPU gives generation times below 100 milliseconds, allowing interactive editing. We build on Gain et al's method in our work (Section 5).

2.3.3. Texture optimisation for terrain synthesis

Texture-optimisation [52, 53, 54, 55] is a technique for texture manipulation whose applications include synthesis, reshuffling, hole-filling and interpolation. The process of synthesis is formulated as a minimisation of an MRF-based similarity metric. This is performed using an expectation maximisation (EM)-like algorithm where the algorithm iterates between finding a correspondence between the synthesis and the example-texture (maximisation M-step) and optimising the synthesis using the correspondence (expectation E-step). No previous terrain synthesis methods have built on the texture optimisation approach and we tackle this here, creating a new method that we call *terrain optimisation* (Section 4).

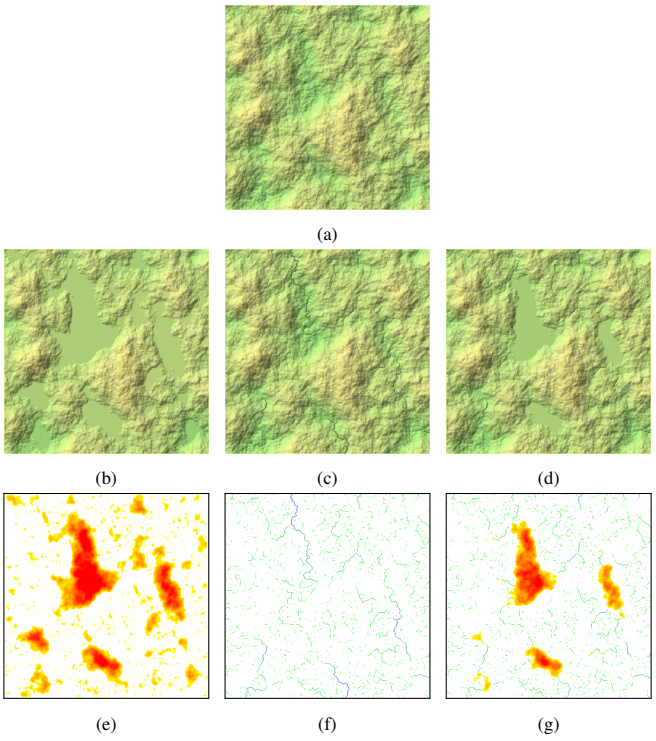


Fig. 1: Examples of pit removal. (a) A terrain height map synthesised using Perlin noise. (b–d) Output from the three algorithms: (b) depression filling, (c) depression breaching, (d) hybrid; (e–g) difference between the source and the output above, where the yellow–red scale shows the *increase* in elevation from minimal to substantial, the green–blue scale shows the *decrease* in elevation from minimal to substantial. Note: decreases in elevation are paths that are a single pixel wide; zoom in to view details.

3. Pit removal

The primary downside to current example-based methods that are based on texture synthesis is the lack of consideration for the global structure of the synthesised terrain. These methods rely on the user providing the appropriate constraints to ensure that the structure of the terrain is realistic, and make no effort to ensure that non-expert users can synthesise realistic terrain.

The surface flow of water plays a large role in shaping the topography of landscapes. Areas of terrain where the surface flow of water does not drain into an external water body are generally known as *depressions* [56]. For our purposes these fall into two groups: large-scale *endorheic basins* [57] and small-scale *pits* [58]. Endorheic basins are caused by tectonic depression and evaporation occurring more rapidly than sedimentation, on a geological timescale. An endorheic basin usually has a landlocked lake at its heart. Pits are smaller-scale features that have no outlet and no water body at their lowest point. They rarely appear naturally owing to hydraulic erosion and gravity-driven sediment transport, where the matter is eroded and transported down-slope until it reaches a sink (e.g., the ocean or a landlocked lake). This natural process fills in existing basins and prevents new basins from being created, as matter would have to be transported up-slope against gravity. There are occasional cases where pits occur naturally owing to underground transport networks or where recently created pits such as sinkholes and craters have not yet been filled through the erosion process. It is also the case that flat landscapes (such as the Laurentian shield in the northern US and eastern Canada) can have substantial areas that are endorheic without necessarily having a clear body of water at their heart [59]. From these observations of the natural world we make the reasonable assumption that, when synthesising terrain that is other than flat, the synthesised terrain will appear more realistic if it has fewer pits rather than more pits, and that an algorithm could produce more realistic terrain if it minimised the number of pits it produced in its synthesised output.

Pit-removal algorithms are used in Geographic Information Systems (GIS) for processing elevation data to ensure that there is always a down-sloping path from any point to a sink. A terrain height map T has no pits if, for every pixel $p_0 \in T$, there exists a connected path of adjacent pixels $(p_0, ..., p_n)$ where $p_n \in S$ for a set of sink pixels S, and $T(p_i) > T(p_{i+1})$. The sink set S is typically the set of pixels on the border of the height map but can also include internal bodies of water such as landlocked lakes or sinkholes that the user wishes to have in the output terrain.

From a GIS perspective, while pits could be the result of natural terrain features (e.g., sinkholes), they are generally considered to be erroneous data [60] likely to be caused by unnatural terrain features (e.g., bridges, embankments, dams) or artificial data (e.g., random noise, data collection artefacts). The reason

for applying pit-removal as a pre-process is that certain GIS algorithms fail in the presence of pits in the input data. The algorithms are thus not designed to produce realistic results, rather they are designed to remove pits with as few changes as possible. Careful consideration needs to be taken to ensure that, when adapting these methods to remove pits for terrain synthesis methods, in an attempt to increase realism, the overall realism is not reduced due to unintentional features, including flat-planes (Figure 1b,e) or deep and thin valleys (Figure 1c,f).

We investigate three pit-removal algorithms from GIS [58, 61, 62] and experiment with ways in which they can be incorporated into multi-resolution example-based terrain synthesis, both into the existing pixel-based synthesis algorithms of Gain et al. [2] (Section 5) and into our own novel terrain optimisation method (Section 4 and 4.4).

The three main approaches for pit removal are depression filling, depression breaching, and hybrid methods.

3.1. Depression filling

Depression-filling algorithms remove pits by increasing the elevations inside the pit region to ensure that all elevations have a down-slope to a sink. We implemented the "Improved Priority-Flood" algorithm [58], which performs depression filling taking into account flow along flat areas. An example of the algorithm's output can be seen in Figure 1b.

Depression filling processes each point of elevation only once, which makes it ideal for large height maps used in GIS. It works well for pits with a small area (as in real-world datasets) and regions that are completely flat that need a flow direction, as there is only a small change in elevation values and the effects are barely noticeable. However it does not work well for pits with a large area as the result is an undesirable flat plane that spans the area of the pit (Figure 1b,e). Filling is still used in GIS, despite evidence that breaching and hybrid methods are more aligned with the causes of depression artefacts in captured data and affect the overall height map to a lesser extent [62, p.3].

3.2. Depression breaching

Depression breaching (also known as carving) algorithms decrease the elevations around pit regions to ensure that all elevations have a down-slope to a sink. We implemented Soille's algorithm [61], which determines the path of the channel by the flood order of the height map. An example is shown in Figure 1c.

Our results (see later) shows that depression breaching is the most successful of the three approaches, so we summarise our algorithm here (the details of the depression-filling and hybrid algorithms can be found in Scott's thesis [1]).

The breach-algorithm takes, as input, a height map and set of pixels S as the sinks. The breach-algorithm uses an open queue (a priority queue with ascending order of elevation), a closed set (a plain set), and a flow map F (a map from pixel to pixel). The open queue acts as a list of pixels that still needs to be processed in an ascending order of elevation, which simulates the flooding process from the specified sinks. The closed set keeps track of pixels to ensure that they are not processed more than once. The flow map is a map of neighbouring pixels that represents the direction that a pixel should flow downhill. It is used to produce a path from any pixel to a point of lower elevation, or a sink pixel. First, the open queue and the closed set are initialized with the sink pixels. Next, the fill-algorithm polls a pixel p from the open queue and iterates over each neighbouring pixel p_n. If the neighbour pixel p_n is already in the closed set it is ignored, otherwise it is added to the open queue, closed set, and flow map ($F(p_n) = p$). If the neighbour pixel p_n has a lower elevation than the pixel p ($T(p_n) < T(p)$), the process of breaching occurs. Starting with the pixel $p_0 = p$, the breach-algorithm creates a path $P = (p_0, ..., p_k)$, where each sequential pixel in the path is found using the flow map $p_{i+1} = F(p_i)$ and this continues until an elevation lower than $T(p_n)$, meaning $T(p_i \in P) \geq T(p_n)$, or a sink pixel is reached, meaning $p_k \in S$. The elevation of each pixel in this path $p_i \in P$ is then lowered to the next representable value below the neighbour pixel $T(p_n)$ and the previous pixel in the path $T(p_{i-1})$. This process is repeated until the open queue is empty.

