Map Style Formalization: Rendering Techniques Extension for Cartography

Sidonie Christophe, Bertrand Duménieu, Jérémie Turbet, Charlotte Hoarau, Nicolas Mellado, Jérémie Ory, Hugo Loi, Antoine Masse, Benoit Arbelot, Romain Vergne, et al.

Non-Photorealistic Animation and Rendering
Pierre Bénard and Holger Winnemoeller (Editors)

EXPRESSIVE 2016

Map Style Formalization: Rendering Techniques Extension for Cartography

S. Christophe[1], B. Duménieu[1], J. Turbet[2], C. Hoarau[1], N. Mellado[3], J. Ory[1], H. Loi[4], A. Masse[1], B. Arbelot[4], R. Vergne[4], M. Brédif[1], T. Hurtut[5], J. Thollot[4], D. Vanderhaeghe[3]

[1] Université Paris-Est, IGN, SRIG, COGIT/MATIS, France [2] IPBS
[3] IRIT, Université de Toulouse, CNRS, INPT, UPS, UT1C, UT2J, France
[4] Univ. Grenoble Alpes, CNRS, Inria [5] Polytechnique Montreal

Abstract

Cartographic design requires controllable methods and tools to produce maps that are adapted to users' needs and preferences. The formalized rules and constraints for cartographic representation come mainly from the conceptual framework of graphic semiology. Most current Geographical Information Systems (GIS) rely on the Styled Layer Descriptor and Semiology Encoding (SLD/SE) specifications which provide an XML schema describing the styling rules to be applied on geographic data to draw a map. Although this formalism is relevant for most usages in cartography, it fails to describe complex cartographic and artistic styles. In order to overcome these limitations, we propose an extension of the existing SLD/SE specifications to manage extended map stylizations, by the means of controllable expressive methods. Inspired by artistic and cartographic sources (Cassini maps, mountain maps, artistic movements, etc.), we propose to integrate into our system three main expressive methods: linear stylization, patch-based region filling and vector texture generation. We demonstrate how our pipeline allows to personalize map rendering with expressive methods in several examples.

Categories and Subject Descriptors (according to ACM CCS): [Computer Graphics]: Rendering—Non-photorealistic rendering [Computer Graphics]: Rendering—Rendering Systems [Computer Graphics]: Visualization—Information Visualization

1. Introduction

A geographical map is a representation of the world, real or imaginary. Designing a map is usually a three-stage process. First, mapmakers choose the adequate data for the informations they want to convey during the *data selection* stage. Then, geometrical and semantic levels of detail of the input data are adapted to fit legibility constraints in the *generalization* stage. Finally, mapmakers associate graphic parameters (e.g., symbols) with the data to portray during the *symbolization* stage. Each step has a strong influence on the final map, and it is crucial for mapmakers to tune the involved parameters in order to obtain a result fitting their intention.

In order to create efficient and legible maps, mapmakers rely on theoretical and practical cartographic knowledge [Rob52, Mon91, Mac95, RMM*95]. This knowledge is mainly based on visual perception, and, in particular, on the conceptual framework of graphic semiology that provides rules and constraints to symbolize cartographic information [Mor74, Ber83, Bre94, Mac95].

Geographical Information Systems (GIS) are designed to capture and manage heterogeneous data for analysis and visualisation purposes. GIS usually provide cartographic tools allowing mapmakers to create maps from the stored data. In order to describe stylization rules to create maps (i.e. the symbols and colors of the portrayed data) in an interoperable way, most GIS rely on two Open Geospatial Consortium (OGC) specifications: the Styled Layer Descriptor (SLD) [Lup07] and the Symbology Encoding (SE) [Mül06] specifications. SE provides a formalism to specify the rendering of geographic data, while SLD structures these styles into layers. These specifications aim at creating a sharp distinction between the description of stylization rules – how things should look like – and their system-specific implementation – how a specific style is obtained.

The SLD/SE specifications express stylization rules through a XML-based formalism: style descriptors named `Symbolizer` are assigned to each category of data, for instance categorized by their nature (e.g., roads). According to these rules, several symbolizers can be applied to represent the data and controlled using a set of parameters (e.g., shape, color or size). We will refer thereafter to this style description of a map with SLD/SE as a *style file*.

Although SLD/SE fulfill the main needs of cartography, their expressiveness is limited to the variety and the complexity of the parameters available in a `Symbolizer`. In particular, it is not possible to reproduce complex cartographic or artistic styles based on expressive techniques. For instance, one may want to apply an artis-

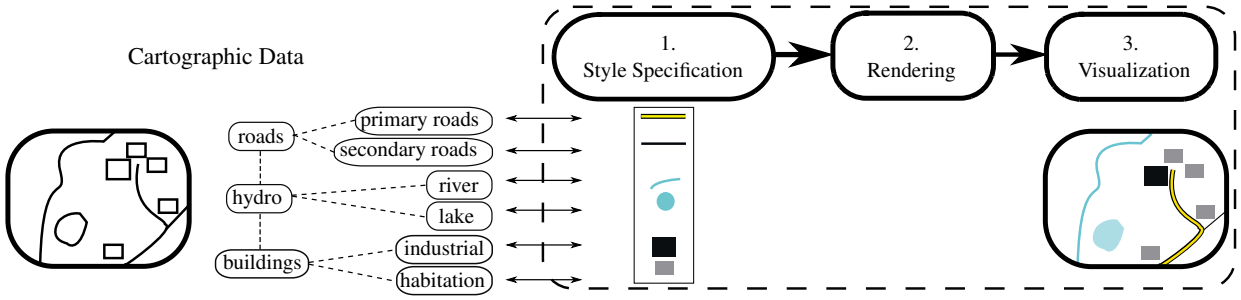


Figure 1: *Focus on the symbolization step, our Map Style Formalization statement (dashed box), in a simplified global map design process. In this work we focus on extending the style specifications (1) and enhancing automatic expressive techniques (2) to be applied to input cartographic data, finally displayed in (3).*

tic style to particular cartographic data (e.g., roads drawn using brushes) or on the entire map (e.g. use textures to mimic watercolor paintings). To date, these artistic designs are made outside of the GIS with raster or vector graphics editors, or by hand on physical supports, and applied as a raster image on the map. For instance Mapbox [Map] or Stamen's watercolored maps [Sta16] rely on pre-computed renderings to obtain expressive renderings. Other map stylization formalisms, i.e. CartoCSS, MapBox GL [Map] or MapServer's Mapfiles [LMD15], are not recognized as standards and thus do not guarantee to be interoperable. SLD/SE have the key advantage of being specifically designed to increase the interoperability of GIS regarding map stylization.

We propose to enrich the expressiveness of style files by extending the specifications with the description of expressive rendering techniques. Using SLD/SE would make the integration of the proposed extension in other GIS easier.

