CGI2017 manuscript No.
(will be inserted by the editor)

Coherent Multi-Layer Landscape Synthesis

Oscar Argudo[1] · **Carlos Andujar**[1] · **Antonio Chica**[1] · **Eric Guérin**[2] · **Julie Digne**[4] · **Adrien Peytavie**[4] · **Eric Galin**[3]

Abstract We present an efficient method for generating coherent multi-layer landscapes. We use a dictionary built from exemplars to synthesize high-resolution fully-featured terrains from input low-resolution elevation data. Our example-based method consists in analyzing real world terrain examples and learning the procedural rules directly from these inputs. We take into account not only the elevation of the terrain, but also additional layers such as the slope, orientation, drainage area, the density and distribution of vegetation, and the soil type. By increasing the variety of terrain exemplars, our method allows the user to synthesize and control different types of landscapes and biomes, such as temperate or rain forests, arid deserts and mountains.

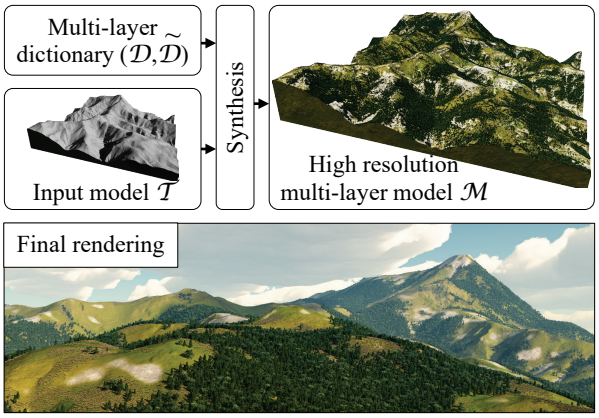

Fig. 1 Overview of our coherent multi-layer landscape synthesis: given a set of input exemplars, our method automatically creates a high-resolution consistent and coherent multi-layer terrain model from a low-resolution elevation model by matching input patches with the nearest dictionary atoms.

1 Introduction

Generating large-scale realistic landscapes with a high level of detail is a perennial challenge in Computer Graphics. With the increasing demand for virtual worlds, there is a growing need for automatic techniques that generate large scale terrains covered with vegetation at a very high resolution.

Procedural modeling, which aims at generating complex geometric models from simple generative rules, has undergone tremendous developments over the past decade. However, several limitations make it difficult to create these rules explicitly. In particular, generating geomorphologically-consistent terrains featuring a vast variety of landforms remains a difficult task. The problem becomes even more challenging when considering the generation of layered terrains or landscapes, *i.e.*, models defined by different types of layered information such as terrain elevation, sand and rock thickness, vegetation type and density, or humidity. Our work comes from the observation that those parameters are strongly correlated and can be modeled as layers inter-influencing each other. This inter-dependency makes the design of coherent, biologically- and physically- plausible generative rules even more difficult.

Traditionally, the standard workflow consists in first synthesizing the terrain, eroding it with procedural or simulation-based algorithms to generate sediment layers, and finally using ecosystem simulations to generate the vegetation. The originality of our method is to generate the different data layers using a joint synthe-

[1] ViRVIG, Computer Science Department, Universitat Politecnica de Catalunya, Spain
[2] Univ Lyon, INSA-Lyon, CNRS, LIRIS, F-69621, France
[3] Univ Lyon, Université Lyon 2, CNRS, LIRIS, F-69676, France
[4] Univ Lyon, Université Lyon 1, CNRS, LIRIS, F-69622, France

sis approach, taking simultaneously into account the correlated elevation of the terrain, the distribution of vegetation density, soil type, slope and orientation. Our method generates large-scale multi-layer landscapes with a high degree of realism and coherence between the different layers. By adopting an example-based procedural synthesis approach, we avoid any explicit modeling. Furthermore, our method allows to add variety to the synthesized models and allows to control and author different types of biomes such as alpine mountains with forests or arid plateaus simply by increasing the number and variety of examples. Therefore, our approach provides a powerful and efficient framework to perform the inverse procedural modeling of multi-layer terrains at a low computational cost.

Our algorithm proceeds in two steps (Figure 1). Given a set of exemplars, a pre-processing step creates a multi-layer dictionary. During the landscape synthesis step, the user edits a low-resolution input elevation and possibly a small subset of other layers. The matching algorithm decomposes the input into patches which are matched with atoms in the dictionary. Selected multi-layer atoms are blended together to generate the final high-resolution multi-layer model.

The main contributions are as follows. **1**. We propose *a set of matching functions* adapted to the multi-layer representation for finding the best patch in the dictionary. Our method combines multi-layer information in a coherent way and guarantees that no ambiguity is created when synthesizing the landscape. **2**. We present an *example-based dictionary extraction* combined with a coherent multi-layer terrain synthesis algorithm. Given a low-resolution input, we automatically generate an augmented high-resolution model. **3**. Our approach allows the user to *control the features* of the output terrain by changing the style of different regions and classes of atoms in the dictionary.

2 Related work

Our work relates to terrain modeling and ecosystem generation, which can be classified into procedural, example-based and simulation-based approaches. This section presents a focused overview; we refer the reader to more general surveys on procedural terrain modeling [21] and plant and ecosystem simulation [5].

Procedural modeling methods exploit the observation that landform features repeat at different scales and define the elevation either as fractals [15, 20] or by using a combination of scaled noise-based functions [17]. Several improvements were proposed to improve user control, such as terrains generated from feature curves [14], rivers [10] or a hierarchical construction tree representation [11]. Specifying generative rules that preserve the overall coherence of the scene is a difficult task, mainly because of the indirect control over the generation processes.

Inverse procedural modeling is a general approach which aims at inferring the input control parameters of procedural models from examples or constraints. Some techniques have been successfully developed for generating vegetation [22]. Recently, the sparse representation of terrains [12] combined atoms whose characteristic landforms features can be extracted from exemplars and stored in an optimized dictionary.

Our method also relies on a dictionary learned from examples. The originality of our approach is that it processes not only the terrain elevation, but also many different channels encoding other parameters such as vegetation density, slope, humidity to generate high resolution terrains in a coherent way.