In GIS, breaching is generally regarded as better than filling [62, p.3] but is slightly less efficient owing to backtracking when carving channels (that is, a small proportion of pixels are visited more than once, owing to backtracking, as illustrated by the non-white pixels in Figure 1f). Breaching algorithms are well suited for shallow pits that cover a large area, as the breaching channel alters a minimal number of pixels to remove the pit. However deep pits cause drastic modification to elevation levels when breaching, so that the resulting channel incises the surrounding terrain deeply (indicated by the dark blue in Figure 1f). These deep pixel-wide channels are undesirable in many cases, especially when the channels are carved from pits created by noisy data, producing an appearance of a river network where there is none. There are other approaches to breaching, such as the least-cost-path method by Lindsay and Dhun [63], but they are developed for more specific reasons, like breaching on fine-resolution height maps of heavily altered landscapes, and are unsuitable for the general case of terrain synthesis.

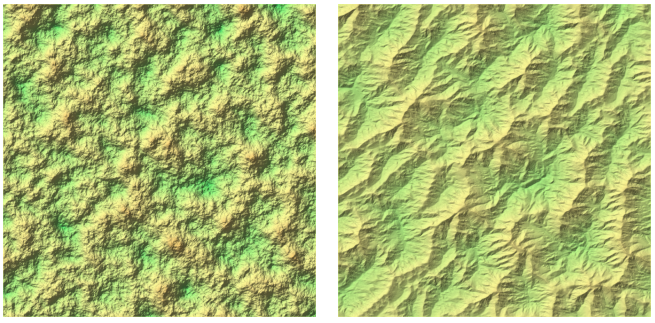
3.3. Hybrid method

Hybrid methods combine the techniques of filling and breaching to remove pits with the intention of minimizing the downsides of each. We implemented the selective breaching mode for the hybrid approach by Lindsay [62], which breaches channels if the cost is lower than the user-defined thresholds and fills the rest of the pits. An example is shown in Figure 1d.

In general, hybrid methods reduce the area affected by depression filling, causing smaller flat regions, and incise less deeply (compare Figures 1e and 1g). However, it is necessary to tune two additional parameters, the maximum channel depth and the maximum channel length, to achieve the best results for the pit-removal for a given terrain. This parameter tuning requires an understanding of the trade-offs involved for each individual case.

3.4. Summary

All three algorithms are geared towards removing pits for GIS applications, such as surface flow simulations. As is clear



(a) (b)

Fig. 2: An example of the terrain optimisation algorithm. (a) Perlin noise used as the initial guide terrain G. (b) Terrain synthesised using the core terrain optimisation algorithm.

from the large flat areas and deep pixel-wide chasms in Figure 1, they are not designed to make the terrain more realistic nor are they particularly suitable for terrain that contains substantial large pits. They are therefore not suitable as a stand-alone *post*-process for example-based terrain generation. However, we demonstrate below that they can solve the challenge of pits in terrain synthesis if adapted to a *multi-resolution* example-based approach, where the pit removal is applied at *each resolution* during generation. We apply these algorithms and ideas to our novel example-based multi-resolution terrain optimisation method (Section 4.4) and then to Gain et al's pixel-based method (Section 5).

4. A terrain optimisation algorithm

Texture optimisation has not previously been considered for terrain generation. Our novel *terrain optimisation* algorithm follows the texture optimisation approach of Kwatra et al. [53], updated to incorporate Generalised PatchMatch [64], with other minor modifications as noted below, and with pit removal added.

4.1. Core terrain optimisation algorithm

The user provides, as input, a source terrain height map (source map) and an initial estimation of elevations for the desired height map (guide map). The source map, S, provides the real terrain features that will be used to synthesise a new terrain height map, and the guide map, G, is used to guide the algorithm to produce terrain in line with artistic direction. The final synthesised terrain, R, is a reconstruction of patches extracted from the source map, S. Figure 2 shows an example of the core algorithm, taking Perlin noise as its guide map, G, and converting this to a more realistic terrain, R. Our algorithm closely follows Kwatra et al. [53]; we summarise it here.

First, the algorithm creates a Gaussian image pyramid of depth L, for both the source map, S^k, and the guide map G^k, and a correspondence map, C^k, for each pyramid level k that maps the patches of the current synthesis R^k to patches in source map S^k. The highest resolution image in the Gaussian pyramid is at level $k = 0$ and the lowest at level $k = L$. The lowest resolution level in the Gaussian pyramid for the guide map G^L is used as the initialisation for the synthesis R^L. The correspondence map C^k maps patches $\boldsymbol{r}_p$ (of size $N \times N$) centred on every pixel $p \in R^k$ to a patch $\boldsymbol{s}_p$ in the corresponding source map S^k. The values stored in this map are transforms, where the correspondence map contains a mapping between the centre pixel p of $\boldsymbol{r}_p$ and the coordinate $[x, y]$ of the centre of $\boldsymbol{s}_p$, such that $C^k(p) = [x, y]$. The values in the correspondence map are initialised by finding the best patches in S^L that match the current synthesis P^L, in the same way as the M-step described below.

Second, after initialisation, starting at the lowest level, $k = L$, of the Gaussian pyramids, the algorithm begins the process of optimisation and up-sampling. Optimisation is an iterative process, minimising the energy function of the synthesis R^k (with respect to S^k) and uses an EM-like approach. The energy of a synthesised terrain is the sum of the SSD between each patch $\boldsymbol{r}_p$ in the synthesis R^k and its closest patch $\boldsymbol{s}_p$ in the source map S^k

$$E_t(R^k) = \sum_{p \in R^k} \left\| \boldsymbol{r}_p - \boldsymbol{s}_p \right\|^2 . \tag{1}$$

The E-step calculates the new values of the synthesis R^L from the patches in the current correspondence map C^k, reconstructing the synthesis with patches from the source map. The new value for every pixel in the synthesis is the mean of values from the overlapping patches $\boldsymbol{s}_p$

$$R^k(p) = \sum_{q \in Q} \frac{\boldsymbol{s}_{p+q}(n_b - q)}{N^2}, \tag{2}$$

for each pixel $p \in P^k$, where $Q = [n_a, ..., n_b]^2$ are the coordinate offsets for each patch, N is the square patch size, and $n_a = \left\lfloor \frac{-N}{2} \right\rfloor$ and $n_b = \left\lfloor \frac{N-1}{2} \right\rfloor$ are the patch offsets on opposite sides of the patch centre.

The M-step finds new patches $\boldsymbol{s}_p \in S^k$ that best match each patch $\boldsymbol{r}_p$ in the current synthesis, that minimise the texture energy function. Rather than the tree-based method used by Kwatra et al., we use the PatchMatch algorithm [54], which more efficiently finds a corresponding patch that approximately minimises the energy function. The correspondence map is updated with the new mapping of $\boldsymbol{r}_p$ to its closest match $\boldsymbol{s}_p$ found using PatchMatch, such that

$$C^k(p) = \arg\min_{\boldsymbol{s}_p} \left(\left\| \boldsymbol{r}_p - \boldsymbol{s}_p \right\| \right). \tag{3}$$

This EM-like optimisation process is computed μ times (we use $\mu = 2$, as greater values of μ provide diminishing returns) before the correspondence map C^k is up-sampled to the next level C^{k-1}. The values in the up-sampled correspondence map are computed as

$$C^{k-1}(p) = 2C^k \left(\left\lfloor \frac{p}{2} \right\rfloor \right) + \left(p - 2 \left\lfloor \frac{p}{2} \right\rfloor \right). \tag{4}$$

After the correspondence map has been up-sampled, the optimization and up-sampling process repeats on level $k - 1$. The optimization and up-sampling is repeated until the optimization at $k = 0$ is complete, and the final output is the reconstructed synthesized terrain P^0.