Outline The full map design process is composed of three steps: data selection, generalization, and symbolization. We focus in this paper on the symbolization step, and present an extension of style specification and rendering in GIS. Figure 1 illustrates our approach, with two key components:

- an extension of the existing SLD/SE specifications integrating the description of expressive methods;
- the implementation of expressive methods as GIS external services in the open source GIS software GeOxygene [BBB*12, Geo16].

The expressiveness of our enhanced specifications is illustrated through the making of various maps inspired by hand-drawn historical maps and artistic paintings.

2. Related Work

Designing a map requires a fine control over the different steps of the process in order to fit efficiency and aesthetic constraints. Recent works in computer-assisted cartography show that computer graphic techniques successfully enhance the possibilities in aesthetic map design. As an example, watercolored maps can be obtained by generalizing and stacking multiple layers of geographic data [Sta16].

Other works in computer-assisted cartography focus on mixing classic 2D map renderings with more realistic representations. For instance, Patterson [Pat02] enhances relief realism with realistic illumination. Example-based texture synthesis for map design [JJC12, JJ13], animated waterlining and labeling [SKTD13] are also examples of computer graphics techniques transposed to map rendering. In the meanwhile, expressive maps are considered by other recent works in computer graphics, human computer-interaction and visualization [AS01, GASP08, AMJ*12, KMM*13]. For instance, Isenberg [Ise13] explores geometric simplification, generalization and stylization techniques to provide aesthetic renderings of 2D data.

Despite the wide range of existing techniques, controlling them from usual cartographic stylization formalisms is still an open problem because GIS are limited to simple and pre-defined stylization techniques. In this paper we present an extension of the SLD and SE specifications to declare and control the application of expressive rendering techniques to cartographic data. The style description will be more generic, while preserving the standard requirements.

Some extensions of the SLD/SE specifications have been proposed to enhance its expressiveness. Dietze *et al.* [DZ07] and Rita *et al.* [RBM10] extend SLD/SE to integrate object types to make thematic maps where quantitative and qualitative information is added on top of a map, for instance to represent pie-charts over countries. Neubauer *et al.* [NZ07] add two new `Symbolizers` to adapt the SE for 3D rendering. Nevertheless, GIS still lack the integration of user parameterizable expressive methods in a proper formalism for cartography.

3. Overview

Our main goal is to enable the production of personalized maps. Figure 2 shows several sources of inspiration. Some of the maps were made by hand, either partially (e.g. textures of scree and wooded areas) or totally, and thus cannot be represented by *style files*. We propose tools to model, describe and implement such styles based on existing standard specifications. The SLD/SE specifications are relevant in regards to their interoperability. A proper extension of the SLD/SE specifications remains a challenge for the different communities of mapmakers. To do so, we first present how

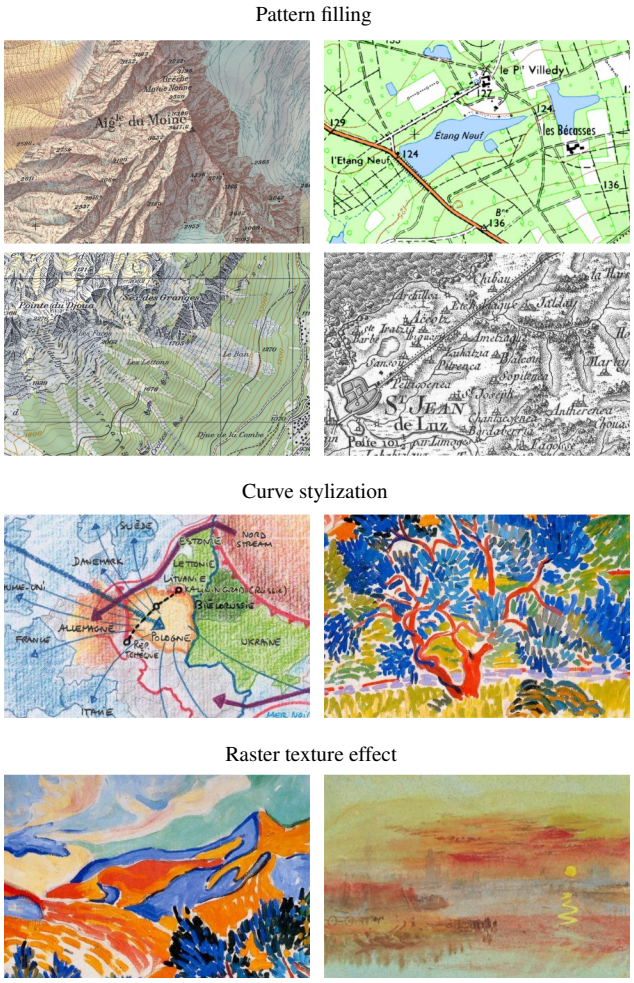


Figure 2: *Inspiration examples, which styles can not be modeled by the SLD/SE specifications: From top to bottom, left to right : Aiguille du Moine, 1:10k, 1952, IGN®; Scan 25, 1:25k, IGN®; National Map 1:25k, SwissTopo®; Cassini map, sheet 139, BnF Paris;* Un rêve, deux cauchemars*, P. Rekacewicz, 2009;* Montagnes à Collioure*, André Derain, 1905, National Gallery of Art, Washington;* Montagnes à Collioure*, A. Derain, 1905, National Gallery of Art, Washington;* The scarlet sunset*, J. Turner, c.1830–40, Tate Britain, London*

we extend the style definition in the SLD specification, and how we implement several map rendering techniques, and apply them on real world examples.

We propose a *service*-oriented SLD extension. New styles are described as services in the SLD, i.e. remote processes that expose some user controllable parameters. While most mapmakers will only use existing services (and consider them as rendering black boxes), more experienced mapmakers or developers should be able to program new services and/or modify the rendering parameters. A key feature of the SLD is to describe a cartographic *style* that can be applied on any cartographic data. Our extension fulfills this constraint, remains backward compatible with the existing schema, and can thus be used seamlessly with existing GIS tools. To illustrate our approach we have chosen to implement three expressive rendering methods that cover the main usual drawing methods found in 2D NPR techniques.

Generation of vector patterns. Many patterns used during a map design are made of a set of vectorial elements distributed in given areas (see Figure 2 top). Typical examples include forest, agricultural zones, sand beach, etc. In particular, mountainous zones and rocky shorelines used to be manually drawn, and subsequently scanned to raster images in order to be manually applied during the map production process. We propose a service that generates new visual variables as vector data to be stylized. The process follows a programmable approach that distributes and modifies vector elements (curves or regions) given the input cartographic data. This service, actually generating cartographic data as input for the symbolization stage, is not yet described in the SLD file, and is thus triggered as a preprocess for now.