Simulations aim at generating realistic landscapes with eroded mountains, sedimentary valleys and realistic plant distributions. *Erosion simulations* [17] are often used as a post processing step to add realism to procedurally generated terrains. Hydraulic erosion techniques were further extended and refined in [2,19]. Large-scale simulation of erosion at the level of entire mountain ranges was addressed in [3]. Erosion-based techniques are difficult to control and cannot be used to simulate large scale terrains at a high resolution.

Ecosystem simulations aim at producing realistic plant distributions [18] according to the characteristics of the environment, and in general rely on particle-based simulations where plants compete for resources such as light and space [4]. Several improvements were proposed such as simulating multilevel plant communities [16] and asymmetric plant competition [1].

Example-based synthesis approaches borrow from texture synthesis methods [13, 25] and aim at generating realistic terrains by combining patches extracted from exemplars. A first method proposed in [26] extracts high-resolution height field patches from a terrain exemplar and combines them according to a user-painted coarse map. This approach was extended and improved to allow better control [8,9]. Recently, an interactive approach for creating virtual worlds using statistical example-based synthesis to automate content synthesis and deformation was proposed in [7].

In contrast, our method analyzes exemplars to generate coherent multi-layer dictionary implicitly storing the relationships between terrain elevation, vegetation densities and other parameters such as solar irradiance or upstream drainage area. A key contribution of our work lies in the use of heterogeneous signals, *i.e.* the signal encodes information of different kinds such as

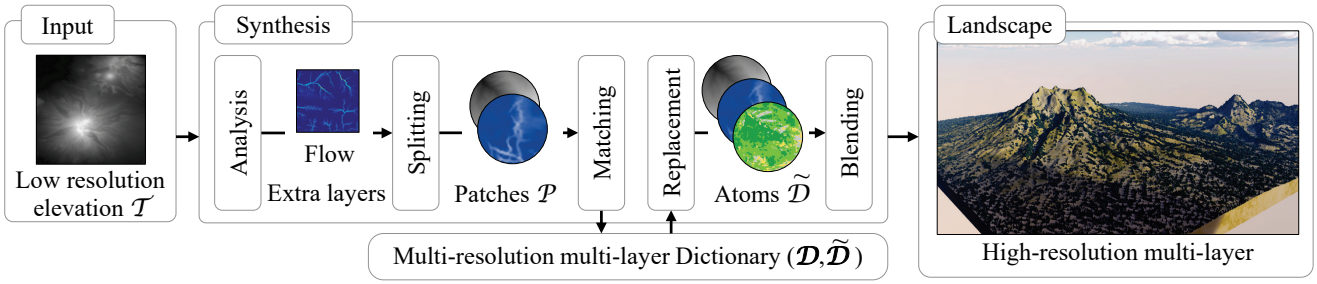

Fig. 2 Given a low-resolution single-layer terrain model, we synthesize a high-resolution multi-layer landscape model with sand, rock layers, and vegetation. Our algorithm analyzes the input $\mathcal{T}$ and decomposes it into patches $\mathcal{P}$. It then matches $\mathcal{P}$ with the atoms $\mathcal{D}_i$ of the dictionary, and guarantees that the synthesized layers are coherent with the input $\mathcal{T}$.

elevation, vegetation information or textures. This research field is known in the signal processing literature as joint sparse approximation [24]. Our framework proposes a simple and efficient implementation with a view to allowing for a faster processing without any sophisticated sparse approximation algorithm.

3 Overview and notations

The overall workflow of our method is divided into two steps: a dictionary extraction from a set of exemplars performed as a pre-processing step (Figure 3), and a high-resolution multi-layer terrain synthesis from a low resolution model (Figure 2).

3.1 Dictionary creation

At the heart of our method is a dictionary built from a set of multi-layer exemplars. The dictionary is created by analyzing multi-layer input exemplars and contains multi-layer atoms. Multi-layer terrain exemplars are first decomposed into partially overlapping patches as described in [12]. The multi-resolution dictionary is a set of two dictionaries denoted as $(\mathcal{D}, \widetilde{\mathcal{D}})$, low- and high- resolution, with the same number of atoms and a one-to-one correspondence between their atoms. Thus, given a decomposition over $\mathcal{D}$, the reconstruction from $\widetilde{\mathcal{D}}$ can be obtained simply by keeping the decomposition and replacing the atoms through the one-to-one correspondence.

3.2 Multi-layer terrain synthesis

The inputs of our algorithm are a low-resolution terrain $\mathcal{T}$ containing either a single elevation layer or additional layers, and a multi-resolution dictionary $(\mathcal{D}, \widetilde{\mathcal{D}})$. The terrain $\mathcal{T}$ can be either a user-drawn sketch, specifying a coarse elevation map and a distribution of different materials and vegetation types over the terrain, or a real digital elevation map that the user wants to augment with additional layers.

Our algorithm decomposes $\mathcal{T}$ into patches, denoted as $\mathcal{P}$, which are matched to the nearest low-resolution atoms $\mathcal{D}$ of the dictionary. The patches $\mathcal{P}$ are then replaced by the *high-resolution and multi-layer* atoms $\widetilde{\mathcal{D}}$ corresponding to $\mathcal{D}$ and the terrain is built by blending the high-resolution atoms. The output of our algorithm is a high resolution multi-layer terrain $\widetilde{\mathcal{T}}$ whose layers are obtained from the different layers extracted from the exemplars.

Our method lends itself for synthesizing landscapes with different kinds of information (detailed elevation, vegetation density and type, sediment thickness) depending on the number and categories of layers included in the dictionary. It can be used to create vegetation and population density, or replace specific regions in a consistent way as demonstrated in Section 6.

Layers store vector or scalar data. Superscripts will refer to layers for both input patches and dictionary atoms. $\mathcal{P}^e$ and $\mathcal{P}^h$ denote the elevation and the normalized elevation respectively. $\mathcal{P}^a$ will refer to the mean elevation of the layer $\mathcal{P}^h$, and $\mathcal{P}^s$ to the mean deviation of the layer $\mathcal{P}^e$, and will be computed as follows:

$$\mathcal{P}^a = \overline{\mathcal{P}^e} \qquad \mathcal{P}^s = \|\mathcal{P}^e - \mathcal{P}^a\| \qquad \mathcal{P}^h = (\mathcal{P}^e - \mathcal{P}^a)/\mathcal{P}^s$$

The other layers for vegetation density, upstream drainage area, solar irradiance and classes will be denoted as $\mathcal{P}^v$, $\mathcal{P}^u$, and $\mathcal{P}^l$ respectively.