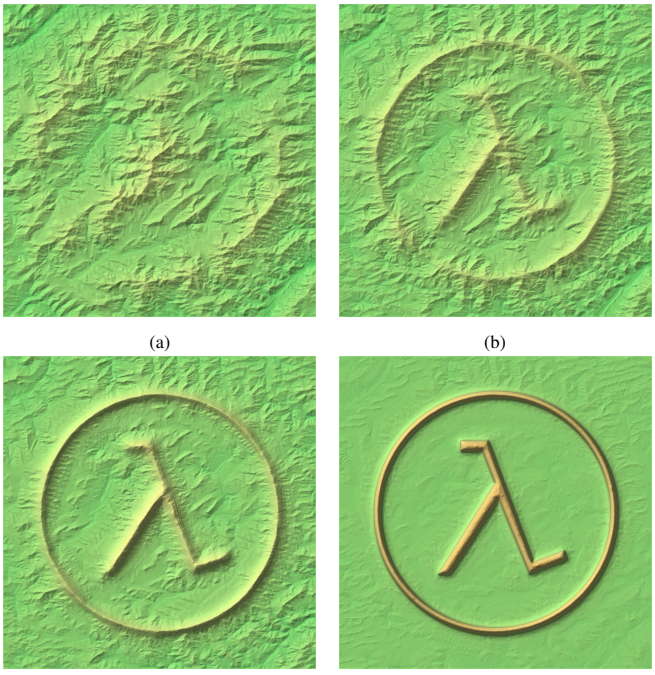


Fig. 3: This highly artificial constraint clearly demonstrates the effects of different soft weight functions w. (a) No weights, $w = 0$ (no soft constraints, initialisation only). (b) Exponential weighting, $w = \alpha\frac{2^k-1}{2^L-1}$. (c) Linear weighting, $w = \alpha\frac{k}{L}$. (d) Constant weighting $w = \alpha$. We use exponential weighting (b).

4.2. Constraints

Kwatra et al's texture optimisation algorithm has three types of constraints that can be used for artistic direction: soft, hard, and type. We use soft constraints. Hard and type constraints are discussed by Scott [1].

Our soft constraints are similar to those proposed by Kwatra et al. [53]. They guide the optimisation process to synthesise values closer to, but not exactly the same as, the constrained values.

To use soft constraints, the guide map provided by the user becomes the soft-constraint guide for each pixel. More specifically, the guide map G^k is used as the soft-constraint for the synthesis R^k at each pyramid level k. We found it sufficient to implement soft-constraints by combining the constrained pixel values in the soft constraints with the reconstruction in a weighted sum, replacing Equation (2) such that

$$R^k(p) = wG^k(p) + (1-w)\sum_{q\in Q}\frac{\boldsymbol{s}_{p+q}(n_b - q)}{N^2}, \qquad (5)$$

where $w = \alpha\frac{2^k-1}{2^L-1}$. The α term controls the maximum blending weight at the lowest resolution in the pyramid, level L.

This weighting term w ensures that the effect of the soft constraints is halved at each successive level of the synthesis. At low resolutions (where k is larger) the soft constraints drive the general shape of the terrain, ensuring that large scale structures are consistent with the artistic direction. At high resolutions (where k is smaller), the large scale structures are inherited from the lower resolutions and the terrain details are almost entirely driven by reconstructing patches from the source map, ensuring a greater degree of realism while being consistent with the general layout of the artistic direction.

We experimented with different weighting functions (Figure 3). Constant weighting $w = \alpha$, and linear weighting $w = \alpha\frac{k}{L}$ influence the synthesis too much at higher resolutions and reduce overall realism. Zero weighting only influences the lowest level initialisation, so the final result is dominated by the terrain structure from the source map. Exponential weighting gives a good balance between the constraint and the structure from the source map.

Figures 8c,f,i,l and 10a,b,c show further examples of the algorithm's output with the addition of an expanded search space (Section 4.3) and pit removal (Section 4.4).

4.3. Expanding the PatchMatch search space

Our core terrain optimisation algorithm uses integer coordinates and a single orientation for the patches in the source map. This limits the use of the real-world data provided because it restricts the algorithm to only translations when more realistic sections may exist in the source map but the algorithm cannot transform them appropriately to be used in the synthesis. For example, rotation of a section of the source may produce better alignment than any translation or modifying the overall height of a section of the source may produce an improved result [65]. This limitation can be overcome by extending the correspondence search to use the Generalised PatchMatch algorithm [64].

The question is: which transformations are appropriate in the context of terrain? For example, Generalised PatchMatch searches over a range of rotations θ and scales s, extending the search space of the original PatchMatch algorithm from (x, y) to (x, y, θ, s). However, scaling is not appropriate for transferring terrain features, as it undermines the physical attributes of terrain and thus the realism. However, the same principles can be used to search for reflection transformations. We further experimented with extending the algorithm to calculate a height-offset that minimises the SSD for a possible candidate, which is stored as part of the final transform. This is similar to the gain and bias adjustments used in Image Melding texture synthesis [55]. Our final implementation is a PatchMatch algorithm that includes a search space (x, y, θ, r, h), of translation (x, y), rotation θ, reflection r, and height-offset h.

Our experiments [1] demonstrated that the continuous translation extension slightly reduces the mean energy while giving a visually similar result. Rotation and reflection have the challenge that, while dramatically increasing the search space, they can cause loss of any alignment of the features in the example map. For example, some terrains have strongly directional features, such as parallel valleys, and it may or may not be appropriate to rotate such features depending on the desired form of the output. The user needs to make an artistic judgement whether or not this is desirable and hence whether to employ these two extensions. Our experiments [1] show that our height-offset extension should not be used. Although this lowers the mean energy function, it cannot guarantee that the resulting synthesis is realistic with respect to the example map because there are features of terrain that cannot be raised or lowered significantly without losing their realism.