Curve stylization. Polylines or strokes are ubiquitous in maps. In the current SE specification, mapmakers manipulate strokes through various properties such as thickness, color, offset, and the type of lines and line joins. To produce more artistic styles, we propose to extend the set of properties the user can control (see Figure 2 middle). We offer to control the curve geometry (e.g. sinuosity) and to manipulate various brushes (e.g. pencil, paint brush).

Raster texture synthesis for region filling. Some textures are easier to synthesize in raster than in vector graphics, like painted regions (see Figure 2, bottom). To produce such visual effects we propose a raster patch-based region filling that distributes and merges user designed patches.

4. Integration of expressive rendering techniques

We focus in this section on the extension of the SLD and SE specifications for expressive rendering techniques. First, we explain how these techniques are integrated as specialized services, referred to as *expressive methods*, that are to be executed by an existing GIS rendering engine. The proposed extension has been implemented in the open source GIS GeOxygene [BBB*12,Geo16]. These services are parameterized by mapmakers by means of new stylization elements added to the SE specification. In order to enhance the control of the rendering at the layer level, we also introduce some extensions to the SLD specification.

4.1. Expressive methods

We define an expressive method as a service which applies an expressive rendering technique to a set of input 2D geometric data. Such a method can create, modify and/or render its input geometry and can expose a set of input parameters to be set by mapmakers. The rendering is done by a GIS rendering engine and is therefore implementation-dependent. Concrete file examples are given in supplementary.

It is important for expressive methods to be described and controllable in the style files, however it is also critical to ensure the style description to be generic and not data-dependent. This is done

by defining a metadata layer that acts as an interface between the stylization description and the service, in a similar fashion than the communication between GIS and Web services (as defined by OGC standards [MP15]). The metadata associated with an expressive method define the following properties:

- the descriptions of each input parameter (shape generation, symbolization and general parameters);
- a way to call the service: e.g., a method name in our implementation.

4.2. SE extension

To enhance the stylization of curves and surfaces, we focus on the declaration of expressive methods in the SE elements dedicated to the symbolization of linear (LineSymbolizer) and surface (PolygonSymbolizer) geometries. Both define the sub-element Stroke dedicated to the stylization of linear features as well as the outline of surface features. The interior of polygons is parameterized in a Fill element inside the PolygonSymbolizer. These two sub-elements are restricted to a pre-defined list of parameters and are thus to be extended to enable the declaration of expressive rendering methods. To do so, Fill and Stroke are extended with the new XML elements ExpressiveFill and ExpressiveStroke. An Expressive* element contains a reference to the expressive method that has to be used on the current visual variable and the values of the exposed parameters needed to execute this method.

4.3. SLD extension

The SLD specification is extended with two new elements. First, we add the possibility to define a Background at the higher level of a SLD file. The background is the canvas of the map, defined as a tiled texture (pre-existing or procedural). We also improve the control over the rendering at the layer level with the addition of LayerFilters, e.g. to control contrast and luminosity.

5. Expressive Rendering Techniques for map design

We have designed, implemented and tested several services as use cases of our extension of SLD/SE based on the GeOxygene GIS rendering engine [BBB*12, Geo16], which already implements the SLD/SE specifications. Technical details about our implementation, and practical style file examples are given in supplementary.

5.1. Generation of vector patterns

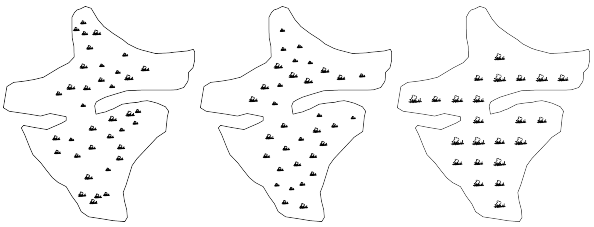
Various vector patterns are used to fill regions in maps. In classic maps these fillings are done with simple elements (dots, circles, crosses) arranged regularly or randomly in a given region. More complex elements and distributions can be found in hand-drawn maps that we would like to represent.

In the expressive rendering literature, a lot of approaches have been designed to synthesize vector textures based on drawn exemplars, such as [BBT*06, IMIM08, HLT*09, MWT11, AKA13, LGH13, XCW14, YBY*13]. These approaches usually target one type of distribution (random or regular), whereas we seek for a method able to control a large spectrum of distributions. Moreover, while the main advantage of by-example methods is that they can be used by artists able to design the exemplar, the resulting synthesized pattern is hard to precisely control.

To increase the controllability and the spectrum of possible synthesized patterns we have chosen to use a programmable approach that allows mapmakers to precisely describe the "recipe" for the vector texture they have in mind. For that we use the approach presented in [LHVT13, LHVT15] where a set of well chosen operators allow to write simple python scripts that describe the texture construction steps. Using the API given in [LHVT15] expert mapmakers can easily write scripts to produce random or regular distributions of any SVG elements. Figure 3 shows a forest zone example with various distributions of trees and their corresponding script. The result of this approach is a set of curves that can be integrated in the full cartographic pipeline as a vector layer. In our current implementation, the generation of vector patterns runs as a preprocess to produce cartographic data to be stylized and rendered by the GIS.

5.2. Curve Stylization

Traditional maps present a large amount of linear visual variables, represented as polylines in geographical databases. To render these polylines, SE lets the user specify the color, transparency, width of a line, and also fine geometric details such as caps and joins. However, one key feature is missing to achieve more artistic styles: the control of the parameterization of the polyline geometry stylization. Therefore, in order to describe artistic styles, like inked,



```
def forest():
    region = ImportSHP("data/shp_jo/forest.shp")
    tree = ImportElt("data/shp_jo/tree.svg", 10.0)
    density = 0.000001
    props = IrregularProperties(density)
    l1 = StripesProperties(0,200)
    l2 = StripesProperties(pi/2,200)
    partition_random  = RandomPartition(props, KEEP_INSIDE)
    partition_uniform = UniformPartition(props, KEEP_INSIDE)
    partition_regular = GridPartition(l1,l2,KEEP_INSIDE)

    def tree_map(f):
        return Scale(MatchPoint(tree,BBoxCenter(tree),Centroid(f)),
               Random(f,2.0,3.0,1))

    tex_random  = MapToFaces(tree_map,partition_random)
    tex_uniform = MapToFaces(tree_map,partition_uniform)
    tex_regular = MapToFaces(tree_map,partition_regular)

    ExportSVG(tex_random , region)
    ExportSVG(tex_uniform, region)
    ExportSVG(tex_regular, region)
```

Figure 3: *Simple example of trees distribution in a forest zone along with the corresponding script. Random (left), uniform (midlle) and regular (right) distribution.*