4 Dictionary construction

The creation of the dictionary is performed as a pre-processing step, independent of the synthesis step itself. Exemplars are down-sampled to get low-resolution multi-layer exemplars. Dictionary atoms are extracted from both original and down-sampled exemplars, processed layer by layer and re-assembled into multi-layer atoms (Figure 3). The computations are performed both

on the low-resolution and high-resolution atoms so that the one-to-one correspondence is preserved. Atoms are defined as square regular grids storing digital elevation data, and combined with a radial falloff function for smooth blending. The layers represent any kind of data: elevation, vegetation density for every different species, and computed data such as upstream area or slope.

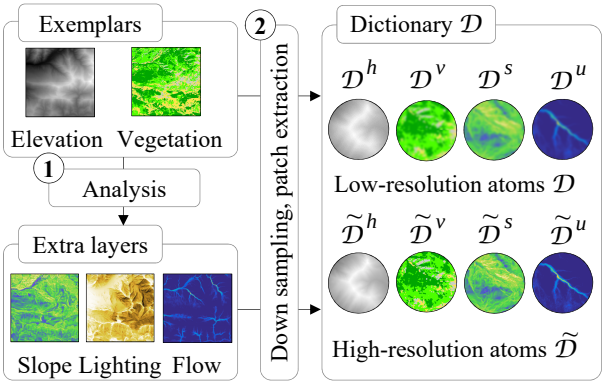

Fig. 3 Dictionary extraction from a set of input exemplars.

Elevation layers are first centered according to the mean elevation value of the patch, and then normalized. Some layers such as the mean elevation, the slope, the global vegetation density, the solar irradiance and upstream drainage area layers are directly computed from the elevation layer.

The dictionary structure is defined as a set of multi-layer low-resolution atoms associated to their high-resolution counterpart. They will be denoted as $\mathcal{D}_i^j$ and $\widetilde{\mathcal{D}}_i^j$ respectively, where $j \in \{0, \dots, l-1\}$ denotes the layer, and $i \in \{0, \dots, n-1\}$ refers to the i-th atom in the dictionary. The low-resolution dictionary $\mathcal{D} = \{\mathcal{D}_i\}$ will be used to match the input to the dictionary whereas the high-resolution dictionary $\widetilde{\mathcal{D}} = \{\widetilde{\mathcal{D}}_i\}$ will be used for synthesizing the multi-layer terrain.

5 Multi-layer terrain synthesis

Our synthesis algorithm is designed so that the matching process should match an input terrain patch to a unique dictionary atom efficiently. Moreover, it successfully handles multi-layer data and can represent nonlinear features such as the norm of an atom, local curvature or any feature of interest.

In order to be matched with a given atom $\mathcal{D}_i$ of the dictionary, an input terrain patch $\mathcal{P}$ comes with a subset Γ of the set of layer indices $\Omega = \{0, \dots, l-1\}$. Let $\mathcal{P}^j$ be the j-th layer of the input terrain patch. Let g_j denote the matching function of the j^{th} layer. We define the matching function $g : \mathbb{R}^n \times \mathbb{R}^n \to [0, 1]$ as:

$$g(\mathcal{P}, \mathcal{D}_i) = \sum_{j \in \Gamma} \omega_j \, g_j(\mathcal{P}^j, \mathcal{D}_i^j) \qquad \sum_{j \in \Gamma} \omega_j = 1$$

The coefficients ω_j form a partition of unit and weight the relative impact of the different layers in the matching. The matching functions g_j detailed in the next subsections evaluate the similarity between patches and atoms for the different types of layers. The higher the value $g(\mathcal{P}, \mathcal{D}_i)$, the better the correspondence between the input patch and the i-th atom. Let k denote the atom index that minimizes the matching function:

$$k = \underset{i}{\operatorname{argmax}}\, g(\mathcal{P}, \mathcal{D}_i)$$

The reconstructed patch $\widetilde{\mathcal{P}} = \{\widetilde{\mathcal{P}}^j\}$ contains the layers of the high-resolution matched atom k. The reconstructed elevation will be computed as:

$$\widetilde{\mathcal{P}^e} = (\mathcal{P}^h \cdot \mathcal{D}_k^h)\, \mathcal{P}^s \, \widetilde{\mathcal{D}}_k^h + \mathcal{P}^a$$

All the other layers will be directly reconstructed from the dictionary atoms, therefore

$$\forall j \in \Gamma - \{e\},\ \mathcal{P}^j = \widetilde{\mathcal{D}}_k^j$$

In the particular setting of $l = 1$, $\Gamma = \{h\}$ and $g_h(\mathcal{P}^h, \mathcal{D}_i^h) = |\mathcal{P}^h \cdot \mathcal{D}_i^h|$ where the layer h contains vector data and atoms in the dictionary are normalized, we obtain the regular Matching Pursuit algorithm [23] with a sparsity of $s = 1$. Our framework generalizes this approach by introducing additional layers in the matching step and allowing complex matching layouts.

The landscape reconstruction from the patches $\widetilde{\mathcal{P}}$ is performed by blending the overlapping patches with a falloff function of the distance to the patch center. In the remainder of this section, we rely on this general framework and show how the different types of layers are processed.

5.1 Orientation

In this section, we consider the layer h that represents the elevation of a terrain. Recall that dictionary atoms $\mathcal{D}_i^h$ and terrain patches $\mathcal{P}^h$ are centered at zero in order to avoid a constant component term in the projection which would make the matching less meaningful. Consequently, if we use the matching function $g_h(\mathcal{P}^h, \mathcal{D}_i^h) = |\mathcal{P}^h \cdot \mathcal{D}_i^h|$, a good matching can be achieved with $\mathcal{P}^h \cdot \mathcal{D}_i^h < 0$, *i.e.* by inverting the dictionary atom, which produces inaccurate results for our landscape generation: North faces may become South faces, ridges may become valleys, with dramatic consequences on additional layers, such as riverside vegetation on top of mountains as illustrated in Figure 4.

