2010 18th Pacific Conference on Computer Graphics and Applications

Sample-Based Synthesis of Illustrative Patterns

Vladimir Alves dos Passos*, Marcelo Walter†, Mario Costa Sousa‡,
*Centro de Informática, UFPE, Recife, Brasil
Email: vap2@cin.ufpe.br
†Instituto de Informática, UFRGS, Porto Alegre, Brasil
Email: marcelo.walter@inf.ufrgs.br
‡Department of computer Science, University of Calgary, Calgary, Canada
Email: smcosta@ucalgary.ca

***Abstract*—We present an improved method for synthesis of patterns defined as 2D collection of vector elements. Current solutions to this problem rely on triangulation of the input space or statistical measures of the sample to drive the synthesis step. We propose a method applicable to colored textures, from regular to stochastic, and which provides control over local density of elements. Also, our results show the same visual quality as previous works. The sample is segmented into groups of similar elements and we use a novel local neighborhood distance metric to compare distinct and incomplete neighborhoods. This metric does not ignores existing unpaired elements. The main synthesis loop consists of a procedural growth, where seeds are replaced by a reference to an element from the sample, generating new seeds until the target space is filled.**

***Index Terms*—texture synthesis; distribution patterns; vectorial textures;**

I. Introduction

Texture Synthesis [1], [2] has been used for more than a decade now to render arbitrarily large images of textures from samples with great success. More recently, the same ideas have been applied to a new class of textures of vectorial elements [3], [4], [5], which we will call patterns of distributions, or only patterns.

These patterns present a new challenge to synthesis methods from image samples, since they are made of collections of vector elements described and traditional texture synthesis is not directly applicable. The input data is described with elements, instead of pixels. The attractiveness of solutions to this problem comes from using as input real patterns, usually hand-drawn. The output pattern should have the same appearance as the input sample, but with additional functionalities such as arbitrary resolution, controls for tasks such as filling a large area, and control over features such as density, scale, and others not easily done in manual systems.

As the main contribution of this work, we present an improved solution for synthesis of this type of patterns inspired by traditional pixel and patch-based texture synthesis. The solution does not require any triangulation of the sample or expensive gathering of statistical data, and yet allows the control of the local density of elements. Also, the method provides good results for a large variety of patterns, from regular to stochastic, and for colored or black and white textures, as illustrated in Figure 1. We collect perceptually-meaningful distribution data about the vector-described elements (a seminal idea from [3]) and use this data in the procedural growth step to check if an element should be inserted onto the target pattern. The 'best' element to fill the target pattern will minimize a local neighborhood metric, much as raster texture synthesis minimizes a color $L2$ norm in most algorithms.

II. Related Work

A. Texture Synthesis from Samples

Texture synthesis from samples is an excellent solution for building textures which are not only visually similar to the given sample but also can be built at user-defined resolutions. Non parametric raster texture synthesis from a sample is the traditional approach, where the basic elements are the pixels [1], [2], [6]. The texture is considered a realization of a Markov Random Field, such that the color of the pixel should be completely determined by its neighborhood. The main problem is to determine the shape of the neighborhood and the rule that relates the pixel with its surroundings. However, the semantics of the texture being synthesized remains unknown, since at the pixel-level only the color defines the basic element and the lattice structure is strictly defined.

Later work extended the idea from combining pixels to patches of the sample [7], [8], [9], [10]. The final texture is formed by joining together pieces or blocks of the original sample, with a RGB metric for selecting the best matches. Later work introduced arbitrarily shaped patches [11], new methods for sewing the patches (graph cuts) [12], and wavelets as the proximity criterion among patches [13]. There has been a lot of activity in this area in the last few years and a good review of the work so far is given in the Siggraph 2007 course presented by Kwatra and colleagues [14].

B. Arrangement Pattern Synthesis

Stroke pattern synthesis was studied in the past to generate stipple drawings [15], pen and ink representations [16], [17], engravings [18], and painterly rendering [19]. Those systems, however, rely on generative rules chosen by the user or brought from traditional drawing techniques. Kalnins *et al.* [20] present a method for synthesizing stroke offsets to generate new strokes similar to those appearing in the supplied sample. Similar problem is addressed by Hertzman *et al.* [21] and Freeman *et al.* [22], but neither method reproduces the