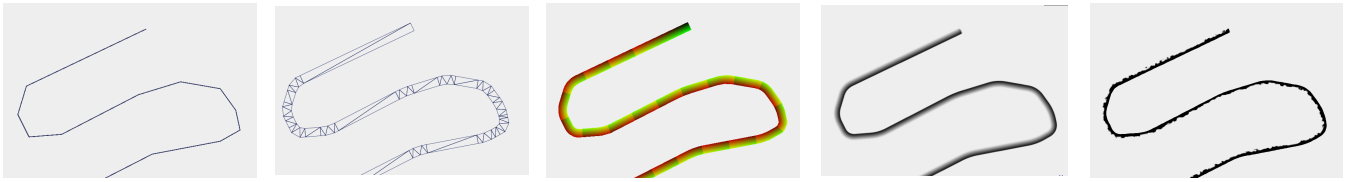


Figure 4: *Curve stylization, from left to right: input polyline, tessellation, texture coordinates, simple texture mapping, color computation using a custom shader.*

pencil, chalk or paint, or even texture mapped strokes, we propose a curve stylization service divided into two main steps: geometry generation and color computation.

Geometry generation This step computes a drawable geometry from the input curve along with a 2D *uv*-parameterization. We implemented the skeletal stroke approach [HLW93, Ase10]: each polyline is tessellated as a triangle strip, with a controllable *uv*-parameterization.

Color computation We propose a simple method to compute a stroke color that simulates the interaction of a medium with the canvas on which the map is drawn. The idea is to consider that the polyline represents the gesture and that a texture mapped along the tessellated polyline represents the pressure of the medium. A user controllable *shader* is then executed to compute the final color taking as input the pressure, a canvas texture, and any parameter defined on the map. This shader computes the final color as the intersection between the pressure texture and the canvas texture.

To increase the range of effects we can produce, we also let the user control the fine geometry of the polyline by tweaking its *uv*-parameterization. However this control does not modify the tessellated geometry.

As the input polylines are of various lengths, we propose to limit the stretching of the mapped textures by making them tileable in their central part. For that we cut the stroke texture in three parts: *start*, *middle* and *end*. We make the middle part tileable (by hand or with the help of a graph cut [KSE*03]) so that the *start* merges with the *middle* and the *middle* merges with the *end*. During the rendering stage, we determine an integer number of repetition of the middle texture to render the polyline and we stretch its *uv*-parameterization accordingly. The repetitive nature of our approach could be lessened with texture synthesis techniques [BCGF10, KS10].

To sum up, we have implemented a curve stylization expressive approach which exposes the following set of user parameters:

- *width*: width of the stroke;
- *texture*: a set of three textures, *start*, *middle* and *end*;
- *paper*: a paper texture;
- *pressure strength*: a scale factor to emphasize the pressure defined by the texture;
- *paper strength*: a scale factor to emphasize the paper;
- *smoothness*: a smoothing parameter to compute the color transparency depending on penetration;
- *jiggling*: jiggle texture according to random noise;
- *width variation*: varying width according to random noise.

One can develop another expressive approach and define another set of user controllable parameters.

5.3. Raster texture synthesis for region filling

Some region-filling textures in standard maps are drawn using tileable raster textures. When dealing with hand-drawn maps, such tilings are not visible and textures might be very continuous (painted or watercolored areas). To simulate such a behavior, we propose a raster filling method that distributes and merges small raster patches provided by mapmakers. As a rule of thumb, a patch is a small part of an image that is more or less repeated to fill an area. The number and size of patches will influence the perceived repetitivity of the final textured region.

These patches are first either drawn by hand or selected in existing maps and are then distributed in the region one by one until no more free space is available. At each step a random position is chosen and a patch is selected according to a user defined probability. The probability of selecting a given patch and its parameters (size, orientation) are given by user-defined functions. These functions can for example rely on the position in the region or a distance field.

The way patches are merged plays an important role on the final result. Two approaches have been implemented: simple alpha blending and graph cut blending [KSE*03]. Graph cut is more likely to preserve contrast while alpha blending smoothes transitions between patches. Since graph cut requires a considerable computation time (about 20s for 600 by 600 pixels), results are cached on disk.

To summarize, this expressive approach exposes the following user parameters:

- a set of patches, identified by a name and an image (with alpha channel);
- for each patch, a probability of selection, optionally restricted to a given distance;
- a distance function selection: distance to region border;
- an orientation function selection: perpendicular or parallel to border, oriented in image space;
- a merging technique: alpha blending or graph cut.

Again this set of parameters is strongly linked to the way we have implemented our black box. Expert mapmakers or developers should define their own black box with the appropriate set of exposed parameters.

These curve stylization and raster texture for region filling are two examples of the related services that can be controlled from our SE extension. Other services can be integrated in the proposed expressive cartographic pipeline as long as they provide a set of parameters to be controlled from a style file.

6. Map Style design

This section presents resulting maps obtained using our global approach to describe different expected styles with the help of the implemented expressive methods. We demonstrate the relevancy to manage a service generating cartographic data, through the example of a mountain style, and the potential of the extension of SLD/SE through two examples of styles, i.e. a Cassini style and a watercolor style.

For each style we highlight some feedback provided by two categories of users, the map producer and the expert mapmaker or developer. The former validates the suitability of the designed maps whereas the latter evaluates the expressiveness of the style files and the resulting maps.

6.1. Mountain Style

The first map, mentioned as `Mountain` and shown Figure 5, represents the "Aiguille du Moine" at a 1:10k scale. The original map is considered as very expressive: the perception of relief is aesthetic and efficient. However, this style cannot be automatically reproduced in current GIS because it relies on lines and hatches to represent relief or slope, and also on complex and ill-defined stylization rules.

We used the pattern generation service presented in Section 5.1 to distribute vector primitives in the `Mountain` map (Figure 5). We have chosen to use hatching primitives, and manually tuned the associated parameters (e.g. orientation, density) to mimic the original style. Two cartographic data are generated as hatching primitives varying in density and length: a first type filling the faces, related to their slopes and orientations; a second type simulating the accumulation of ink at the top of edges, related to height and slope. Cartographic data are then stylized by a curve stylization in order to simulate the hand-drawn design.

The resulting hatching layers are composited over the other layers of the original map (various colors of shading relief, toponymy, contour lines and glacier, Figure 5) for comparison with the original map. Some examples of the use of various brushes are given in the supplementary.

6.2. Cassini Style

We consider the "map of Cassini", a famous topographic map from the XVIIIth century, mentioned as `Cassini` in the following. Although the style of the map of Cassini is widely known amongst mapmakers and thus is easily recognized, it cannot be automatically reproduced in current GIS because of the lack of suitable expressive methods to simulate old cartographic practices. Indeed, the map of Cassini is entirely hand-drawn with multiple drawing techniques such as stamping, etching, and ink painting. We focus here on the 139th sheet of the black and white re-edition of the map of Cassini published in 1815 [CdT15].