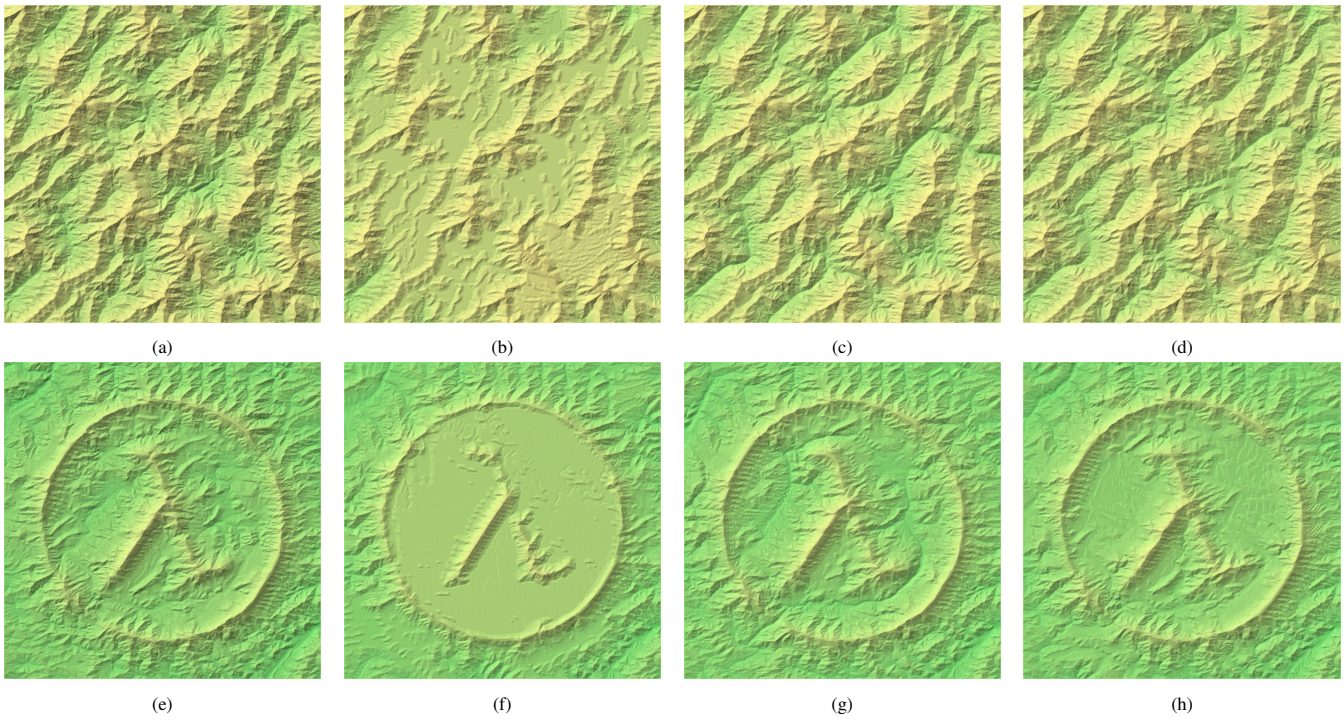


Fig. 4: Comparison between the core terrain optimisation algorithm and the results of pit-removal extension, using the three different pit-removal algorithms. Top row: a pseudo-realistic terrain driven by Perlin noise. Bottom row: a highly artificial terrain driven by a lambda in a circle. (a),(e) The core terrain optimisation algorithm. (b),(f) Depression filling. (c),(g) Depression breaching. (d),(h) Hybrid algorithm.

4.4. Pit removal applied to terrain optimisation

Similar to other example-based texture-synthesis methods, our new terrain optimisation algorithm requires user guidance to ensure that a realistic structure is synthesised. As such, the core algorithm produces terrain that contains many unwanted pits, reducing the overall realism of the result. We applied the three pit-removal algorithms to the core terrain optimisation algorithm to ascertain which would be most suitable.

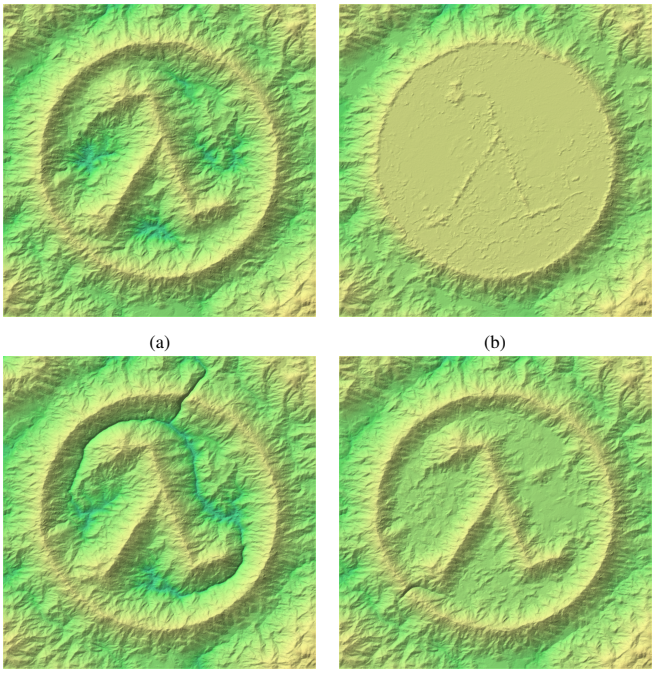
The core terrain optimisation algorithm is modified as follows. During the optimisation process, after the E-step when the synthesis is reconstructed, a pit-removal algorithm is used to ensure there are no pits. This process is repeated for every iteration of optimisation except at the highest level of synthesis $k = 0$, to ensure that no pixel-wide artefacts from the pit-removal are created after the last iteration of the optimisation.

Examples of the pit-removal application are shown in Figure 4. Depression filling produces unrealistic flat regions. This is obvious in the highly artificial lambda example (4f), where the entire central region becomes a high plateau, but it is also clearly visible in the more realistic example (4b), where it creates a number of unrealistic flat plains. Depression breaching produces V-shaped valleys owing to the use of the pixel-wide algorithm at every level of the multi-resolution generation. There is little to choose between depression breaching and the hybrid method in the more realistic case (4c,d), but in the highly artificial case we see that the hybrid method (4h) produces less convincing results than the breaching method (4g) owing to the increase in undesirable flat regions. On this evidence, we therefore choose to apply depression breaching when testing our terrain optimisation approach against other example-based methods (Section 6).

5. Pit removal applied to pixel-based synthesis

To provide a suitable comparator for our novel terrain optimisation algorithm, we want an existing example-based terrain synthesis algorithm that incorporates pit removal. To this end, we adapted the state-of-the-art pixel-based terrain generation of Gain et al. [2] to incorporate pit removal. We investigated several ways to apply pit removal to their algorithm and show the most successful approach here. Other approaches are discussed by Scott [1].

We developed our "Gain-pit" algorithm by adding height-offset modification to Gain et al's algorithm [2]. Gain et al. use an offset map to drive multi-resolution synthesis. We adapted this algorithm so that, during the correction phase of the algorithm, at each pyramid level, an extra step is taken to remove pits for the current synthesised terrain. After each iteration of correction, where candidates are selected to replace values in the synthesis and offset map, the terrain is reconstructed using the coordinates in the synthesis to sample the elevations from the example map and adding the height offsets from the offset map. We then apply one of the pit-removal algorithms (Section 3) to remove the pits from the reconstruction R to produce a new pit-free height map R'. The difference between the reconstruction and the pit-free height map, $R' - R$, is added to the current offset map. This ensures that, if Gain-pit reconstructs the synthesis using the modified offset map, it will result in the same pit-free height map R'.



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

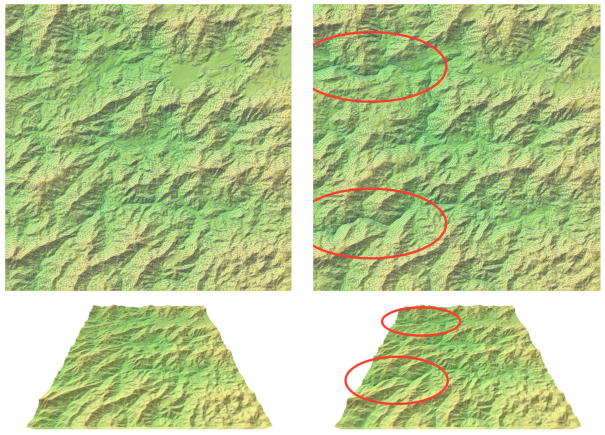
Fig. 5: A comparison between the original Gain algorithm and the results of the Gain-pit algorithm using the three different pit-removal algorithms on a highly artificial construct. (a) The original Gain algorithm. (b) Gain-pit using depression filling. (c) Gain-pit using depression breaching. (d) Gain-pit using the hybrid algorithm.

We ran the algorithm on several terrains with all three pit-removal algorithms. The highly artificial terrain in Figure 5 is chosen from our examples because it most clearly shows the differences. Depression filling is inappropriate because it destroys the desired features and creates an unnatural flat plane. Breaching is clearly superior to filling. Hybrid is similar to breaching but suffers from flat-plane artefacts.

Pit-removal is run many times during the synthesis because the correction process introduces new pits in each iteration. This compounds any flat regions produced by depression filling or the hybrid algorithm, and it produces an overall rise in the centre of the terrain relative to the sides. Because the correction and de-pitting are executed at all resolutions, small flat regions, created by filling the pits at low resolution, are enlarged during up-sampling. While flat regions may be adjusted through the correction process, our experiments found that the correction process is insufficient to prevent the generation of terrain that does not exist in the example. We therefore chose to use depression breaching, rather than the hybrid approach, as our method of choice in "Gain-pit".

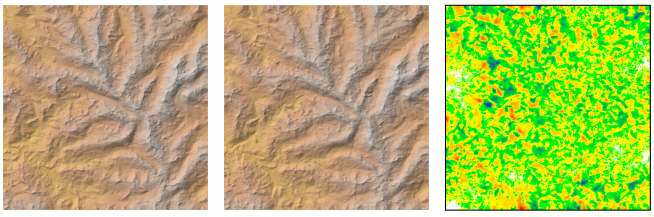
6. Results

We have created two novel methods that include pit removal through depression breaching in a multi-resolution algorithm. These are our novel terrain optimisation method (Section 4) and our modification of Gain's algorithm to create our "Gain-pit" algorithm (Section 5).