978-0-7695-4205-8/10 $26.00 © 2010 IEEE
DOI 10.1109/PacificGraphics.2010.22

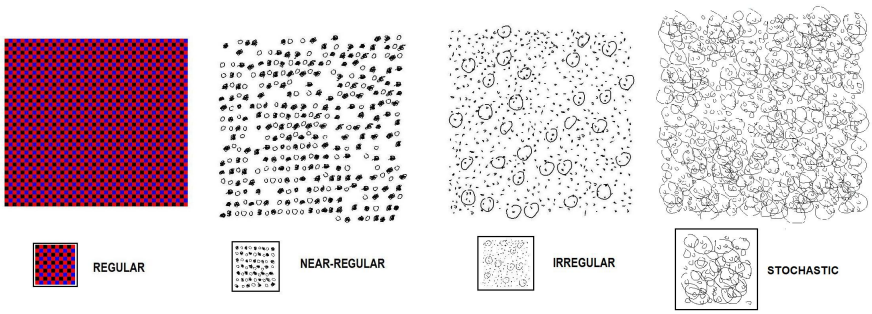

Fig. 1. Illustrative patterns synthesized by our method.

interrelation of strokes within a pattern. Jodoin *et al.* [23] address the problem of synthesizing hatching strokes arranged linearly along a path.

Barla *et al.* [5] address the more general problem, synthesizing two-dimensional patterns of distributions. Their method is divided into two steps: input analysis and synthesis. In their analysis step, a collection of curves given as input is processed and strokes are grouped together into elements (such as circles, spirals, line segments, etc) that compose the sample. The distribution is triangulated using the center of each element as a vertex. In the synthesis step, such elements are copied many times to the vertices of a regular triangulation artificially generated. Random displacements on the final position of the elements are included to improve the overall visual result. Their main limitation is in the analysis step, where only well behaved textures can be used, unlike many of the irregular patterns presented on later work and also on this paper. Also, there is not any control over the local density of elements. Ijiri *et al.* [4] propose a similar method, although they take the input as a collection of already defined elements. The final texture is constructed from a procedural growth, starting from one single seed, in order to simulate the same process that created the texture. The process is further refined trough user interaction, with a good visual match among samples and results, however the process is not completely independent of the user and no control of local density is provided. Hurtut *et al.* [3] present a statistical approach, where the texture is modeled as a function that can be replicated by the algorithm used on the synthesis step, generating a different distribution but with similar parameters. The input is also a collection of elements, and the range of images that can be synthesized is larger than in previous work, with some issues to synthesize regular patterns. Their work greatly improves previous results, since they use a perception-based gathering of geometric properties from the sample, and use this in a multi-point statistical synthesis step.

We share the same goals and overall approach of these three works [3], [4], [5], merging procedural growth of texture and local density control and obtaining visually similar results as previous works, but also, our method is applicable to colored textures and good results are generated for a wide range of textures, such as regular, near regular, irregular and stochastic.

III. Classification and Synthesis

A. Overview

In this section, we detail our method, which has two main parts: classification and synthesis. The user will provide a sample input image and the desired resolution of the resulting pattern. The sample image is assumed to be a collection of elements, each one described by vector data. In the classification part, the elements from the input sample are grouped together into collections according to similarity measures, such that ideally, any two elements of the same group should be visually similar. From this resulting data, important features, illustrated in Figure 2, are evaluated. The features are used to control the synthesis and correspond to the minimal distance between pairs of distinct elements from two groups.

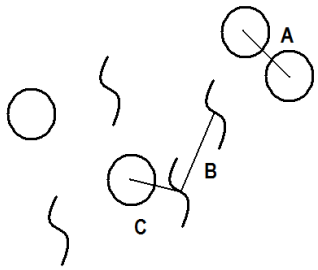

Fig. 2. Features evaluated for an illustrative example. Here, the elements were grouped into two distinct groups, curve and circle. In this case, three features are evaluated. The smaller distance between distinct circles (A), distinct curves (B), and circles and curves (C).

The synthesis starts by dividing the target output image into squared regions of appropriate size. A recursive procedure starts by visiting a random region and placing a seed in a random position on its interior. This seed is replaced by a chosen element from the sample and new seeds are generated accordingly to the relative position of the neighborhood of this element on the sample. The list of seeds grows iteratively until the target space is completely visited, in other words, until there are no unvisited regions left. During the synthesis, we keep a list of seeds located in the target image space. This list is FIFO, i.e. the first seed from this list is extracted and replaced by a reference to one element from the sample, inserting new seeds at the end of the list. Therefore, the

texture grows in a spiral order starting from the central seed. If eventually the list of seeds becomes empty, a new seed is randomly created into a random non-visited region. The growth is illustrated in Figure 3, and shares many similarities with the method proposed by [4]. The main difference is the absence of underlying triangulation of the arrangement, which will provide the control over the local density as will be explained later.