We use our curve stylization service to stylize roads and rivers for `Cassini`. The brush textures are taken from the original map to stay as close as possible to its hand-drawn style.

We use our raster filling service for `Cassini` to fill the the sea area: the distribution of wave patches is guided by the distance from the coastline. Patches come from the map of Cassini itself, aiming at enhancing the hand-drawn effect.

6.3. Watercolor Style

We demonstrate how our approach enables to design `Watercolor` maps. `Watercolor` improves the aesthetic of the map and softens its global visual impact. We assume that such an artistic stylization can be achieved without degrading the perception of the geographical space.

For `Watercolor` we take inspiration from [BKTS06]. Regions are filled with a flat color and overlayed by a procedural noise raster layer. In order to add an edge darkening effect, we have designed a specific brush stroke which produces a grayscale gradient on the region contours and is also blended with the procedural noise. Finally, thanks to the SLD, we can control which data have to be watercolorized. In our experiment we have chosen to render linear data (roads and rivers) with an ink brush stroke on top of the watercolorized regions (Figure 7).

Users feedbacks

Resulting maps have been presented to institutional map producers. Despite being different from the original maps, `Mountain` and `Cassini` maps provide satisfactory feedbacks: both convey correctly the characteristics of the related geographical space properties and the old cartographic practices used at the time. The `Watercolor` map is considered as a pleasant initiative in order to highlight data in a given space.

The challenge for the `Mountain` map is related to relief perception: the proposed stylized hatching primitives are relevant because conveying it, without seeing exactly the textures design themselves at the map scale of 1:10k. The simulation of accumulated ink has been greatly highlighted because playing an important role to consider relief, most of all in the dark faces where a simple layer of hatches is insufficient to consider multi-scale faces.

The challenge for the `Cassini` map is related to all depicted objects in the map: an exact reproduction of Cassini maps is very difficult to attempt because visual information is very cluttered and requires many types of expressive techniques that are still to be integrated in our system or still to propose, for instance the complex rendering of the relief. Our result is still satisfactory because it conveys the design techniques of the Cassini maps well: most of all the textures, coming from both raster filling and pattern generation services, have been considered as relevant to reproduce the Cassini style. A proposed improvement would be to make it more "worn", because strokes are still too clean.

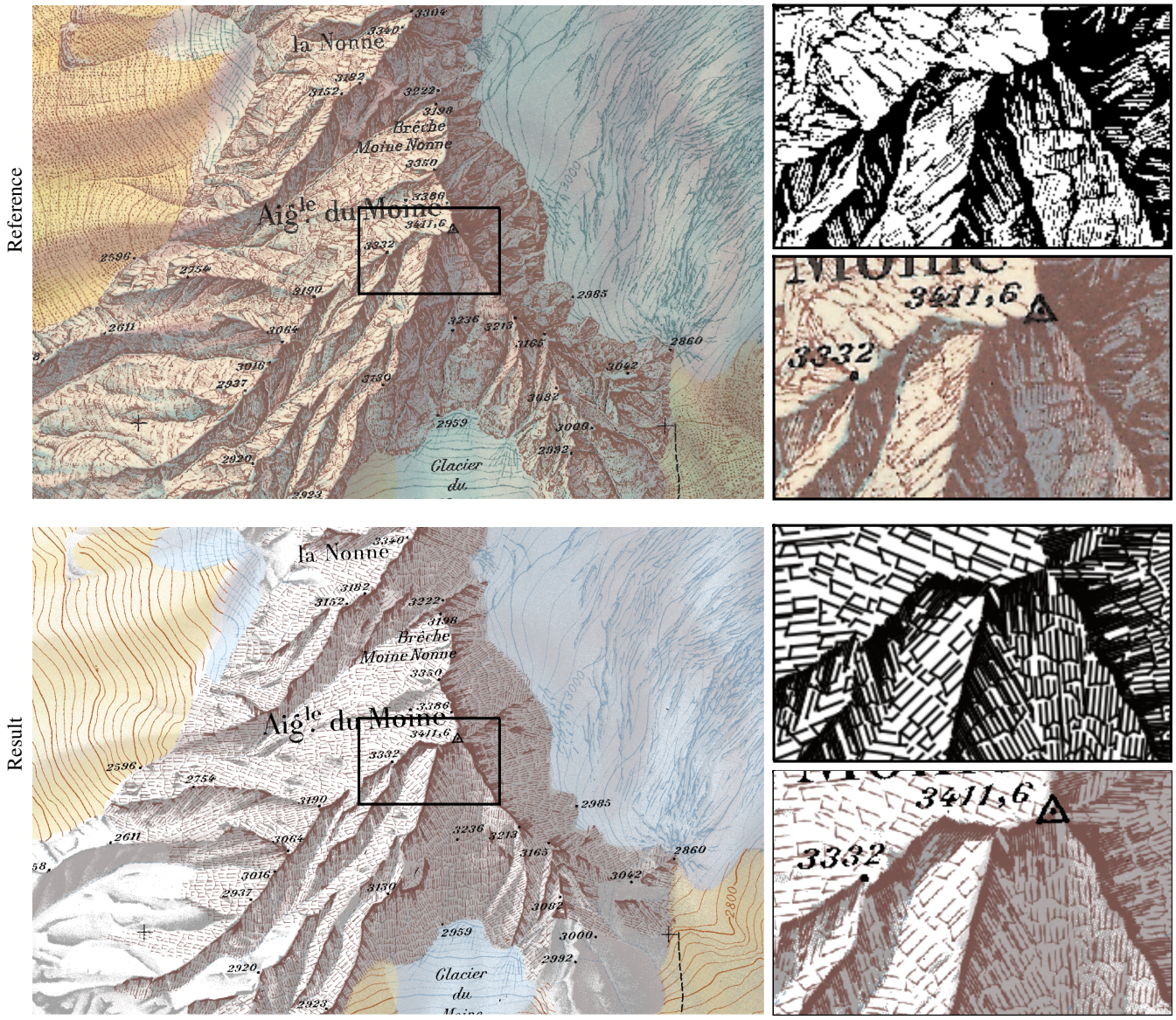


Figure 5: *Reference and Resulting Mountain map "Aiguille du Moine", 1:10k scale: extracts of reference (first line) and resulting rocky areas (second line): on the right, zooms on, first the hatching primitives, second the stylized same ones. For a fair comparison, we provide resulting map at a resolution similar to the reference map.*