(a) Gain (b) Gain-pit

Fig. 6: A comparison between the original Gain algorithm and the results of the Gain-pit on an artificial terrain. Notice the improved flow on the two main valleys that disgorge on the left edge of the terrain



(a) (b) (c)

Fig. 7: A comparison between the original Gain algorithm and the results of the Gain-pit for constraints that imitate real terrain. (a) The original Gain algorithm (b) Gain-pit using the breach-algorithm. (c) Difference between the two images, where the yellow-red scale and green-blue shows the increase and decrease in elevation respectively, from minimal to substantial.

6.1. Implementation details

For terrain optimisation, our un-optimised CPU implementation, written in C++, takes approximately five minutes to synthesise a 1000×1000 pixel terrain on a machine with an Intel Core i5 3570K (3.4GHz) and 8GB of DDR3 1600MHz memory. In this implementation the largest performance bottleneck is the PatchMatch algorithm which takes approximately 95% of the total synthesis time. While not an issue for our experimental work, in a commercial environment this can be addressed by optimising PatchMatch with a parallel implementation on CPU or GPU [66, 67].

Our implementation of Gain's algorithm, in C++, runs in approximately two minutes on the same machine. The addition of pit removal to create the Gain-pit algorithm does not change the run time significantly. That is, the pit-removal algorithm adds less than a second of compute time for a 1000×1000 terrain. Pit removal takes similar time (i.e., less than a second) in the terrain optimisation algorithm.

Although the pit removal adds no significant extra cost to run time, since completing our experiments new algorithms for removing depressions have been published that would bear investigation, as they purport to be more efficient [56, 68].

6.2. Artefacts of depression breaching

For both terrain optimisation and Gain-pit, we choose to use depression breaching as the best of the three pit-removal methods. In Section 3 we observed that depression breaching produces unrealistic single-pixel wide valleys in its single-pass implementation (Figure 1c,f). The multi-resolution nature of both terrain optimisation and Gain-pit convert this behaviour into steep-sided V-shaped valleys (Figures 4c,g, 5c, 6b). The single-pixel wide carving can still cause unrealistic results in the final layer of synthesis, which we address by not performing the pit-removal in the final layer.

6.3. Gain-pit vs Gain

Compared to the original Gain algorithm, Gain-pit ensures that there are no pits and produced significantly different results in cases where the constraints would otherwise produce substantial pits (e.g., Figures 5 and 6). Consider, however, pit-removal when the constraints are designed so that a "natural" terrain would result (e.g., Figure 7). In these cases, Gain-pit has a small effect on the generated terrain. There is only a slight difference in elevations between the two synthesised terrain images: only a few regions have needed to be lowered significantly due to the addition of breaching. Our experimental assessment (Section 6.5) bears out this informal observation: when constraints are defined that consider drainage patterns, the pit-removal offers limited improvement because the user has explicitly allowed for drainage in their constraints; the benefit of pit-removal comes when the user has not considered drainage in their input constraints.

6.4. Example output

Example outputs of both methods are shown in Figure 8. Three of these examples are generated by giving a real-world exemplar and asking the algorithm to generate a similar terrain, using a different nearby piece of real terrain as the source material [1]. The fourth example is generated using the same highly-artificial construct as used by Zhou et al. [32]. Three further examples of the terrain optimisation algorithm are shown in Figure 10.

To generate these examples, each algorithm needs appropriate input constraints. For the constraints of our terrain optimisation method, we down-sampled the real-world target terrain by a factor of sixteen to a resolution of 62×62 pixels, we then up-sampled this to 1000×1000 pixels, to use as the input for soft-constraints. This emulates the rough input an artist would use to guide the algorithm to produce the correct shape of terrain, without specifying specific details. For the constraints of Gain-pit, we created a program to aid in the specification of the constraints. We traced the ridge-lines and other prominent features of the target terrain, and the program created curve constraints using the traced paths, extracting the correct elevation and gradient values. We manually corrected the area-of-effect for the constraints and other small aspects of the constraints. The final result was a set of curve constraints that matched the target terrain.

Both methods produce *feasible* terrain in *all* examples that we have tried. However, two of the examples shown here demonstrate the type of minor undesirable characteristics that can be produced by the two algorithms, which detract from their *believability*: overly smooth terrain from Gain-pit in Figure 8h and repetitive structures from terrain optimisation in Figure 8l.

6.5. Experimental evaluation

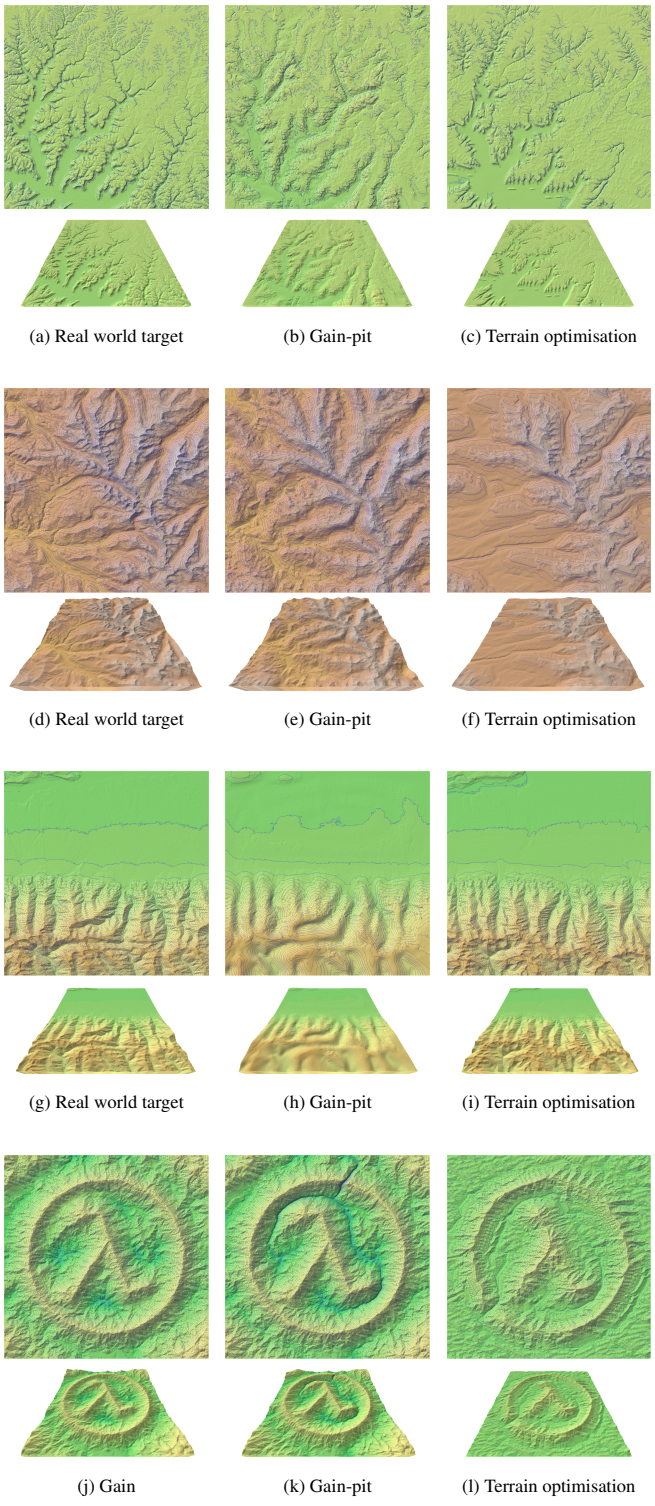
We conducted a substantial experimental evaluation to evaluate the relative realism of our methods on these and other examples. The full details of the experiment are reported elsewhere [3, 1]. Subjects were asked to judge how *believable* each terrain is relative to other methods generating the same terrain and relative to the real-world terrain in cases where there was a real-world equivalent. The intention here is to use subjective *believability* as a proxy for objective *realism*. We summarise the relevant results here.