B. Input classification

Unlike raster texture synthesis, our approach aims the arrangement of elements located in the bidimensional space. Initially, such elements are ungrouped and defined only by their own vector data. Thus, we must define and evaluate relevant characteristics that will be used to classify the elements into distinct similarity groups. The number of those characteristics should be the least necessary to fully identify each group the pattern, much as a human observer would use a limited number of characteristics to naturally identify groups.

We use the same approach of [3], classifying elements within two stages. The first stage corresponds to the classification by area, and the second stage corresponds to the remaining characteristics, reduced by PCA to avoid high dimensionality issues. We included in the second stage the grayscale conversion of the RGB color of each element, in order to be able to process colored textures as well. Also, we made a simplification of the process. Instead of using *A Contrario* method to find relevant modes on the histogram, we use a simple 5 bins histogram, where each non-empty bin will become a group itself. Five bins were chosen as an initial guess due to the visual observation of the histograms generated for all tested samples. In practice, this number was enough to classify all samples with straightforward implementation, although for a few patterns there are over categorizations.

Here, we present the classification step. First, the 5 bins histogram is evaluated from the areas of each element. If the bin is empty, nothing is done. When the bin contains 12 elements or less, the same number used on previous work [3], it is considered a group by itself. If the bin contains more than 12 elements, those elements are further classified. On the second classification, dimensionality reduction with PCA is applied on the remaining characteristics, including grayscale color, and the histogram of the resulting data is evaluated. Again, the histogram contains 5 bins, and each non-empty bin will become a distinct group. Figure 4 shows the steps of classification for one sample. It was verified that this simplified procedure is enough to obtain good results not only with the samples from previous work, but also new complex ones, as it will be shown later in the results.

C. Spatial features of the pattern

During the positioning of elements on the target space, spatial features of the input sample are used to guide the synthesis. Those features represent spatial restrictions that the new pattern must follow in order to maintain the overall appearance of the input sample. In other methods, those features are codified into the triangulation of the arrangement or in the parameters of a statistical model. In our method, they are defined by the following functions:

$$d(x, A) = min\{d(x, y)|y \in A, y \neq x\}$$

$$d(B, A) = min\{d(x, A)|x \in B\}$$

where $d(x, y)$ is the Euclidean distance between the center of two elements x and y, and A and B are groups. We evaluate the function d(A,B) for each pair of groups and also, we evaluate all the $d(x, A)$ for each element x. Each element x will contain its own table of features to be used later, during synthesis. More features could be added, but this would increase the computational cost, and also make the result less controllable. In this work, these features are used to control the local density of the texture, as we will show later.

D. Covering The Target Space

On raster texture synthesis, the target space is defined by the resolution and the synthesis is concluded when all pixels have been visited. Based on this idea, we set the target output size as a parameter, but still remains the problem of knowing when the synthesis is complete. We artificially create an underlying grid by dividing the area of the target texture as a matrix of squared regions, initially marked as unvisited, and use them as our 'pixels'. We define the edge of each region as the minimal distance between elements such that it is small enough to assure that the result will have none undesired holes. When a new seed or element is included, the status of the respective region is switched to visited. The synthesis is complete when there are no remaining unvisited regions.

E. Seeds and Neighborhoods

Following similar definition from previous works, seeds are locations that will reference copies of elements from the texture. In our system, before being synthesized, seeds may contain a reference value to a group found during the classification step. When there is such reference, during the synthesis, the best element will be searched only among those contained on the referenced groups. When there is no reference, the best element can be any one from the sample.

In our method, neighborhoods are circular regions. Similarly to pixel-based synthesis methods, the size of the neighborhood can be defined by the user. In this method, this parameter is called the visibility radius of the neighborhood. We believe that using this definition instead of the *n-ring* vicinity around the central vertex on a triangulation is better, because in the second case, neighborhoods could be extremely small or unnecessarily big.

Since the sample input is static, the neighborhood of each element can be previously calculated on the analysis step. However, as the target texture grows, new elements are added and the neighborhood of some elements may change. Because of that, the neighborhood of each element is dynamically updated as the synthesis progress.

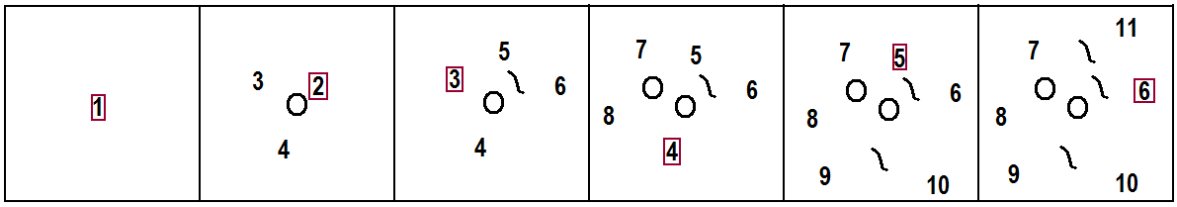

Fig. 3. Illustration of the local texture growth. Each seed is represented by a number, and the highlighted seed is the first on the list.