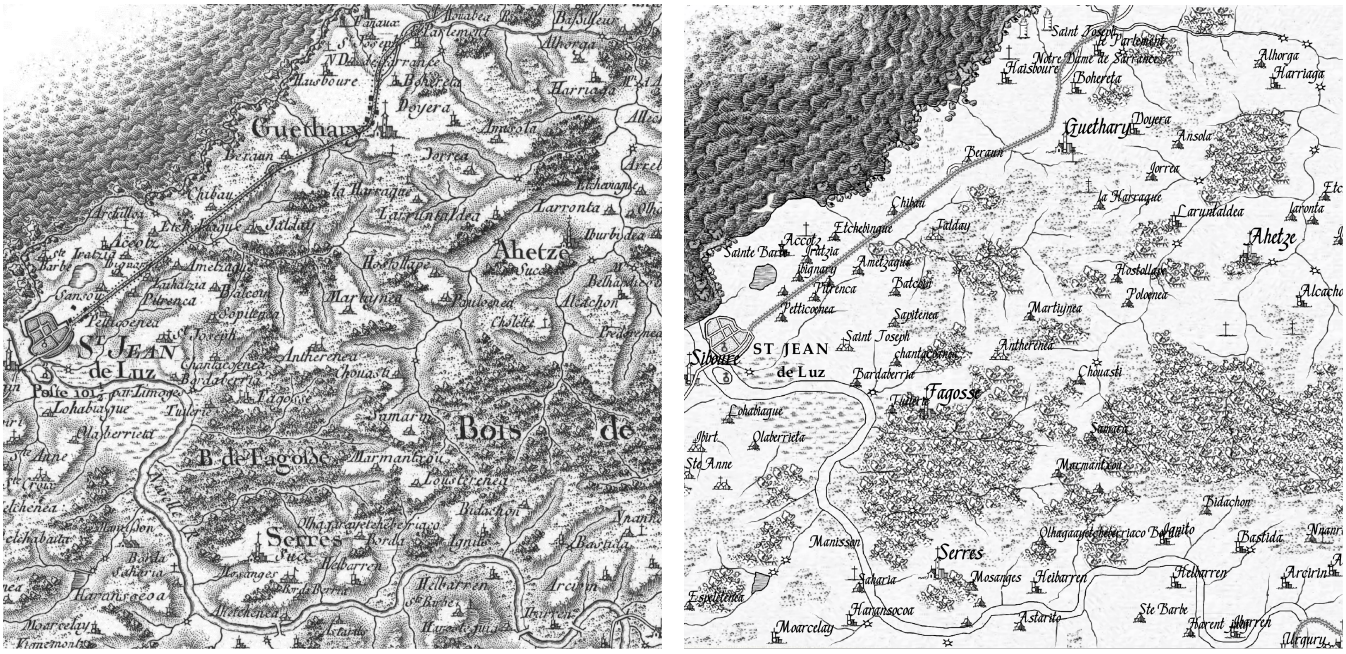


Figure 6: *Comparison of the reference Cassini map (sheet 139, Bayonne, 1:86 000 scale, BnF, Paris) on the left and the resulting Cassini styled map on digitalized historical data on the right.*

The challenge for the `Watercolor` map is to manage aesthetic effects on cartographic data and on a map, according to users' requirements, in order to add value to a geographical space. This technique would be very useful for a general audience. Moreover, this technique would be very useful to soften some data and make other data or phenomena salient (uncertainty effects, for instance).

Expert developers in map design highlight that the extension of the SLD/SE specifications is suitable to their expectations to make personalized or institutional maps, first because guaranteeing interoperability between cartographic systems and second because expressiveness has been augmented: the resulting maps are a great evidence of what could be done, and provide clues for what could be personalized in the future.

Our proposition is very useful to easily manipulate those rendering techniques in a GIS context, in particular to parameterize a rendering technique or to mix several rendering techniques to make a personalized map. Moreover, the pattern generation service is also relevant actually to generate various cartographic data, especially any patterns for any regions and thus for instance, various land cover uses, to be stylized later on. Variability of our expressive rendering methods is illustrated in the supplementary.

7. Conclusions

In this paper, we show how to extend existing GIS standards and enrich the expressiveness of the map style files, with the purpose to help users to make personalized maps. To do so, we extend the SLD/SE specifications by adding expressive rendering methods, designed as *services*. These services are external programmable components, which can be used by mapmakers by tweaking exposed parameters, and also customized and extended by expert users and programmers.

We demonstrate our approach using three expressive rendering techniques, giving control over vector pattern generation, curve stylization and raster texture synthesis for region filling. We demonstrate the effectiveness of our approach on several examples, to define styles inspired by existing maps drawn by hand (rocky mountain, Cassini maps), and produce artistic maps inspired by painting techniques (watercolor). Our results show that the SLD/SE extension is generic enough to model and control customized rendering techniques for map design. In addition, our *service*-oriented stylization integrates seamlessly with existing GIS softwares, and follows original SLD/SE specifications. First users, map producers and expert developers in map design, provide satisfactory feedbacks on such results. We believe that our extension opens standardized map stylization to a wide audience, from the general public to expert mapmakers and GIS programmers. Therefore, we plan a user study to evaluate the proposed map styles regarding users needs and preferences.

Expressive map stylization opens several challenges that we plan to study in future works. In the current state, the vector pattern generation is done in pre-process, and should be integrated in our global stylization pipeline. In addition, an expressive rendering method can take any parameter as input, which could for instance be the cartographic data themselves. Such a way of thinking has already been explored by the NPAR community, by defining the style as a transfer function between a scene properties and some style attributes [Dur02].