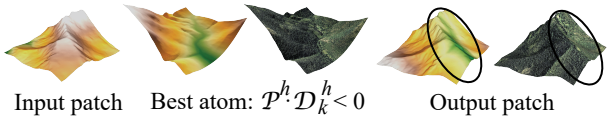

Fig. 4 By using standard sparse synthesis, the atom that best matches the input ridge elevation data has a negative coefficient and corresponds to a valley. The vegetation density of the valley atoms are incorrectly placed on top of the terrain patch. In contrast, our approach avoids inversions.

In our framework, we are synthesizing layers that store other important properties, therefore we need to preserve the overall coherence between layers. To solve this problem, we use the following matching function that maps onto $[0, 1]$:

$$g_h(\mathcal{P}^h, \mathcal{D}_i^h) = (1 + \mathcal{P}^h \cdot \mathcal{D}_i^h)/2$$

Figure 5 demonstrates the importance of preserving the terrain orientation. The dictionary was created from multi-layer data containing the elevation (used for patch matching) and the normal (used in synthesis). The generated terrain is a high resolution elevation map augmented with a coherent normal map.

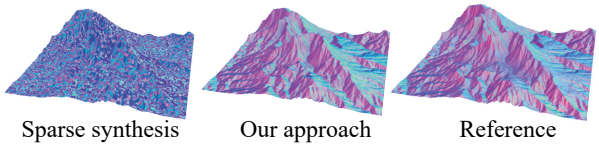

Fig. 5 Sparse synthesis generates patches with arbitrarily-oriented atoms, resulting in inconsistent orientations which affects the generation of other layers: North rims would be replaced by South rims, thus misplacing orientation-dependent content such as solar irradiance or vegetation. Our method preserves the orientation and generates coherent patches.

5.2 Elevation and slope

The matching algorithm can also benefit from information about the altitude of the atoms and patches, as well as their mean elevation deviation that approximates slope. The altitude matching function is:

$$g_a(\mathcal{P}^a, \mathcal{D}_i^a) = k(|\mathcal{P}^a - \mathcal{D}_i^a\|) \qquad k(x) = e^{-x^2/\sigma^2}$$

The standard deviation coefficient σ serves as a user-control parameter. We chose σ so that the Gaussian should be equal to 0.5 at the medium difference:

$$\sigma = (2\sqrt{\ln 2})^{-1} \max_i(\|\mathcal{P}^a - \mathcal{D}_i^a\|)$$

We use the same matching function for the slope function g_s. The weight ω_a allows users to control altitude-dependent content such as snow on high peaks, whereas ω_s controls slope-dependent content such as sediments or trees.

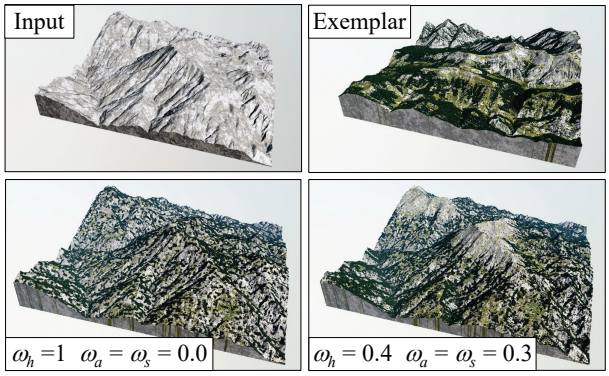

Fig. 6 Activating the altitude and slope matching functions gives a vegetation distribution that better fits the exemplar.

Figure 6 illustrates the impact of the coefficients ω_a and ω_s on the vegetation synthesis when using a dictionary containing a mean altitude layer and a slope layer that are derived directly from the elevation data (mean elevation and slope magnitude). The synthesis was performed on a input elevation map $\mathcal{T}$ with steeper slopes than the exemplar, which explains the scattered vegetation compared to large forests in the exemplar.

5.3 Context layers

environmental properties (such as specific climate or illumination conditions) that may extend over multiple patches. Taking into account the context is crucial for improving the overall matching process; therefore, we use context layers computed from the neighboring patches to improve the matching process.

Context data for a given patch $\mathcal{P}^j$ is computed over a spatial domain Ω_j embedding $\mathcal{P}^j$. In our experiments, the radius of the domain ranged for twice to eight times the radius of the patch. In our implementation, we perform a hierarchical down-sampling of the information contained in Ω_j so as to define a multi-scale description of the patch *neighborhood* and speed up the evaluation of the matching function. Although context layers include data from a larger domain $\Omega_j \supset \mathcal{P}^j$, atoms $\mathcal{D}_j$ and $\widetilde{\mathcal{D}_j}$ have identical spatial extents. Atoms in $\widetilde{\mathcal{D}}$ are used exclusively in the last step of the terrain synthesis when replacing patches with their high resolution counterparts.

Figure 7 compares the results obtained by increasing the size of the neighborhood when using context lay-

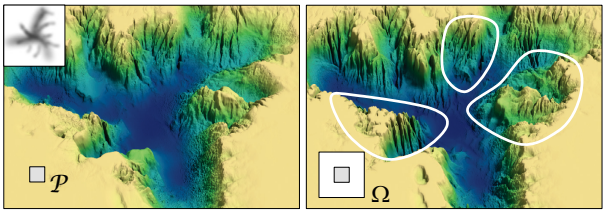

Fig. 7 Elevation synthesis using context layers. The input sketch is shown as an inset. From left to right: output terrain with simple matching, and with a context layer. The grey square denotes the patch size, and the white one indicates the approximate extent of the neighborhood Ω considered for computing the context layers.

ers for capturing the landform features. Context layers allow for a more realistic reconstruction of the cliffs, valleys and flat terrain landforms such as plateaus as the best matching atom can be determined according to the features in the neighborhood of the patch.