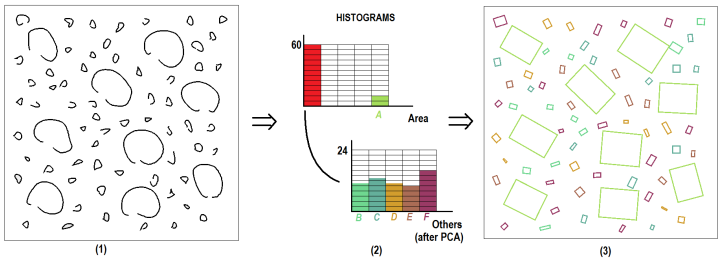

Fig. 4. The classification pipeline: (1) is the initial sample; (2) shows the result of histogram segmentation. The upper histogram is the classification by area. The green column, with 9 elements, becomes a group itself and the red column, with 61 elements, is further refined. The white empty columns are ignored. The lower histogram shows the refinement, were each of the remaining characteristics are reduced to 1 dimension by PCA. (3) is the resulting classified sample. All the parameters, except for the number of bins, are the same as used by Hurtut [3].

F. Synthesis and Seeding

Initially, a seed is placed at a random location inside any of the regions of the target space, and the texture is expanded outwards substituting seeds by copies of elements from the sample and placing new seeds around them. The element to be copied is chosen based on neighborhood comparisons, and the location of new seeds is defined from the relative position of the neighborhood of the copied element. All the seeds are stored in a FIFO list, where the head contains the seed that will be replaced at the next iteration, and new seeds are inserted at the tail of the list.

When a seed is being processed, we search in the sample texture for the best matching element, which represents the element whose neighborhood is the most similar to the vicinity of the seed. For that, we use a metric that returns a value between 0 and 1, representing the dissimilarity between two distinct neighborhoods, with no restrictions on the number of elements on them. The reference for the group contained on the seed, is used to restrict the search into one specific group. Once the best matching is found, we replace the seed with a copy of this element and create new seeds using the relative position of its neighboring. In order to be accepted, seeds must be checked whether will violate any of the minimal distances previously evaluated from the sample. Only if the seed pass all tests, it will be accepted. This process is illustrated in Figure 5.

Using this approach it is possible to control the density of elements on the result. Instead of using exactly the mentioned features, we multiply them by a density parameter. When this parameter is less than 1, closer pairs of elements will be accepted on the final distribution, and therefore, a more crowded texture will be created. When this parameter is greater than 1, a coarser texture will be created. Figure 6 shows the effect obtained by varying this parameter.

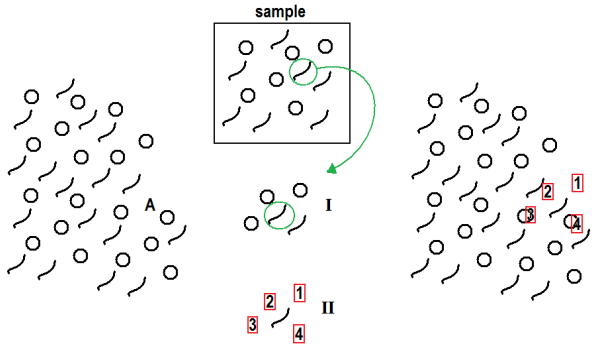

Fig. 5. Seeding process: On the left, we have the current arrangement being synthesized. (A) marks the position of the seed extracted from the head of the list. We search for the best matching element throughout the sample texture. In this case, it must be one element from the group with curves and it was found to be the highlighted curve. We extract the neighboring elements (I) and create four new seeds from them (II). This set is pasted over the seed (A), and on the right arrangement we see that only seed 1 will be accepted.

G. Distance Metric For Neighborhoods

In order to compare the similarity between two distinct neighborhoods, raster texture synthesis methods use the $L2$ norm. We also need to compare neighborhoods, but because this synthesis method deals with distributions on the plane instead of regular grid of pixels, some difficulties arise: the metric should be able to compare two neighborhoods with different number of elements, and the metric should consider the position of each element on the two-dimensional space.