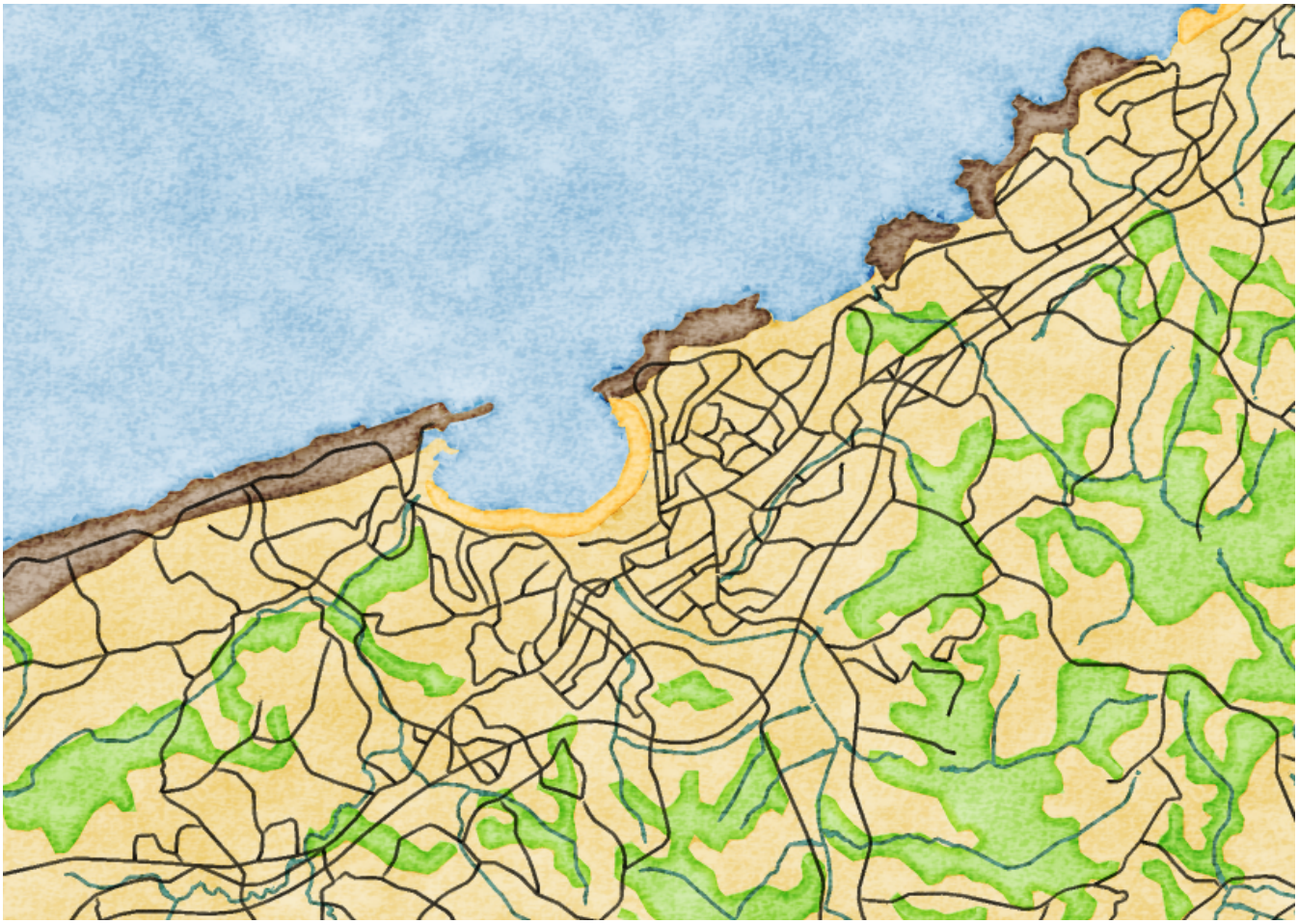


Figure 7: *Watercolored map on cartographic data IGN®, Bayonne 1:100k scale*

Our work focuses on stylization, which occurs at the end of the map design pipeline. In future works, we also plan to study how to increase the expressiveness of other steps of the process (data selection, generalization), and to model the interaction between these steps. For instance, the stylization of a line may require to change its geometry, which could impact the position of other elements on the map (e.g. toponyms), and thus affect the full pipeline.

Acknowledgments

This work was supported by a grant overseen by the French National Research Agency (ANR) as part of the Mapstyle project [ANR-12-CORD-0025].

References

[AKA13] ALMERAJ Z., KAPLAN C. S., ASENTE P.: Patch-based geometric texture synthesis. In *Proc. CAE* (2013), ACM, pp. 15–19. doi:10.1145/2487276.2487278. 4

[AMJ*12] AFZAL S., MACIEJEWSKI R., JANG Y., ELMQVIST N., EBERT D. S.: Spatial Text Visualization Using Automatic Typographic Maps. *TVCG 18*, 12 (Dec. 2012), 2556–2564. doi:10.1109/tvcg.2012.264. 2

[AS01] AGRAWALA M., STOLTE C.: Rendering effective route maps: Improving usability through generalization. In *Proc. SIGGRAPH* (2001), ACM, pp. 241–249. doi:10.1145/383259.383286. 2

[Ase10] ASENTE P. J.: Folding avoidance in skeletal strokes. In *Proc. SBIM* (2010), pp. 33–40. doi:10.2312/SBM/SBM10/033-040. 5

[BBB*12] BUCHER B., BRASEBIN M., BUARD E., GROSSO E., MUSTIÈRE S., PERRET J.: GeOxygene: Built on Top of the Expertise of the French NMA to Host and Share Advanced GIS Research Results. In *Geospatial Free and Open Source Software in the 21st Century*, Bocher E., Neteler M., (Eds.), Lecture Notes in Geoinformation and Cartography. Springer Berlin Heidelberg, 2012, ch. 2, pp. 21–33. doi:10.1007/978-3-642-10595-1_2. 2, 3, 4

[BBT*06] BARLA P., BRESLAV S., THOLLOT J., SILLION F., MARKOSIAN L.: Stroke pattern analysis and synthesis. *Computer Graphics Forum 25*, 3 (2006), 663–671. Proc. EG. doi:10.1111/j.1467-8659.2006.00986.x. 4

[BCGF10] BÉNARD P., COLE F., GOLOVINSKIY A., FINKELSTEIN A.: Self-similar texture for coherent line stylization. In *Proc. NPAR* (2010), ACM, pp. 91–97. doi:10.1145/1809939.1809950. 5

[Ber83] BERTIN J.: *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, Madison, 1983. 1

[BKTS06] BOUSSEAU A., KAPLAN M., THOLLOT J., SILLION F.: Interactive watercolor rendering with temporal coherence and abstraction. In *Proc. NPAR* (2006), ACM. doi:1124728.1124751. 6

[Bre94] BREWER C. A.: *Visualization in modern cartography*. Elsevier Science, 1994, ch. 7 - Color use guidelines for mapping and visualization, pp. 123–147. 1

[CdT15] CASSINI DE THURY C.-F.: Carte générale de la France. 139, [Bayonne. New edition.]. Sheet 139 / [created under the direction of César-François Cassini de Thury], 1815. URL: http://catalogue.bnf.fr/ark:/12148/cb408607946. 6

[Dur02] DURAND F.: An invitation to discuss computer depiction. In *Proc. NPAR* (2002), ACM, pp. 111–124. doi:10.1145/508530.508550. 8

[DZ07] DIETZE L., ZIPF A.: Extending OGC Styled Layer Descriptor (SLD) for thematic cartography. In *Proc. Symp. on LBS and Telecartography* (2007), Springer H., (Ed.). 2

[GASP08] GRABLER F., AGRAWALA M., SUMNER R. W., PAULY M.: Automatic Generation of Tourist Maps. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 1+. URL: http://dx.doi.org/10.1145/1360612.1360699, doi:10.1145/1360612.1360699. 2