A total of 245 subjects (of whom 124 identified themselves as experts in physical geography) completed an online survey containing pair-wise comparisons of seven sets of terrain images. Each set contained one terrain height map synthesised by the following methods: Tasse et al. [33]; Gain et al. [2]; "Gain-pit" (Section 5); and terrain optimisation (Section 4.4). In five of the sets, each method was used to reproduce a real-world terrain example and a similar real-world example was included in the pair-wise comparisons (making a total of 10 pair-wise comparisons for each set). For a sixth set, the algorithms were used to reproduce a highly artificial structure in the shape of a lambda surrounded by a ring of mountains. For a seventh set, the algorithms were unconstrained. Note that the five real-world examples (Sets 1–5) are all of terrain where fluvial erosion would be expected to be the main cause of shaping the landscape. Set 3 (Figure 8g–i) and Set 5 (Figure 10(b)) do both have regions that are almost flat but, even here, the expectation is of a smooth shallow slope shaped by erosion and deposition. The guide map for Set 6 (Figure 8j–l) has large flat areas but the source terrain gives the algorithms no source data that is flat and so the algorithms have to fill in this material as best they can, the user's intention being that the algorithm should produce a high circular mountain range surrounded by lower hill country.

Subjects were presented with pairs of terrain images from each set (a total of 62 pair-wise comparisons) in a random order and ask to select the terrain that was "more realistic." On the recommendation of our statistical consultant, the Bradley-Terry model [69] with Holm-Bonferonni correction was used to analyse the data.

The results are summarised in Figure 9 and are explained in detail by Scott [3]. Across the seven example terrains used in the experiment, each of the tested methods generated at least one example that was indistinguishable from real-world terrain in terms of believability at the 99% confidence level (e.g., our new terrain optimisation method ("S") was the most believable in sets 3 and 7). Each method also generated at least one example that was clearly considered to be less realistic than the other methods (e.g, our new method was the least believable in sets 1, 4 and 6).

Looking at the specific examples in Figure 8, the experimental results showed that our new terrain optimisation method was considered indistinguishable from reality in the case of an erosional and depositional landscape along the edge of a mountain



(a) Real world target (b) Gain-pit (c) Terrain optimisation

(d) Real world target (e) Gain-pit (f) Terrain optimisation

(g) Real world target (h) Gain-pit (i) Terrain optimisation

(j) Gain (k) Gain-pit (l) Terrain optimisation

Fig. 8: Four examples of generated terrain from four of the seven sets of images used in our experiment. The middle column is the Gain-pit algorithm (Section 5) and the right column is our terrain optimisation algorithm (Section 4.4), both with depression breaching. The left column is the real-world target for the top three examples and the output of the Gain algorithm for the highly artificial construct in the bottom example. All examples are $30 \times 30\,\mathrm{km}^2$, comprising 1000×1000 pixels. These images are examples of the stimuli used in our perceptual experiment [3].

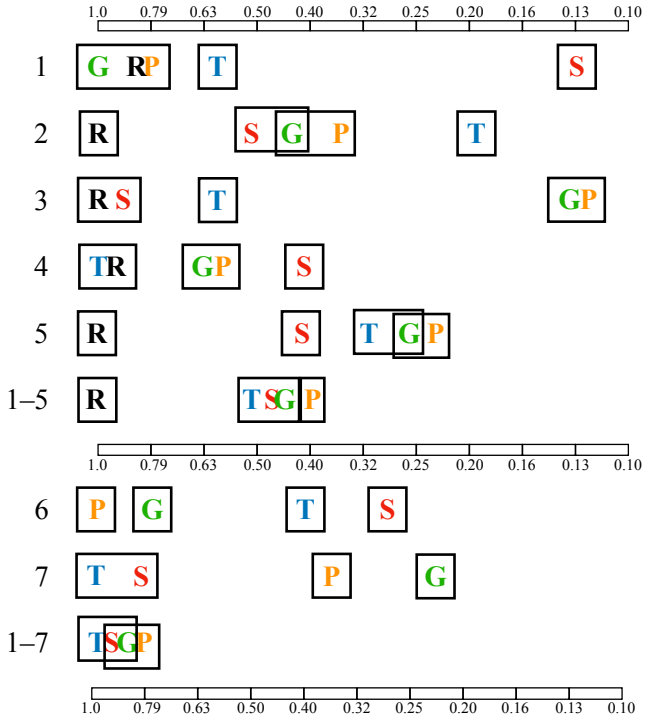


Fig. 9: A visualisation of the results of the Bradley-Terry analysis for the seven tested examples and for combinations of sets 1–5 and sets 1–7. For each set, a letter is placed at the location of the Bradley-Terry parameter, π_i. More realistic results are to the left (higher π_i), less realistic results are to the right. Boxes are put round those letters that are *not* statistically significantly different at the 99% level. All letters within a box can be considered to be equivalent while all letters that do not share a box are statistically significantly different from one another. R=real, T=Tasse, S=terrain optimisation, G=Gain, P=Gain-Pit.

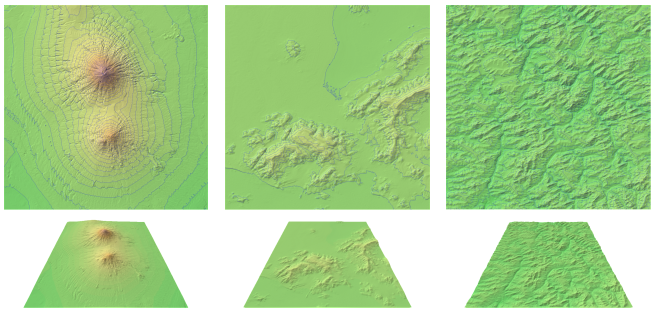
range (set 3, source terrain taken from the Făgăraş Mountains, Figure 8g–i), while Gain and Gain-pit were considered most remote from being realistic.

Our terrain optimisation was the method considered closest to reality in the case of an alpine post-glacial landscape (set 2, source terrain taken from the Rocky Mountains, Figure 8d–f), with Gain being statistically equivalent and Gain-pit ranking only slightly worse.

In the case of an incised plateau with dry riverbeds and a high drainage density (set 1, source terrain taken from the Yemeni desert), Gain and Gain-pit were considered indistinguishable from reality (Figure 8a–c) while our terrain optimisation was considered the least realistic of the four artificial terrains.

For the highly artificial lambda-in-a-circle (set 6, Figure 8j–l), Gain-pit was considered the most realistic result, better than Gain alone, while our terrain optimisation produced what was considered the least realistic result.

Taken over all seven examples, no one method performed better overall in terms of realism. However, some methods clearly have better performance in some circumstances than others. The wide variation in performance across individual sets means that the combination of results into a single overall statistic has limited validity. The subtleties of this are discussed in detail by Scott [1, 3]. Considering the examples in Figure 8, all generated examples could be considered feasible, but there



(a) (b) (c)

Fig. 10: Terrain optimisation examples for the three sets not shown in Figure 8. (a) Set 4: young stratovolcanoes with source terrain from Central Java, Indonesia. (b) Set 5: degraded ancient landscape of inselbergs with source terrain from Ngarutjaranya, Australia. (c) Set 7: Algorithm allowed to run without guide map, with source data from fault bounded mountain ranges in Marlborough, New Zealand.

are clear differences in the *appearance* of terrain generated by terrain optimisation compared with that generated by the Gain algorithms.

From these results, we can conclude that, on average, our new terrain optimisation can perform as well as existing state-of-the-art terrain synthesis methods and that, for certain types of terrain, it can produce superior results. terrain optimisation therefore provides another tool in the armoury of terrain synthesis and provides terrain that has a different character to other approaches.

With regard to whether pit-removal improves the results of the algorithm by Gain et al., the experimental results show that the addition of pit-removal performs statistically identically to the unmodified Gain algorithm when replicating real-world terrain in which drainage patterns are present in the guiding input (sets 1–5), but it outperforms the unmodified algorithm by statistically significant margins in the artificial cases (sets 6 and 7, set 7 is shown in Figure 6), where the Gain-pit algorithm improves the drainage pattern of the generated terrain. Therefore, in situations where the target map has been set up by the user to incorporate drainage, pit removal is of limited benefit; but in situations where substantial pits will result, pit removal has significant benefit in increasing believability in example-based terrain synthesis.