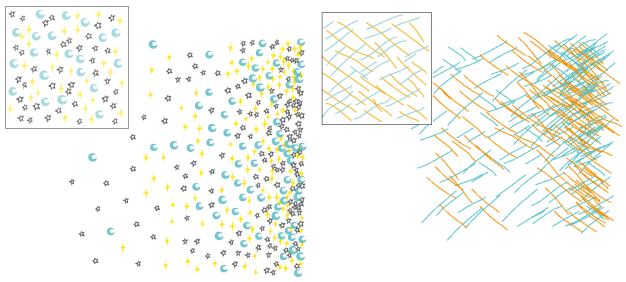

Fig. 6. Varying density along the horizontal axis. Sample and result.

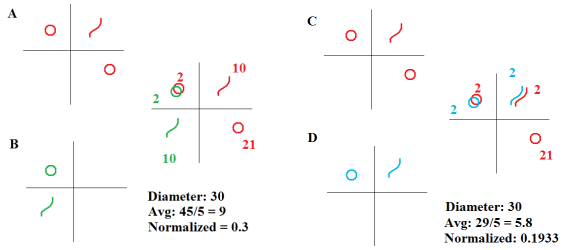

Fig. 7. Distance metric for neighborhoods, on artificial examples. We have the comparison between two pairs of neighborhoods, AB and CD. Neighborhoods A and C are exactly the same, while B and D differ only on the position of one curve. Neighborhoods are illustrated together in the same frame, with the smallest distances evaluated for each element (before normalization). We see that only the displacement of one of the elements, bringing it closer to another element from the other neighborhood, makes both neighborhoods more similar and this effect is easily detected by the final value of the metric. A lower value denotes a more similar pair of neighborhoods.

The metric of this method is normalized to a value between 0 and 1 and gives the dissimilarity between two neighborhoods I and J as follows. For each relative location i of the elements on I, we use the smaller Euclidean distance between i and j, where j is the relative location of the elements on J. When the element at j is from a different group of the element at i, it is ignored. If we reach the end of the list of elements from J and no distance was evaluated, then we consider the element i as too far and set the value to 1. Otherwise, we normalize the found value by the diameter of the visibility window. Once all those values are evaluated for the elements of I, we similarly evaluate the distances for the elements on J. The final overall dissimilarity will be the average of all those values. A value of 1 represents a completely different neighborhood, while a value of 0 represents an exactly equal neighborhood. Our metric is illustrated in Figure 7. Special care is needed for the neighborhoods near the edges that are often incomplete. To avoid the edges, we consider only those elements from the sample whose visibility range does not cross the sample boundaries.

In Barla [5], two neighborhoods are intersected and only the paired elements are considered. If another neighborhood contains the same distribution for the paired elements, but different distribution for unpaired elements or even a different number of them, the metric would return the same value. Ijiri [4] used a similar metric. The main difference from this new metric is that none of the neighbors are discarded and the resulting value is sensitive to small variations on the distribution. We believe that for being more sensitive and because no element is discarded, the metric can capture better the distance value between neighborhoods with distinct number of elements.

H. Choosing The Best Matching

We use the same approach from raster texture synthesis, building an ordered list with the best matching elements and using the corresponding dissimilarity value to order the list. Then, the best element is randomly chosen from those whose dissimilarity factor is smaller than 1.1 times the smallest factor obtained. To avoid repetitions, or growing garbage, for each element we count how many times it was already chosen, and use this value to modify the dissimilarity factor. The new factor is $log(1+n)*d$ where n is the number of times the element was chosen and d is the regular dissimilarity. We use log to avoid the quick growing of the factor.

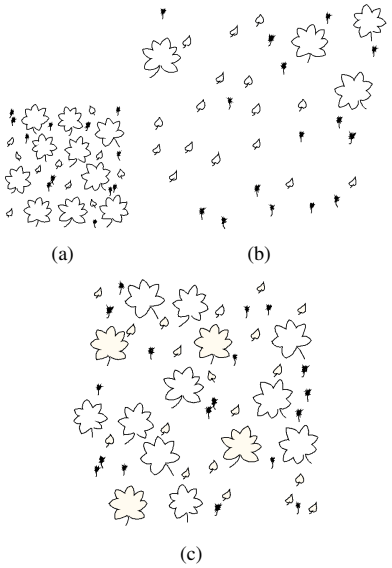

Fig. 8. Given a sample (a) and comparison between results with different values of visibility (b) small and (c) large.

