TEXTURE MIXING VIA UNIVERSAL SIMULATION

By

Gustavo Brown

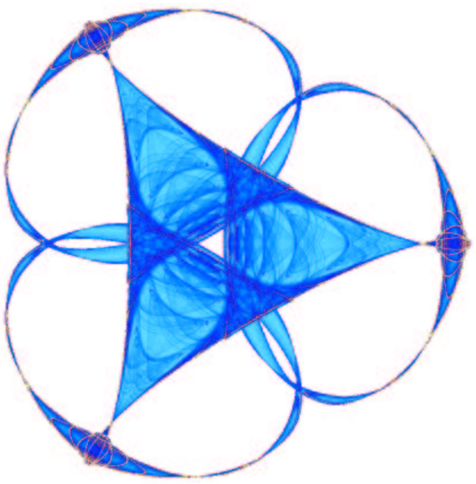
Guillermo Sapiro

and

Gadiel Seroussi

IMA Preprint Series # 2060

(August 2005)



Texture Mixing via Universal Simulation[*]

Gustavo Brown,[†] Guillermo Sapiro,[‡] and Gadiel Seroussi[§]

Abstract

A framework for studying texture in general, and for texture mixing in particular, is presented in this paper. The work follows concepts from universal type classes and universal simulation. Based on the well-known Lempel and Ziv (LZ) universal compression scheme, the universal type class of a one dimensional sequence is defined as the set of possible sequences of the same length which produce the same dictionary (or parsing tree) with the classical LZ incremental parsing algorithm. Universal simulation is realized by sampling uniformly from the universal type class, which can be efficiently implemented. Starting with a source texture image, we use universal simulation to synthesize new textures that have, asymptotically, the same statistics of any order as the source texture, yet have as much uncertainty as possible, in the sense that they are sampled from the broadest pool of possible sequences that comply with the statistical constraint. When considering two or more textures, a parsing tree is constructed for each one, and samples from the trees are randomly interleaved according to pre-defined proportions, thus obtaining a mixed texture. As with single texture synthesis, the k-th order statistics of this mixture, for any k, asymptotically approach the weighted mixture of the k-th order statistics of each individual texture used in the mixing. We present the underlying principles of universal types, universal simulation, and their extensions and application to mixing two or more textures with pre-defined proportions.

1 Introduction and Motivation

Understanding texture is one of the most fundamental problems in image sciences. Questions that have been asked, and still remain largely unanswered, include what is texture [10], how to characterize texture [15, 16], what are the different texture components [4], how to mix textures and create intermediate ones [1], and how to synthesize texture [3] (the latter probably being the one that has been most successfully addressed so far). For significant advances in these directions, see the above mentioned works and references therein.

In this work, we consider a new, information-theoretic, direction, which proves useful in mixing textures so that the statistics of the resulting texture provably satisfies intuitive notions of "mixture," within a framework that can also further our understanding of basic properties of texture. Our proposed study is based on the concept of universal type classes [13] and universal simulation [9, 13]. The universal type class, [13], of a one-dimensional sequence is defined as the set of sequences of the same length that produce the same tree (dictionary) under the Lempel-Ziv (LZ) incremental parsing defined in the well-known LZ78 universal compression scheme [8]. More formally, let x^n be an input sequence of n symbols from an alphabet Λ of cardinality α. The LZ *incremental parsing rule* [8] writes x^n as a concatenation, $x^n = p_0 p_1 p_2 ... p_{c-1} t_x$, of substrings (*phrases*), with p_0 being the null string (λ), p_i, $1 \leq i < c$, the shortest substring of x^n starting one symbol after the end of p_{i-1} and such that $p_i \neq p_j$ for all $j < i$, and t_x (the *tail* of x^n) a (possibly empty) suffix for which the parsing rule was truncated due to the end of x^n. By construction, each non-null phrase is an extension of a previous phrase by one symbol, and the tail t_x must be equal to one of the phrases p_i.