5.4 Class layers

We introduce class layers as a powerful tool to control the style of the synthesized multi-layer terrain model, *e.g.* rocky mountains, snow peaks, hills with forests. Classes are defined by using a specific abstract layer that defines the desired class of patch.

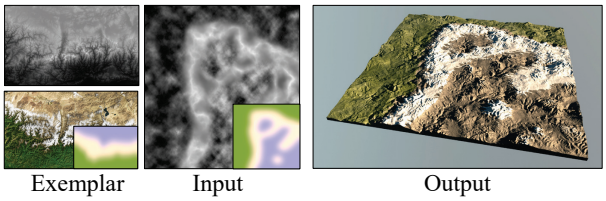

Fig. 8 Terrain authoring from a sketch defining 3 classes: hills with forests, desert mountains, snow peaks. The dictionary includes segmented exemplars from the central part of the Himalayas.

Dictionary atoms are first labeled to identify multiple differentiated regions from distinct classes (Figure 8). A classification layer stores vectors whose components define the relative probability (terms range in $[0,1]$ and sum to 1) that the patch should belong to the corresponding class. An exemplar with 3 different classes leads to 3-component vectors for the class layer (Figure 8). The matching method consists in choosing atoms that have preferably the same class as the input terrain patches. We define the matching function g_c as:

$$g_c(\mathcal{P}^c, \mathcal{D}_i^c) = \frac{1}{n} \sum_{k=1}^{n} \mathcal{P}^c(k) \cdot \mathcal{D}_i^c(k)$$

where n represents the number of classes; $\mathcal{P}^c(k)$ and $\mathcal{D}_i^c(k)$ represents the k^{th} class information sample for the input patch and dictionary atom respectively. Samples are normalized class vectors: normalization is important so that patches with smoothly varying classes should not match well with atoms of a single class.

5.5 Matching multiple exemplars

Our method allows to use dictionaries created from different exemplars and featuring various types of landscapes. Without loss of generality we present multiple exemplars matching with 2 biomes. In this particular setting, we consider two dictionaries $\mathcal{A}$ and $\mathcal{B}$ that represent the two biomes. The user has to paint two additional input layers α and β that describe the relative influence of the respective biomes. Note that the sum of α and β should be 1 everywhere. For the matching and reconstruction, we use the following algorithm: for all patches $\mathcal{P}$ in the input terrain:

1. Find the atoms $\mathcal{A}_i \in \mathcal{A}$ that maximizes $g(\mathcal{P}_\alpha \odot \mathcal{P}, \mathcal{P}_\alpha \odot \mathcal{A})$. Perform the same task with $\mathcal{P}$ for the other dictionary $\mathcal{B}$ weighted by β to find $\mathcal{B}_j$.
2. Blend the two high resolution atoms and generate $\mathcal{P} = \widetilde{\mathcal{P}}_\alpha \odot \widetilde{\mathcal{A}}_i + \widetilde{\mathcal{P}}_\beta \odot \widetilde{\mathcal{B}}_i$ where $\widetilde{\mathcal{P}}_\alpha$ and $\widetilde{\mathcal{P}}_\beta$ are the upsampled versions of $\mathcal{P}_\alpha$ and $\mathcal{P}_\beta$.

where $\odot$ denotes Hadamard element-wise vector multiplication.

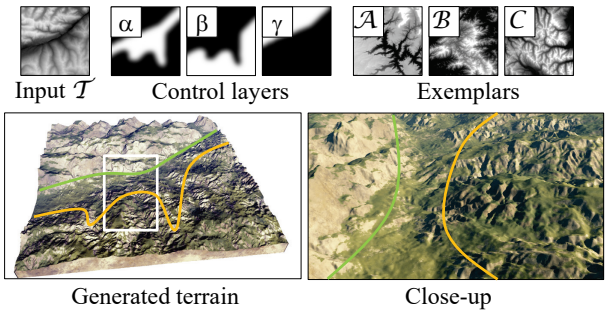

Fig. 9 High-resolution landscape generated from a low resolution elevation map, three exemplars and control layers indicating the preferred exemplar. The dictionaries were created from the Rocky Mountains, the Grand Canyon and Smokies National Park elevation maps.

5.6 Environmental layers

Layers representing *global* environmental information can be used to further improve the overall landscape, *i.e.* terrain and vegetation, synthesis process. Contrary to elevation or vegetation density layers that only provide *local* information at a given point, global layers

store parameters that are derived from a *global* analysis of the terrain. Such layers are particularly useful for implicitly representing correlations between neighboring patches and introducing coherence terms in the equation that evaluates the matching between patches and atoms. In our system, we experimented with two types

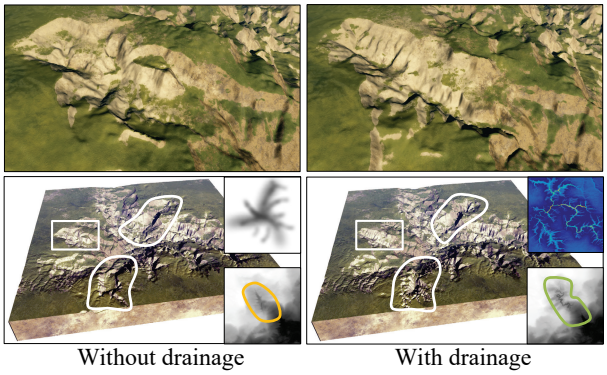

Fig. 10 Influence of the upstream drainage area layer: gullies and ravines are better reconstructed using it.

of global information layers: the upstream drainage area that approximates the average flow of water passing through a patch, and the average solar irradiance that represents the average amount of sunlight received by a patch. If not provided, those layers can be computed from the elevation data of the exemplars and the input $\mathcal{T}$ to generate the corresponding layers for patches $\mathcal{P}^u$ and $\mathcal{P}^l$ and dictionary atoms $\mathcal{D}^u$ and $\mathcal{D}^l$. The matching function is simply defined as:

$$g_u(\mathcal{P}^j, \mathcal{D}_i^j) = (1 + \mathcal{P}^j \cdot \mathcal{D}_i^j)/2 \qquad j \in \{u, l\}$$

The upstream drainage area is computed by simulating the flow of a large number of particles randomly distributed over the surface of the terrain and measuring the number of particles passing through every patch. Figure 10 shows that taking the drainage area into account allows for a better matching reconstruction of ravines and gullies. Solar irradiance (layer $\mathcal{P}^l$) is calculated based on latitude and longitude by intersecting rays from the sun position along its trajectory with the terrain. This captures terrain self-shadowing and provides average direct illumination from the sun.