[Geo16] Geoxygene. http://oxygene-project.sourceforge.net/, 2016. 2, 3, 4

[HLT*09] HURTUT T., LANDES P.-E., THOLLOT J., GOUSSEAU Y., DROUILLHET R., COEURJOLLY J.-F.: Appearance-guided synthesis of element arrangements by example. In *Proc. NPAR* (2009), ACM, pp. 51–60. 4

[HLW93] HSU S. C., LEE I. H. H., WISEMAN N. E.: Skeletal strokes. In *Proc. UIST* (New York, NY, USA, 1993), ACM, pp. 197–206. doi:10.1145/168642.168662. 5

[IMIM08] IJIRI T., MĚCH R., IGARASHI T., MILLER G.: An example-based procedural system for element arrangement. *Computer Graphics Forum 27*, 2 (2008), 429–436. Proc. EG. 4

[Ise13] ISENBERG T.: Visual Abstraction and Stylisation of Maps. *The Cartographic Journal 50*, 1 (2013), 8–18. doi:10.1179/1743277412Y.0000000007. 2

[JJ13] JENNY H., JENNY B.: Challenges in adapting example-based texture synthesis for panoramic map creation: a case study. *Cartography and Geographic Information Science 40*, 4 (2013), 297–304. 2

[JJC12] JENNY H., JENNY B., CRON J.: Exploring transition textures for pseudo-natural maps. In *GI_Forum 2012: Geovisualization, Society and Learning* (2012), Jekel T., Car A., Strobl J., Griesebner G., (Eds.), Wichmann, pp. 130–139. 2

[KMM*13] KIM S., MACIEJEWSKI R., MALIK A., JANG Y., EBERT D. S., ISENBERG T.: Bristle Maps: A Multivariate Abstraction Technique for Geovisualization. *TVCG 19*, 9 (Sept. 2013), 1438–1454. doi:10.1109/tvcg.2013.66. 2

[KS10] KIM M., SHIN H. J.: An example-based approach to synthesize artistic strokes using graphs. *Computer Graphics Forum 29*, 7 (2010), 2145–2152. doi:10.1111/j.1467-8659.2010.01802.x. 5

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3 (July 2003), 277–286. doi:10.1145/882262.882264. 5

[LGH13] LANDES P.-E., GALERNE B., HURTUT T.: A shape-aware model for discrete texture synthesis. *Computer Graphics Forum 32*, 4 (2013), 67–76. Proc. EGSR. 4

[LHVT13] LOI H., HURTUT T., VERGNE R., THOLLOT J.: Discrete Texture Design Using a Programmable Approach. In *Siggraph Talks* (July 2013), ACM, p. Article No. 43. doi:10.1145/2504459.2504513. 4

[LHVT15] LOI H., HURTUT T., VERGNE R., THOLLOT J.: *A Programmable Model for Designing Stationary 2D Arrangements*. Research Report RR-8713, Inria - Research Centre Grenoble – Rhône-Alpes ; INRIA, Apr. 2015. URL: https://hal.inria.fr/hal-01141869. 4

[LMD15] LIME S., MCKENNA J., DOYON J.-F.: Mapserver 7.0.0 documentation, 2015. Accessed: 2016-02-12. URL: http://www.mapserver.org/mapfile/. 2

[Lup07] LUPP M.: Styled layer descriptor implementation specification. *Open Geospatial Consortium Document Number: OGC 05-078r4, Version: 1.1.0* (jun 2007). URL: http://www.opengeospatial.org/standards/sld. 1

[Mac95] MACEACHREN A. M.: *How maps work: Representation, Visualization, and Design*. Guilford Publications, New York, 1995. 1

[Map] MAPBOX: Mapbox studio gallery. Accessed: 2016-02-12. URL: https://www.mapbox.com/gallery/. 2

[Mon91] MONMONIER M.: *How to lie with maps*. University of Chicago Press, 1991. 1

[Mor74] MORRISON J. L.: A theoretical framework for cartographic generalization with the emphasis on the process of symbolization. *International Yearbook of Cartography 14* (1974), 115–127. 1

[MP15] MUELLER M., PROSS B.: OgcⓇwps 2.0 interface standard corrigendum 1. *Open Geospatial Consortium Document Number: 14-065, Version: 2.0.1* (oct 2015). URL: http://www.opengeospatial.org/standards/wps. 4

[Mül06] MÜLLER M.: Styled layer descriptor implementation specification. *Symbology Encoding Implementation Specification, version 1.1.0, OpenGIS Implementation Specification, 05-077r4, OpenGIS Consortium* (jul 2006). URL: http://www.opengeospatial.org/standards/symbol. 1

[MWT11] MA C., WEI L.-Y., TONG X.: Discrete element textures. *Transactions on Graphics 30*, 4 (July 2011), 62:1–62:10. Proc. SIGGRAPH. doi:10.1145/2010324.1964957. 4

[NZ07] NEUBAUER S., ZIPF A.: Suggestions for extending the ogc styled layer descriptor (SLD) specification into 3d. In *Towards Visualization Rules for 3D City Models, Urban Data Management Symposium. UDMS* (2007). 2

[Pat02] PATTERSON T.: Getting Real: Reflecting on the New Look of National Park Service Maps. *Cartographic Perspectives*, 43 (2002), 43–56. doi:10.14714/CP43.536. 2

[RBM10] RITA E., BORBINHA J., MARTINS B.: Extending SLD and SE for cartograms. In *Proc. GSDI* (2010), vol. 12. 2

[RMM*95] ROBINSON A. H., MORRISON J. L., MUEHRCKE P. C., GUPTILL S. C., KIMERLING A. J.: *Elements of cartography, Sixth Edition*. John Wiley & Sons, Inc., 1995. 1

[Rob52] ROBINSON A. H.: *The Looks of Maps*. Madison: University of Wisconsin Press, 1952. 1

[SKTD13] SEMMO A., KYPRIANIDIS J. E., TRAPP M., DÖLLNER J.: Real-time rendering of water surfaces with cartography-oriented design. In *Proc. CAE* (2013), ACM, pp. 5–14. doi:10.1145/2487276.2487277. 2

[Sta16] STAMEN: Stamen map design, 2016. accessed 01/02/2016. URL: http://maps.stamen.com. 2

[XCW14] XING J., CHEN H.-T., WEI L.-Y.: Autocomplete painting repetitions. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 172:1–172:11. doi:10.1145/2661229.2661247. 4

[YBY*13] YEH Y.-T., BREEDEN K., YANG L., FISHER M., HANRAHAN P.: Synthesis of tiled patterns using factor graphs. *ACM Transactions on Graphics 32*, 1 (Feb. 2013), 3:1–3:13. doi:10.1145/2421636.2421639. 4