Let $T_{x^n} := \{p_0, p_1, ..., p_{c-1}\}$ denote the *dictionary*, or set of phrases, in the incremental parsing of x^n. The notation hints to the fact that T_{x^n} is best represented by a rooted α-ary tree, where branches are labelled with alphabet symbols, and each node represents a phrase. The root corresponds to $p_0 = \lambda$, and a node (phrase) p_i is connected to its extension $p_i = p_j a$ via a branch labelled with the symbol $a \in \Lambda$. Thus, each phrase is read off the branches of the path from the root to the corresponding node. The *universal type class* of x^n is defined as $U(x^n) := \{y^n \in \Lambda^n : T_{x^n} = T_{y^n}\}$ [13]. It is shown in [13] that if $y^n \in U(x^n)$, then for every integer $k \geq 1$, the k-th order empirical distributions of x^n and y^n converge in the variational sense as $n \to \infty$. On the other hand, any set, S, of sequences whose elements satisfy this property cannot be much larger than $U(x^n)$, in the sense that $\log |S| \leq \log |U(x^n)| + \epsilon$ for any $\epsilon > 0$ and all but a vanishing fraction of sequences x^n (see [13]). Thus, sampling uniformly at random from $U(x^n)$ results in a sequence that has (asymptotically) the same statistics as x^n, and maximum possible entropy (uncertainty) given the sta-

[*]This work is partially supported by the Office of Naval Research, the National Science Foundation, and the National Geospatial-Intelligence Agency. GB performed the work here reported while working at the UofM and at HPL supported by the Office of Naval Research.

[†]Universidad de la Republica-Uruguay

[‡]University of Minnesota

[§]Hewlett-Packard Laboratories

tistical constraint.

We use the above properties of universal type classes to simulate and mix textures. Starting from a source texture image, we represent its universal type class, and sample uniformly from it to obtain textures that approach the same statistics as the source texture, yet they are sampled from the broadest pool of possible sequences that comply with the statistical constraint. An efficient generic procedure for sampling from a universal type class is given in [13]. A modified procedure, adapted for textures, is introduced in this paper. This procedure is the basis of a further modification that leads to texture mixture by combining the parsing trees corresponding to the different source textures (Appendix A). By extension of the theory in [13], it can be shown that the k-th order statistics of this mixture, for all k, approaches the weighted mixture of the k-th order statistics of each individual texture used in the mix.

Beyond its importance in applications such as multidimensional data visualization [7], studying how to mix textures can help us obtain a better understanding of texture in general. Mixing textures has been much less studied than pure texture synthesis. The reasons for this are twofold; on one hand it is a more difficult problem, and on the other hand it is not universally defined. In other words, what do we exactly mean by mixing textures? Intuitively, we want a texture "in between" given samples, though this needs to be formally defined. One of the contributions of this work is that thanks to the results on universal types and simulation, we can formally define the process of mixing textures, as one that, as mentioned above, leads to a new signal which (asymptotically) is statistically identical to the mixture of the individual statistics, for statistics of arbitrary order, and with maximal uncertainty.

We briefly discuss previous art in mixing textures. Leaving aside works on texture blending [2] and mixing of pre-separated texture characteristics [6], probably the most fundamental work is [1]. While their approach is based on mixing by computing the mutual source (the distribution that minimizes the Kullbak-Leibler divergence to the given sources), ours is based on the concepts of universal types and universal simulation and leads as mentioned above to a texture with the prescribed mixture of statistics of the original textures. Although both techniques are based on trees, the one in [1] is obtained from a wavelets decomposition, while ours is derived from the Lempel-Ziv universal parsing (which, see below, can be combined with wavelet decompositions). Our approach does not suffer from some of the problems reported in [1], where special procedures need to be designed to avoid getting locked into one texture. Also, the flexibility of our framework allows for mixing the textures at arbitrary user-provided proportions as well as to mix more than two samples; these features are not reported in [1].

It is interesting to observe that although the LZ compression algorithm is one of the most widely used lossless compression techniques, and its more general combinatorial and statistical properties have been extensively studied in the information theory literature, it has received very little attention from the imaging community beyond compression. Our investigation demonstrates that when properly extended and adapted, these well studied properties can lead to new applications in the imaging sciences. We should also note that the LZ parsing, together with the modifications described in this paper, is just one example of a universal modeling tool that can be brought to bear on problems such as texture synthesis and mixture. Other information-theoretic tools could also be similarly used, e.g., universal context modeling [12]. These modeling tools tend to be very good at capturing the local statistical behavior of images and textures, but are weaker for global structure or "large" patterns. Thus, these tools are best utilized in conjunction with multiresolution techniques from more traditional image processing. Our practical implementations of texture mixture employ such a combination in a very simple form, as briefly discussed in Section 2, and shown in the examples. A full discussion and description are given in the forthcoming full paper; see also Section 3 for open questions in this direction.

The rest of this paper deals with the presentation of the framework and its use for mixing texture. The presentation is mostly conceptual, with some technical details deferred to the appendix. Several examples are presented.

2 Universal Simulation of Mixed Textures

We start from the description of how to simulate (synthesize) a single texture from a single source, and then extend this to the task at hand of mixing multiple sources.