6 Results

Our system automatically synthesizes *coherent* high-resolution multi-layer landscapes, *i.e.*, terrains with detailed elevation, soil type, sediment layers covered with a realistic distribution of different types of vegetation from a few input low-resolution layers (in general elevation and control layers). Instead of relying on complex procedural ecosystem simulations, our method reproduces the patterns and characteristics of the dictionary exemplars and preserves the overall coherence of the different layers (Figure 11, 14).

An important feature of our framework is its *versatility*: it can be used to generate an arbitrary number of layers of different types. Moreover, our dictionary-based system allows the user to enhance the database with as many atoms as needed, completing it with atoms featuring sediments or different kinds of plants.

The user *controls* the multi-layer terrain generation process by adjusting the weighting coefficients for the input and synthesized layers (Figure 6). Artists can freely provide, besides the elevation data, arbitrary layers as inputs, choose the layers and define their purpose: soil type, humidity, vegetation density or biome as long as the dictionary atoms encode these layers. The algorithm produces high-resolution terrains or additional layers consistent with the user-provided input.

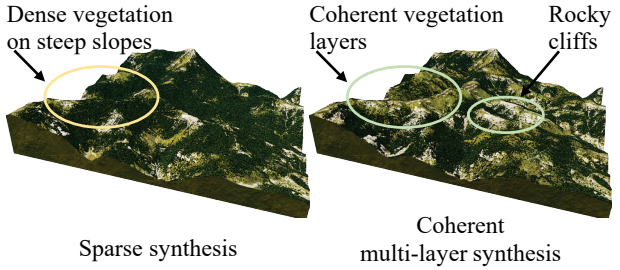

Fig. 11 Comparison between sparse synthesis and our method. Left image shows the vegetation distribution without taking into account the orientation, mean altitude and slope. Right image shows the coherent distribution: as exemplars do not have trees at high altitudes, our method produces trees that conform to this rule.

Figures 8 and 9 show examples of sketch-based authoring. The sketch contains a coarse description of the desired classes (forest, desert, peaks), with the purpose of synthesizing consistent biomes. The influence of the sketches in the matching process can be controlled by modifying the weighting coefficients. Figure 14 shows an example where control is achieved by sketching the expected distribution of different vegetation layers (tree, shrub, grass). Figure 15 shows an example of an authoring session. This example demonstrates that the synthesized layers (here the vegetation layer) can be in turn used and modified to guide the synthesis, in a coherent feedback loop.

Instead of defining a vegetation distribution for each vegetation type, the user may simply provide a vegeta-

Layers		Multi-layer terrain synthesis							Time (s)		
Input	**Output**	**Figure**	**Patch**	**Atoms**	**Patches**	**Input**	**Output**	**Scale**	Dict.	Match.	Synth.
h, c	h, t	8	24^2	3920	289	185^2	740^2	$\times 4$	0.20	0.07	0.43
h, c	h, t	9	24^2	5401	1521	450^2	1350^2	$\times 3$	0.35	3.19	9.95
h	h, v	11	32^2	2116	324	280^2	1400^2	$\times 5$	0.08	0.03	0.47
h, v, c	h, v	14	10^2	3969	10816	513^2	5130^2	$\times 10$	0.07	1.91	7.45
h, v	h, v	12	24^2	3481	29584	2048^2	2048^2	$\times 1$	0.11	3.07	7.47

Table 1 Statistics: patch size, number of atoms in the dictionary, number of patches, size of the input and output terrains, amplification factor. We also report timings (in s) for the dictionary construction, matching process, and patch replacement.

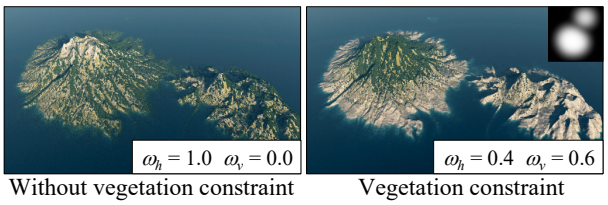

Fig. 12 Vegetation control: left image shows a terrain without vegetation density control, *i.e.*, following the distribution of the dictionary exemplar, whereas right image shows a smooth disc-shaped constraint. The algorithm automatically selects atoms with no vegetation under water.

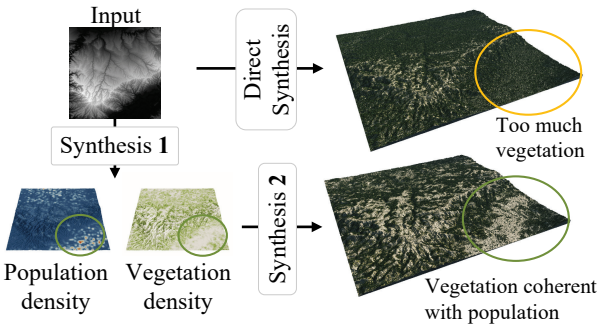

Fig. 13 Example of a two-step vegetation synthesis. First, we produce a vegetation layer coherent with a population density. Then, this layer is used to produce the distribution of three vegetation classes. In the first step the resolution was increased by 3 whereas it was preserved in the second step. Using directly the second dictionary on the input terrain cannot account for population density and thus places forests on areas with a high population density.

tion density layer (single scalar), and use the dictionary to convert the overall density into vegetation distributions for the different types of vegetation. Figure 12 illustrates this case: we created another layer for the dictionary that encodes the (weighted) sum of other vegetation layers, and used it to compare the average density between atoms and patches.

6.1 Multi-step synthesis

Our method allows to execute the synthesis process iteratively, using some of the output layers of one iteration as input layers for the next iteration. Therefore, we can use several dictionaries with different layer subsets in a coherent way.