I. Patches

We used the same idea of patches of texture from raster synthesis to improve the efficiency of our method. Instead of copying only one element from the sample, we will try to copy a cluster of elements. We define a parameter – patch size – which will split the visibility region into two parts: inner and outer. The inner part will represent the patch region, where, instead of seeds, the elements will be copied directly onto the target. The same check for violation of features is made before any of those elements are accepted but they do not generate new seeds. In Figure 9 we illustrate the influence of this parameter varying between 0 and 1 times the range of

visibility. This possibility greatly improves the computational cost, and allows similar quality in the visual results.

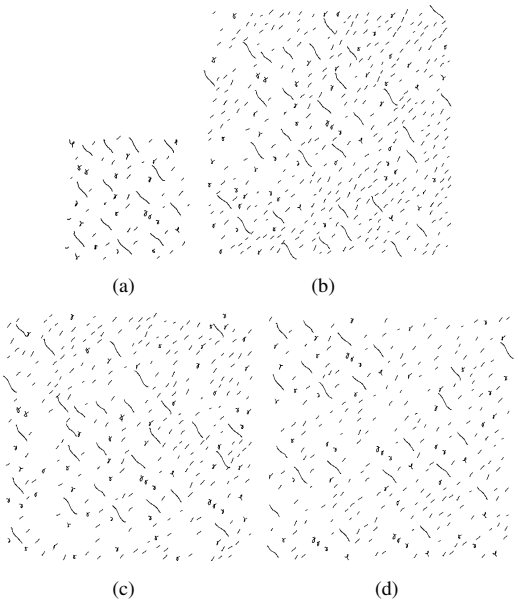

Fig. 9. Given a sample (a) and varying the patch size 0%, 50% and 100%, (b, c, d), respectively. Notice the same quality in the visual result.

IV. Results and Discussion

We have tested our method on a variety of samples and values for the available parameters. Here we illustrate some of these. The number of elements on all samples vary from 10 up to a couple hundred, and in the worst case scenario, the time required strongly depends on the efficiency of the classification. Since the search for the best element is done through all the elements of one group on the sample, if the population of the group is too large, the synthesis can take more than one minute to be completed. However, the average sample presented here could be processed in times varying from real time, up to 10 seconds on a $2.10Ghz$ dual core processor with $2.0GB$ RAM. In Figure 1 we illustrate synthesis results for the four general types of textures: regular, near-regular, irregular and stochastic, accordingly to a rough classification presented in [24]. Our system presented good results for all those types of texture, being near-regular the most difficult to handle. This same figure illustrates a result with the color being one of the discerning characteristics. In the chess texture we also illustrate the capability of the method to synthesize regular patterns. The introduction of color on the second stage of the classification step provided good results for the used colored samples, and did not had influence on the previous black and white textures. In Figure 8 we show the influence of the visibility range parameter on the final result.

Figure 10 shows a comparison between our results and Hurtut's [3], using many of the samples presented in their work. Our model produces good results for irregular, near regular and regular textures as well. We assess the visual quality of both approaches very similar, given the visual characteristics of samples.

In Figure 11, we show results obtained by our method for hand-drawn geological patterns. Notice the continuous variation of the inclination value of the elements on the samples. This presents issues due to the use of histogram, since the inclination value changes in a circular way, where elements with 2π value have the same inclination of 0 valued elements. Also, the lack of strong similarity between elements on pattern (a) implies the necessity of a large number of groups. With our implementation of the method used in previous work, those patterns were not well classified, while with the simple 5 bin histogram method, we had better results. Over-categorization does not represent a big problem, but under-categorization will affect negatively the results, especially in such patterns with compacted long thin elements and no intersections between them.

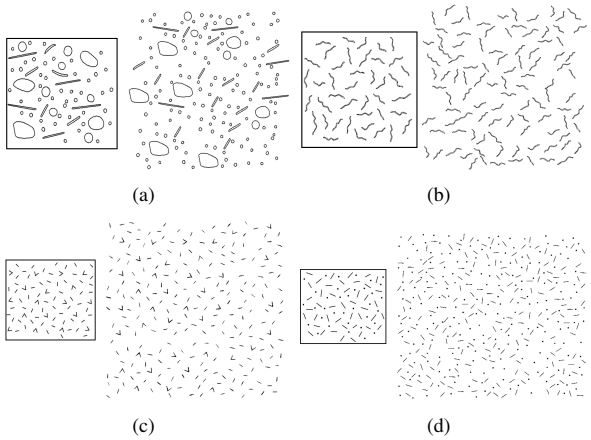

Fig. 11. Texture synthesis results for illustrative geological patterns.