7. Conclusion

Our terrain optimisation algorithm is a novel approach to terrain synthesis based on the texture optimisation approach of Kwatra et al. [53]. On average, its output is comparable in realism to the other state-of-the-art example-based terrain synthesis methods; but it produces terrain that is different in character to other example-based methods so can provide a useful alternative.

Multi-resolution pit removal is a novel approach to improving realism in existing terrain by removing or reducing the presence of endorheic basins, where depression breaching is the most appropriate of the various pit-removal algorithms tested. Were the user does not explicitly avoid endorheic basins in their artistic direction, our method improves the resulting terrain.

References

[1] Scott, JJ. Realism in data-based terrain synthesis. Ph.D. thesis; Victoria University of Wellington; 2020. http://hdl.handle.net/10063/9025.

[2] Gain, J, Merry, B, Marais, P. Parallel, realistic and controllable terrain synthesis. Computer Graphics Forum 2015;34(2):105–116. doi:10.1111/cgf.12545.

[3] Scott, JJ, Dodgson, NA. Evaluating realism in terrain synthesis. ???? Under revision for submission to *ACM Trans. Applied Perception*.

[4] Galin, E, Guérin, E, Peytavie, A, Cordonnier, G, Cani, MP, Benes, B, et al. A review of digital terrain modeling. Computer Graphics Forum 2019;38(2):553–577.

[5] Mandelbrot, BB. Stochastic models for the earth's relief, the shape and the fractal dimension of the coastlines, and the number-area rule for islands. Proceedings of the National Academy of Sciences 1975;72(10):3825–3828.

[6] Mandelbrot, BB. The Fractal Geometry of Nature. W. H. Freeman and Company; 1977.

[7] Mandelbrot, BB. Fractal landscapes without creases and with rivers. In: Peitgen, HO, Saupe, D, editors. The Science of Fractal Images; chap. Fractal Landscapes Without Creases and with Rivers. New York, NY, USA: Springer-Verlag New York, Inc. ISBN 0-387-96608-0; 1988, p. 243–260.

[8] Fournier, A, Fussell, D, Carpenter, L. Computer rendering of stochastic models. Commun ACM 1982;25(6):371–384. doi:10.1145/358523.358553.

[9] Miller, GSP. The definition and rendering of terrain maps. In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '86; New York, NY, USA: ACM. ISBN 0-89791-196-2; 1986, p. 39–48. doi:10.1145/15922.15890.

[10] Lewis, JP. Generalized stochastic subdivision. ACM Trans Graph 1987;6(3):167–190. doi:10.1145/35068.35069.

[11] Adams, D, Egbert, P, Brunner, S. Feature-based interactively sketched terrain. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. I3D '12; New York, NY, USA: ACM. ISBN 978-1-4503-1194-6; 2012, p. 208–208. doi:10.1145/2159616.2159654.

[12] Stachniak, S, Stuerzlinger, W. An algorithm for automated fractal terrain deformation. In: In Proceedings of Computer Graphics and Artificial Intelligence; vol. 1. 2005, p. 64–76.

[13] Rusnell, B, Mould, D, Eramian, M. Feature-rich distance-based terrain synthesis. The Visual Computer 2009;25(5-7):573–579. doi:10.1007/s00371-009-0332-6.

[14] Golubev, K, Zagarskikh, A, Karsakov, A. Dijkstra-based terrain generation using advanced weight functions. Procedia Computer Science 2016;101:152 – 160. 5th International Young Scientist Conference on Computational Science, YSC 2016, 26-28 October 2016, Krakow, Poland.

[15] Belhadj, F, Audibert, P. Modeling landscapes with ridges and rivers: Bottom up approach. In: Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia. GRAPHITE '05; New York, NY, USA: ACM. ISBN 1-59593-201-1; 2005, p. 447–450. doi:10.1145/1101389.1101479.

[16] Hnaidi, H, Guérin, E, Akkouche, S, Peytavie, A, Galin, E. Feature based terrain generation using diffusion equation. In: Computer Graphics Forum; vol. 29. Wiley; 2010, p. 2179–2186.

[17] Bernhardt, A, Maximo, A, Velho, L, Hnaidi, H, Cani, MP. Real-time terrain modeling using cpu-gpu coupled computation. In: Proceedings of the 2011 24th SIBGRAPI Conference on Graphics, Patterns and Images. SIBGRAPI '11; Washington, DC, USA: IEEE Computer Society. ISBN 978-0-7695-4548-6; 2011, p. 64–71. doi:10.1109/SIBGRAPI.2011.28.

[18] Becher, M, Krone, M, Reina, G, Ertl, T. Feature-based volumetric terrain generation. In: Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. I3D '17; New York, NY, USA: ACM. ISBN 978-1-4503-4886-7; 2017, p. 10:1–10:9. doi:10.1145/3023368.3023383.

[19] Musgrave, FK, Kolb, CE, Mace, RS. The synthesis and rendering of eroded fractal terrains. In: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '89; New York, NY, USA: ACM. ISBN 0-89791-312-4; 1989, p. 41–50. doi:10.1145/74333.74337.

[20] Chiba, N, Muraoka, K, Fujita, K. An erosion model based on velocity fields for the visual simulation of mountain scenery. The Journal of Visualization and Computer Animation 1998;9(4):185–194. doi:10.1002/(SICI)1099-1778(1998100)9:4<185::AID-VIS178>3.0.CO;2-2.

[21] Nagashima, K. Computer generation of eroded valley and mountain terrains. The Visual Computer 1998;13(9):456–464.

[22] Neidhold, B, Wacker, M, Deussen, O. Interactive physically based fluid and erosion simulation. In: Proceedings of the First Eurographics Conference on Natural Phenomena. NPH'05; Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. ISBN 3-905673-29-0; 2005, p. 25–33. doi:10.2312/NPH/NPH05/025-032.

[23] Beneš, B, Těšìnskỳ, V, Hornyš, J, Bhatia, SK. Hydraulic erosion. Computer Animation and Virtual Worlds 2006;17(2):99–108. doi:10.1002/cav.77.

[24] Beneš, B. Physically-based hydraulic erosion. In: Proceedings of the 22Nd Spring Conference on Computer Graphics. SCCG '06; New York, NY, USA: ACM. ISBN 978-1-4503-2829-6; 2006, p. 17–22. doi:10.1145/2602161.2602163.

[25] Št'ava, O, Beneš, B, Brisbin, M, Křivánek, J. Interactive terrain modeling using hydraulic erosion. In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA '08; Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. ISBN 978-3-905674-10-1; 2008, p. 201–210. URL: http://dl.acm.org/citation.cfm?id=1632592.1632622.

[26] Natali, M, Lidal, EM, Parulek, J, Viola, I, Patel, D. Modeling terrains and subsurface geology. In: EuroGraphics 2013 State of the Art Reports (STARs), 2013. Eurographics; Eurographics 2013 - State of the Art Reports; 2013, p. 155–173. doi:10.2312/conf/EG2013/stars/155-173.

[27] Cordonnier, G, Braun, J, Cani, MP, Beneš, B, Galin, E, Peytavie, A, et al. Large scale terrain generation from tectonic uplift and fluvial erosion. In: Computer Graphics Fotrum; vol. 35. Wiley; 2016, p. 165–175.

[28] Farr, TG, Rosen, PA, Caro, E, Crippen, R, Duren, R, Hensley, S, et al. The shuttle radar topography mission. Reviews of Geophysics 2007;45(2).

[29] ASTER, . Aster global digital elevation map announcement. ???? https://asterweb.jpl.nasa.gov/gdem.asp Accessed: 28-06-2019.

[30] MapMart, H. Harris-IntraSearch GeoEye-1 1m DSM and 5m DTM. ???? http://cms.mapmart.com/Products/DigitalElevationModel/HIDEMGeoEye1.aspx Accessed: 30-06-2016.

[31] Brosz, J, Samavati, FF, Sousa, MC. Terrain synthesis by-example. In: Advances in Computer Graphics and Computer Vision: VISAPP and GRAPP 2006. Springer. ISBN 978-3-540-75274-5; 2007, p. 58–77. doi:10.1007/978-3-540-75274-5_4.