Figure 13 shows an example of a two-step workflow. The first iteration generated a dictionary extracted from a real dataset of Catalonia, which was in turn used to synthesize coherent population and vegetation densities. The second iteration generated a higher resolution dictionary containing per-class distributions which was used to synthesize the different types of vegetation according to the previously generated vegetation density.

6.2 Performance

Our method was implemented and tested on an Intel Core i7 with 16 GB of RAM. Table 1 presents an overview of the different cases and reports the corresponding statistics. Timings demonstrate that our approach is efficient and can be used in practical terrain authoring applications. Although our current framework has been coded into a single-threaded CPU implementation, our method lends itself for a parallel implementation on the GPU.

The dictionaries can be extended easily by adding new layers to existing atoms, or by adding new atoms from other exemplars. Increasing the size of the dictionary allows for more variety and yields better results, at the cost of a more computationally demanding matching step. *Matching* performs in a few seconds and even less than a second for average-size dictionaries. The matching step of Figure 9 required one pass per class; timings take into account the multiple passes. *Synthesis and blending* performs in less than a few seconds on most examples. Notable exceptions are reported for the largest synthesized model (Figure 14), with a large terrain (5130×5130) involving more than 10 k patches, and Figure 12 which contains almost 30 k patches.

The time needed to evaluate $g_j(\mathcal{P}^j, \mathcal{D}_i^j)$ for complex data becomes the more expensive as the number of lay-

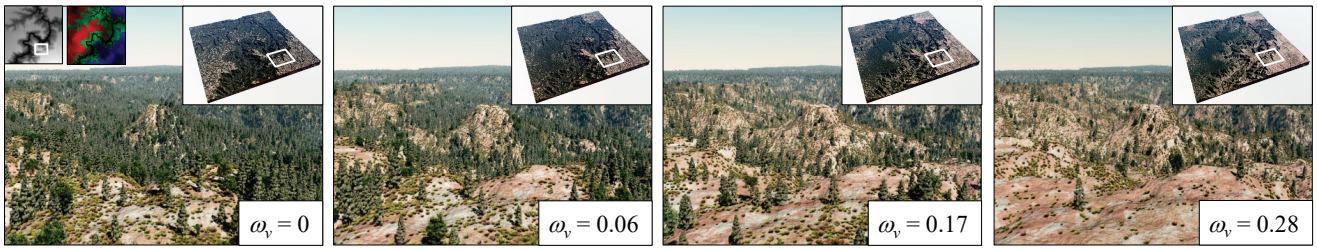

Fig. 14 Vegetation control over a large terrain: the vegetation distribution dictionary was created from the eastern Pyrenees exemplars (Figure 1), and the different species were prescribed by the user by defining three different classes (trees in red, bushes in green and sand in blue). Results are shown for different values of the control parameter ω_v.

ers increases. In our implementation, we optimized the computation by using a Poisson disc-based distribution of samples inside the patch area and evaluating g_j only over the reduced set of center points.

6.3 Discussion

Our dictionary-based framework has several applications. The input can be real digital elevation models, rough sketches drawn by hand, or a combination of both, containing a single layer (*e.g.* elevation), or multiple layers. The output terrain contains always as many layers as available in the dictionary, thus providing coherent data amplification.

A key feature of our approach is to provide a unified, easy-to-control, and flexible model for multi-layer landscape synthesis generating plausible and predictable results. Although alternative methods exist for some specific problems, none of them cover simultaneously all the applications supported by our framework. Our method provides control to the user and allows him to create any arbitrary layer in a coherent way. These two features are key for dictionary reusability and, ultimately, for effective terrain creation, editing and synthesis. *Context* layers combined with *global environmental layers* allow us to generate spatially coherent patches that more faithfully reproduce landform features such as gullies, erosion lines or plant clusters. Finally, our approach can be extended easily by considering other types of layers and defining the corresponding appropriate matching function.

As for all example-based approaches, our method may require a large input dataset to synthesize terrains. The dictionary extraction pre-processing step is very efficient. Although it may be difficult to find real world exemplars with appropriate layers, the set of exemplars can be completed with results obtained by computer simulations. Our framework offers many possibilities for reusing dictionaries, since it is built independently of the synthesis step.

Although the coherence between the different layers of an output patch is guaranteed by construction, the coherence between neighboring patches in the output is affected by the variety of atoms in the dictionary. This limitation has two consequences. First, mixing exemplars from radically different biomes (*e.g.* rain forest and desert) into the same dictionary may result in poor spatial coherence or sharp transitions. That limitation may be alleviated by providing a sketch of the desired distribution, as described in Section 5.5.

Another limitation of our method is that it does not properly handle structured layouts such as road-networks, villages or cities: the synthesis process does not guarantee that the structures would seamlessly link between two neighboring patches. Our method can nevertheless synthesize statistic information such as population density (Figure 13), which in turn may be used as input to generate villages or cities [6].

7 Conclusion

We have presented a multi-layer example-based approach to synthesize realistic landscapes, *i.e.* terrains containing heterogeneous information layers such as elevation, vegetation density or soil type. The cost function for matching dictionary atoms with terrain patches allows joint synthesis of coherent information layers. While our method lends itself for representing statistical data layers such as vegetation or population density, it cannot be directly used to synthesize constructs such as road networks or cities. Bridging the gap between our approach and road and city generation techniques is a challenging problem worth investigating as future work.