V. Conclusions and Future Work

We have presented a simple example-based method for synthesis of patterns defined as 2D collection of vector elements. Our solution is inspired on previous work, both on patch-based texture synthesis and vectorial arrangement synthesis. This method can produce good results from regular and irregular distribution of elements in the samples, being near regular patterns the most difficult to deal with. Other applications could be explored, as animation synthesis for example. Also, the method could be improved to deal with symmetry of elements.

We plan to extend the model for synthesis over 3D arbitrary surfaces, and improve the direct synthesis of illustrative patterns over 3D models taking into account tonal value maps at different scales. Other areas of future work will include a formal qualitative evaluation of the results by illustrators.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their careful and valuable comments and suggestions. This research

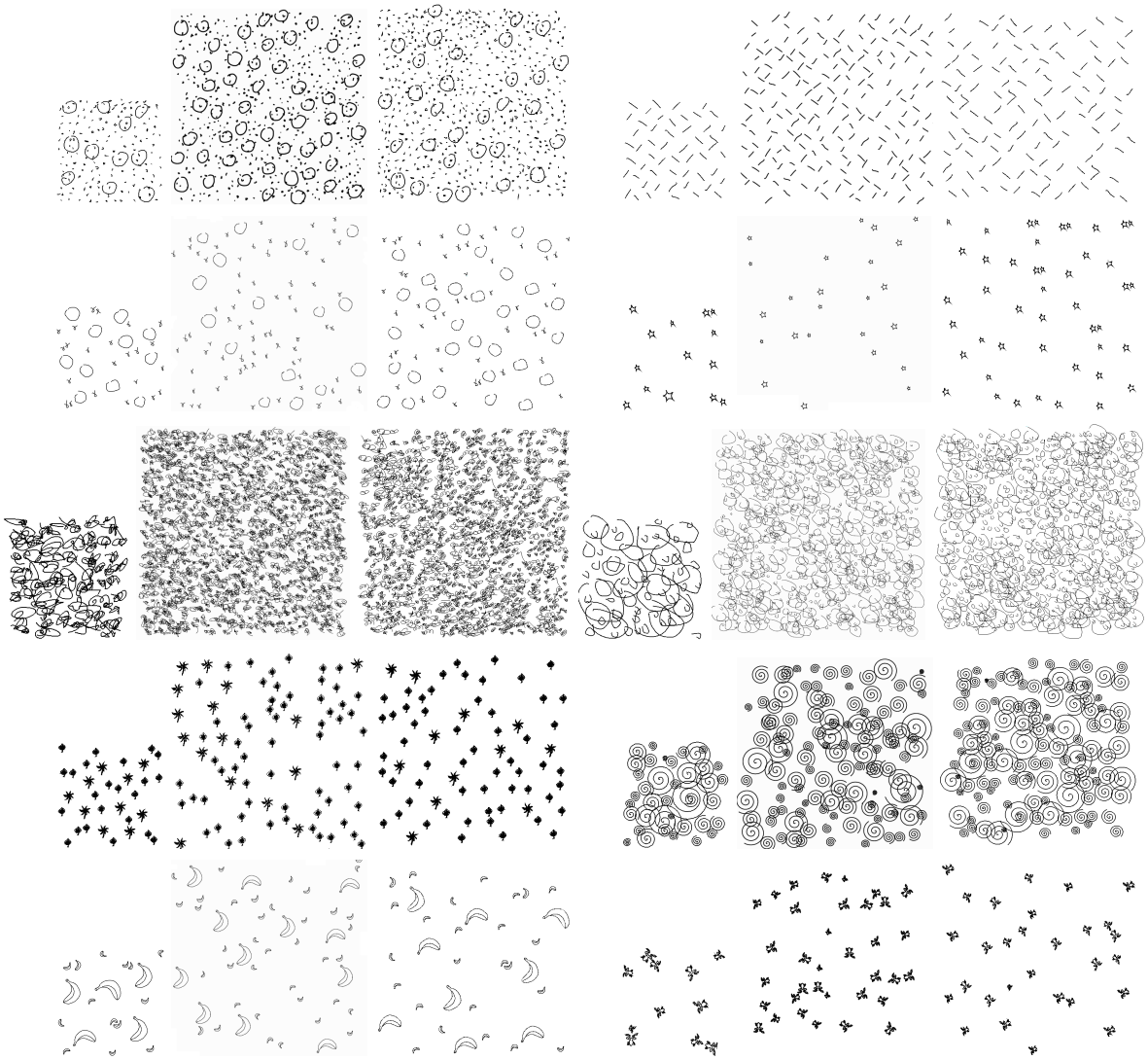

Fig. 10. Texture synthesis results: (from left to right) sample (smaller square), using the method proposed by Hurtut and ours.

was supported in part by grants from the Brazilian funding agency CNPq, by the iCORE/Foundation CMG Industrial Research Chair in Scalable Reservoir Visualization, University of Calgary, Canada.