[32] Zhou, H, Sun, J, Turk, G, Rehg, JM. Terrain synthesis from digital elevation models. IEEE Trans Visualization & Comp Graphics 2007;13(4):834–848. doi:10.1109/TVCG.2007.1027.

[33] Tasse, FP, Gain, J, Marais, P. Enhanced texture-based terrain synthesis on graphics hardware. Computer Graphics Forum 2012;31(6):1959–1972. doi:10.1111/j.1467-8659.2012.03076.x.

[34] dos Passos, VA, Igarashi, T. Landsketch: A first person point-of-view example-based terrain modeling approach. SBIM '13; ACM. ISBN 978-1-4503-2205-8; 2013, p. 61–68. doi:10.1145/2487381.2487382.

[35] Cruz, L, Velho, L, Galin, E, Peytavie, A, Guérin, E. Patch-based terrain synthesis. Proceedings of the 10th International Conference on Computer Graphics Theory and Applications; Berlin, France: GRAPP 2015; 2015,.

[36] Cruz, L, Velho, L. High-level techniques for landscape creation. Ph.D. thesis; The Instituto Nacional de Matemática Pura e Aplicada; Rio de Janeiro, Brazil; 2015.

[37] Saunders, RL. Realistic terrain synthesis using genetic algorithms. Master's thesis; Texas A&M University; 2006.

[38] Guérin, E, Digne, J, Galin, E, Peytavie, A, Wolf, C, Beneš, B, et al. Interactive example-based terrain authoring with conditional generative adversarial networks. ACM Trans Graph 2017;36(6):228:1–228:13. doi:10.1145/3130800.3130804.

[39] Wei, LY, Lefebvre, S, Kwatra, V, Turk, G. State of the art in example-based texture synthesis. In: Eurographics 2009, State of the Art Report, EG-STAR. Eurographics Association; 2009, p. 93–117.

[40] Efros, AA, Freeman, WT. Image quilting for texture synthesis and transfer. SIGGRAPH '01; ACM. ISBN 1-58113-374-X; 2001, p. 341–346. doi:10.1145/383259.383296.

[41] Kwatra, V, Schödl, A, Essa, I, Turk, G, Bobick, A. Graphcut textures: image and video synthesis using graph cuts. In: ACM Trans. Graphics; vol. 22. ACM; 2003, p. 277–286.

[42] Wu, Q, Yu, Y. Feature matching and deformation for texture synthesis. In: ACM SIGGRAPH 2004 Papers. ACM; 2004, p. 364–367. doi:10.1145/1186562.1015730.

[43] Pérez, P, Gangnet, M, Blake, A. Poisson image editing. In: ACM Trans. Graphics; vol. 22. ACM; 2003, p. 313–318.

[44] Shepard, D. A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM national conference. ACM; 1968, p. 517–524.

[45] Efros, AA, Leung, TK. Texture synthesis by non-parametric sampling. ICCV '99; IEEE Computer Society. ISBN 0-7695-0164-8; 1999, p. 1033–1038.

[46] Wei, LY, Levoy, M. Fast texture synthesis using tree-structured vector quantization. SIGGRAPH '00; ACM Press/Addison-Wesley. ISBN 1-58113-208-5; 2000, p. 479–488. doi:10.1145/344779.345009.

[47] Wei, LY, Levoy, M. Order-independent texture synthesis. 2003. https://graphics.stanford.edu/papers/texture-synthesis-sig03/ Accessed: 28-06-2019 (Earlier version is Stanford University Computer Science TR-2002-01).

[48] Lefebvre, S, Hoppe, H. Parallel controllable texture synthesis. In: ACM SIGGRAPH 2005 Papers. ACM; 2005, p. 777–786. doi:10.1145/1186822.1073261.

[49] Han, C, Risser, E, Ramamoorthi, R, Grinspun, E. Multiscale texture synthesis. In: ACM SIGGRAPH 2008 Papers. ACM. ISBN 978-1-4503-0112-1; 2008, p. 51:1–51:8. doi:10.1145/1399504.1360650.

[50] Dachsbacher, C, Meyer, M, Stamminger, M. Height-field synthesis by non-parametric sampling. Vision, Modeling and Visualization 2005 2005;:297–302.

[51] Lefebvre, S, Hoppe, H. Appearance-space texture synthesis. In: ACM SIGGRAPH 2006 Papers. ACM. ISBN 1-59593-364-6; 2006, p. 541–548. doi:10.1145/1179352.1141921.

[52] Kaspar, A, Neubert, B, Lischinski, D, Pauly, M, Kopf, J. Self tuning texture optimization. Computer Graphics Forum 2015;34:349–359.

[53] Kwatra, V, Essa, I, Bobick, A, Kwatra, N. Texture optimization for example-based synthesis. In: ACM SIGGRAPH 2005 Papers. ACM; 2005, p. 795–802. doi:10.1145/1186822.1073263.

[54] Barnes, C, Shechtman, E, Finkelstein, A, Goldman, DB. PatchMatch: A randomized correspondence algorithm for structural image editing. In: ACM SIGGRAPH 2009 Papers. ISBN 978-1-60558-726-4; 2009, p. 24:1–24:11. doi:10.1145/1576246.1531330.

[55] Darabi, S, Shechtman, E, Barnes, C, Goldman, DB, Sen, P. Image melding: Combining inconsistent images using patch-based synthesis. ACM Trans Graph 2012;31(4):82–1.

[56] Barnes, R, Callaghan, KL, Wickert, AD. Computing water flow through complex landscapes – part 3: Fill–spill–merge: flow routing in depression hierarchies. Earth Surface Dynamics 2021;9(1):105–121. doi:10.5194/esurf-9-105-2021.

[57] Yapiyev, V, Sagintayev, Z, Inglezakis, VJ, Samarkhanov, K, Verhoef, A. Essentials of endorheic basins and lakes: A review in the context of current and future water resource management and mitigation activities in central asia. Water 2017;9(10). doi:10.3390/w9100798.

[58] Barnes, R, Lehman, C, Mulla, D. Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. Computers & Geosciences 2014;62:117–127.

[59] Lai, J, Anders, AM. Modeled postglacial landscape evolution at the southern margin of the laurentide ice sheet: Hydrological connection of uplands controls the pace and style of fluvial network expansion. Journal of Geophysical Research: Earth Surface 2018;123(5):967–984. doi:10.1029/2017JF004509.

[60] Planchon, O, Darboux, F. A fast, simple and versatile algorithm to fill the depressions of digital elevation models. Catena 2002;46(2-3):159–176.

[61] Soille, P. Morphological carving. Pattern Recognition Letters 2004;25(5):543–550.

[62] Lindsay, JB. Efficient hybrid breaching-filling sink removal methods for flow path enforcement in digital elevation models. Hydrological Processes 2016;30(6):846–857.

[63] Lindsay, JB, Dhun, K. Modelling surface drainage patterns in altered landscapes using lidar. International Journal of Geographical Information Science 2015;29(3):397–411.

[64] Barnes, C, Shechtman, E, Goldman, DB, Finkelstein, A. The generalized PatchMatch correspondence algorithm. ECCV'10. ISBN 3-642-15557-X, 978-3-642-15557-4; 2010, p. 29–43.

[65] Zhou, Y, Shi, H, Lischinski, D, Gong, M, Kopf, J, Huang, H. Analysis and controlled synthesis of inhomogeneous textures. In: Computer Graphics Forum; vol. 36. Wiley; 2017, p. 199–212.

[66] Yu, P, Yang, X, Chen, L. Parallel-friendly patch match based on jump flooding. In: Advances on Digital Television and Wireless Multimedia Communications. Springer; 2012, p. 15–21. doi:10.1007/978-3-642-34595-1_3.

[67] Fei, D, Qingsong, Y, Teng, X. A gpu-patchmatch multi-view dense matching algorithm based on parallel propagation. Acta Geodaetica et Cartographica Sinica 2020;49(2):181. doi:10.11947/j.AGCS.2020.20180459.

[68] Cordonnier, G, Bovy, B, Braun, J. A versatile, linear complexity algorithm for flow routing in topographies with depressions. Earth Surface Dynamics 2019;7(2):549–562. doi:10.5194/esurf-7-549-2019.

[69] Bradley, RA, Terry, ME. Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika 1952;39(3/4):324–345.