Acknowledgments

This work has been partially funded by the Spanish Ministry of Economy and Competitiveness and FEDER Grant TIN2014-52211-C2-1-R, the Spanish Ministry of

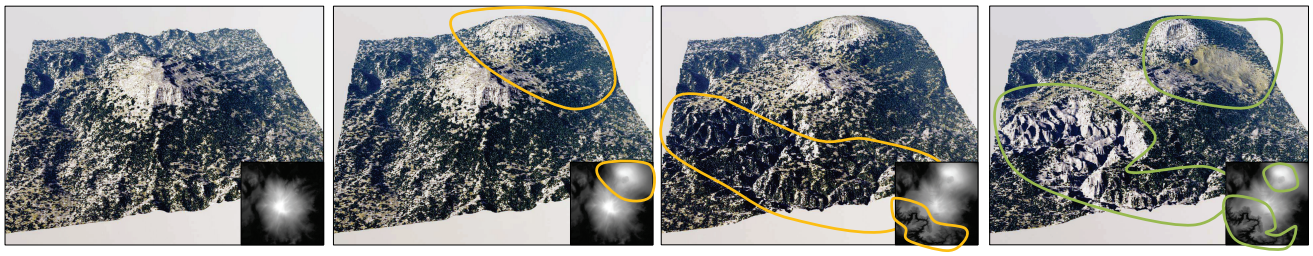

Fig. 15 Example of an incremental authoring session. Given an input digital elevation (grey scale inset), our method automatically generated a detailed layered terrain model with bedrock, soil and vegetation layers from the multi-layer dictionary (left image). The input was incrementally edited by adding a volcano and a canyon (center left and center right). Finally, the designer re-used the synthesized vegetation layer, down-sampled it, changed the density in the canyon and relaunched a complete synthesis process with a two-layer input to get the final landscape (right image). This authoring session took less than 10 minutes, including user editing and terrain synthesis.

Education, Culture and Sports PhD Grant FPU13/01079. This work is part of the project PAPAYA, funded by the Fonds National pour la Société Numérique, and project HDW ANR-16-CE33-0001.

References

1. Alsweis, M., Deussen, O.: Modeling and Visualization of symmetric and asymmetric plant competition. In: Eurographics Workshop on Natural Phenomena. The Eurographics Association (2005)
2. 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 **9**(4), 185–194 (1998)
3. Cordonnier, G., Braun, J., Cani, M.P., Benes, B., Galin, E., Peytavie, A., Eric, G.: Large Scale Terrain Generation from Tectonic Uplift and Fluvial Erosion. Computer Graphics Forum **35**(2), 165–175 (2016)
4. Deussen, O., Hanrahan, P., Lintermann, B., Měch, R., Pharr, M., Prusinkiewicz, P.: Realistic modeling and rendering of plant ecosystems. In: Proceedings of SIGGRAPH, pp. 275–286 (1998)
5. Deussen, O., Lintermann, B.: Digital design of nature: computer generated plants and organics. Springer Science & Business Media (2006)
6. Emilien, A., Bernhardt, A., Peytavie, A., Cani, M.P., Galin, E.: Procedural generation of villages on arbitrary terrains. The Visual Computer **28**(6-8), 809–818 (2012)
7. Emilien, A., Vimont, U., Cani, M.P., Poulin, P., Benes, B.: Worldbrush: Interactive example-based synthesis of procedural virtual worlds. ACM Transactions on Graphics **34**(4), 106:1–106:11 (2015)
8. Gain, J., Marais, P., Strasser, W.: Terrain sketching. In: Proceedings of Symposium on Interactive 3D Graphics and Games, pp. 31–38 (2009)
9. Gain, J.E., Merry, B., Marais, P.: Parallel, realistic and controllable terrain synthesis. Computer Graphics Forum **34**(2), 105–116 (2015)
10. Génevaux, J.D., Galin, É., Guérin, É., Peytavie, A., Beneš, B.: Terrain generation using procedural models based on hydrology. ACM Transaction on Graphics **32**(4), 143:1–143:13 (2013)
11. Génevaux, J.D., Galin, E., Peytavie, A., Guérin, E., Briquet, C., Grosbellet, F., Benes, B.: Terrain modelling from feature primitives. Computer Graphics Forum **34**(6), 198–210 (2015)
12. Guérin, E., Digne, J., Galin, E., Peytavie, A.: Sparse representation of terrains for procedural modeling. Computer Graphics Forum **35**(2), 177–187 (2016)
13. Han, C., Risser, E., Ramamoorthi, R., Grinspun, E.: Multiscale texture synthesis. In: ACM SIGGRAPH 2008, pp. 51:1–51:8 (2008)
14. Hnaidi, H., Guérin, É., Akkouche, S., Peytavie, A., Galin, É.: Feature based terrain generation using diffusion equation. Computer Graphics Forum **29**(7), 2179–2186 (2010)
15. Kelley, A.D., Malin, M.C., Nielson, G.M.: Terrain simulation using a model of stream erosion. Computer Graphics **22**(4), 263–268 (1988)
16. Lane, B., Prusinkiewicz, P.: Generating spatial distributions for multilevel models of plant communities. In: Proc. of Graphics Interface, pp. 69–80 (2002)
17. Musgrave, F.K., Kolb, C.E., Mace, R.S.: The synthesis and rendering of eroded fractal terrains. Computer Graphics **23**(3), 41–50 (1989)
18. Měch, R., Prusinkiewicz, P.: Visual models of plants interacting with their environment. In: Proceedings of SIGGRAPH, pp. 397–410 (1996)
19. Nagashima, K.: Computer generation of eroded valley and mountain terrains. The Visual Computer **13**(9-10), 456–464 (1998)
20. Prusinkiewicz, P., Hammel, M.: A fractal model of mountains with rivers. In: Proc. Graphics Interface, pp. 174–180 (1993)
21. Smelik, R.M., Tutenel, T., Bidarra, R., Beneš, B.: A survey on procedural modelling for virtual worlds. Computer Graphics Forum **33**(6), 31–50 (2014)
22. Stava, O., Pirk, S., Kratt, J., Chen, B., Mech, R., Deussen, O., Benes, B.: Inverse procedural modelling of trees. Computer Graphics Forum **33**(6), 118–131 (2014)
23. Tropp, J.A., Gilbert, A.: Signal recovery from random measurements via orthogonal matching pursuit. IEEE Transactions on Information Theory **53**(12), 4655–4666 (2007)
24. Tropp, J.A., Gilbert, A.C., Strauss, M.J.: Algorithms for simultaneous sparse approximation: Part i: Greedy pursuit. Signal Process. **86**(3), 572–588 (2006)
25. Wei, L.Y., Lefebvre, S., Kwatra, V., Turk, G.: State of the art in example-based texture synthesis. In: Eurographics State of the Art Report (2009)
26. Zhou, H., Sun, J., Turk, G., Rehg, J.M.: Terrain synthesis from digital elevation models. Transactions on Visualization and Computer Graphics **13**(4), 834–848 (2007)