REFERENCES

[1] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," in *International Conference on Computer Vision*, vol. 2, 1999, pp. 1033–1038.

[2] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," *Proceedings of SIGGRAPH 2000*, pp. 479–488, 2000.

[3] T. Hurtut, P.-E. Landes, J. Thollot, Y. Gousseau, R. Drouilhet, and J.-F. Coeurjolly, "Appearance-guided synthesis of element arrangements by example," in *NPAR 2009: Proceedings of the 7th International Symposium on Non-photorealistic Animation and Rendering*. ACM Press, 2009. [Online]. Available: http://artis.imag.fr/Publications/2009/HLTGDC09

[4] T. Ijiri, R. Mech, T. Igarashi, and G. Miller, "An example-based procedural system for element arrangement," *Computer Graphics Forum*, vol. 27, no. 2, pp. 429–436, Apr. 2008.

[5] P. Barla, S. Breslav, J. Thollot, F. Sillion, and L. Markosian, "Stroke pattern analysis and synthesis," *Computer Graphics Forum*, vol. 25, no. 3, pp. 663–671, Sep. 2006.

[6] M. Ashikhmin, "Synthesizing natural textures," in *The proceedings of 2001 ACM Symposium on Interactive 3D Graphics*, 2001, pp. 217–226.

[7] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Transactions on Graphics*, vol. 20, no. 3, pp. 127–150, July 2001.

[8] A. Efros and W. Freeman, "Image quilting for texture synthesis and transfer," *Proceedings of SIGGRAPH 2001*, pp. 341–346, 2001.

[9] A. Nealen and M. Alexa, "Hybrid texture synthesis," in *Eurographics Symposium on Rendering: 14th Eurographics Workshop on Rendering*, Jun. 2003, pp. 97–105.

[10] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen, "Wang tiles for image and texture generation," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 287–294, 2003.

[11] J. m. Dischler, K. Maritaud, B. Lvy, and D. Ghazanfarpour, "Texture particles," *Computer Graphics Forum*, vol. 21, pp. 401–410, 2002.

[12] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut tex-

tures: Image and video synthesis using graph cuts," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 277–286, 2003.

[13] L. Tonietto, M. Walter, and C. Jung, "A randomized approach for patch-based texture synthesis using wavelets," *Computer Graphics Forum*, vol. 25, no. 4, pp. 675–684, 2006.

[14] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the art in example-based texture synthesis," in *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association, 2009. [Online]. Available: http://www-sop.inria.fr/reves/Basilic/2009/WLKT09

[15] O. Deussen, S. Hiller, C. W. A. M. van Overveld, and T. Strothotte, "Floating points: A method for computing stipple drawings," *Comput. Graph. Forum*, vol. 19, no. 3, 2000.

[16] M. P. Salisbury, S. E. Anderson, R. Barzel, and D. H. Salesin, "Interactive pen-and-ink illustration," in *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1994, pp. 101–108.

[17] G. Winkenbach and D. H. Salesin, "Computer-generated pen-and-ink illustration," in *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1994, pp. 91–100.

[18] V. Ostromoukhov, "Digital facial engraving," in *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 417–424.

[19] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," in *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1998, pp. 453–460.

[20] R. D. Kalnins, L. Markosian, B. J. Meier, M. A. Kowalski, J. C. Lee, P. L. Davidson, M. Webb, J. F. Hughes, and A. Finkelstein, "Wysiwyg npr: drawing strokes directly on 3d models," in *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 2002, pp. 755–762.

[21] A. Hertzmann, N. Oliver, B. Curless, and S. M. Seitz, "Curve analogies," in *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2002, pp. 233–246.

[22] W. T. Freeman, J. B. Tenenbaum, and E. C. Pasztor, "Learning style translation for the lines of a drawing," *ACM Trans. Graph.*, vol. 22, no. 1, pp. 33–46, 2003.

[23] P.-M. Jodoin, E. Epstein, M. Granger-Piché, and V. Ostromoukhov, "Hatching by example: a statistical approach," in *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*. New York, NY, USA: ACM, 2002, pp. 29–36.

[24] W.-C. Lin, J. Hays, C. Wu, V. Kwatra, and Y. Liu, "A comparison study of four texture synthesis algorithms on near-regular textures," in *SIGGRAPH '04: ACM SIGGRAPH 2004 Posters*. New York, NY, USA: ACM, 2004, p. 16.