To generate a simulated texture image we first perform an LZ incremental parsing of the input image. Since the original LZ parsing is inherently one-dimensional, we traverse the image following a Hilbert scan [5, 11] (which is optimal based on proximity measures). This, of course, is one of various options available to deal with the two-dimensionality of image data. Other two-dimensional generalizations of LZ studied in the literature could also be employed. Using the dictionary tree resulting from the scanning and the parsing, we draw a new sequence at random from the universal type class (universal simulation), following an adaptation of the algorithm described in [13]. Finally, we convert the simulated (1D) sequence to form the new 2D image. This is done with a modified Hilbert scan that better preserves the texture orientations.[1]

[1]This modified Hilbert process tracks the orientations during the orig-

To adapt the general universal simulation approach introduced in [13] to processing textures, a number of modifications and improvements are introduced (in addition to the modified Hilbert scan already mentioned above). The first modification deals with the way the sampling algorithm works. One of the known problems of the LZ parsing as a modeling tool is the "loss of context" occurring at phrase boundaries. Since phrases are randomly permuted to form the simulated texture, there might be a loss of coherence between two consecutive phrases (which are "snippets" from potentially different places in the original texture). While the effect of this "discontinuity" is statistically negligible (this fact is at the heart of the proof in [13]), it might be visually unpleasant. To make the random sampling output more visually coherent phrases, we extended the LZ incremental parsing algorithm to collect side information while constructing the tree, so that, for each phrase p, a list of "preferred" phrases is kept which have previously occurred preceded by a "context" of a prescribed length m matching the m-suffix of p. These phrase will be preferred as continuations of p, thus preserving context in the phrase transition. In terms of the parsing tree, this modification can be interpreted as causing the sampling algorithm to restart, if possible, at depth m in the tree after outputting a phrase, rather than restarting from the root, as in [13]. When no preferred phrases are available, the conventional restart rule is used. As other modifications described here, this improvement in visual quality of the reproduced texture comes at the expense of a slight reduction in the entropy of the output. Properties of statistical similarity are unaffected.

Another known problem of context modeling tools when applied to practical images (one can regard the LZ parsing as such a tool; see, e.g., [12]) is that of "context dilution." Since the symbol alphabet in continuous tone images is relatively large, there are usually very few exact context repetitions in an image of practical size, except for very short context lengths. In the LZ setting, this means that phrases will tend to be quite short, and will not capture higher order dependencies in the data. To ameliorate this problem, we employ symbol quantization and allow "approximate matches" in the incremental parsing, in order to allow the dictionary to collect longer phrases that better capture image patterns. However, we only quantize the data for the purpose of building the LZ-tree, but keep track of the exact, unique, input string that lead to the creation of the node. When producing the simulated output texture, the original strings are faithfully reproduced, thus preserving the statistics of the texture. As before, this modification, which also improves visual quality, is done at the expense of the output entropy. The quantization threshold is adaptively set in order to achieve a certain mean length for the phrases in the LZ-tree, which can be used as a parameter of the algorithm.

Finally, to avoid possible artifacts due to the Hilbert scan ("false contours"), we used a steerable decomposition [14]. After performing the texture synthesis following the above described algorithm, the high bands obtained from the steerable decomposition of the simulated texture are adjusted using the corresponding permuted high bands of the original texture. The permutation is obtained from the universal simulation. The texture is then reconstructed from these modified high bands and the un-modified lower bands from the simulated texture. The use of these type of decompositions will be further discussed in Section 3.

With a few additional modifications, the above framework for simulating textures can also be used to mix two or more textures. The basic idea is to generate an extended LZ parsing tree for each texture, as described above. To produce a mixed output texture, phrases are randomly sampled from all the trees, drawing the source of the next phrase with probabilities dictated by the pre-specified weight (percentage) of each source texture, as well as the amount of output already drawn from each. Once a tree is selected, the procedure for selecting the next phrase is the same as in the single-texture universal simulation scheme. The combined mixing procedure is described in more detail in Appendix A.

3 Examples and Conclusions

We performed a series of universal simulations for mixing two or three different textures at a time, with a variety of texture weight distributions. For dealing with color images, we first converted them to the YCrCb color space and determined the phrase permutation utilizing only the information in the luminance channel, keeping track of the full color information for each sample. Figures 1 and 2 show the examples, see also http://www.ece.umn.edu/users/gusbro/iccv.html.

In this work we reported results on the use of universal simulation ideas to study textures. This is based on concepts of universal types, derived from the well known Lempel-Ziv parsing algorithm. The goal is not just to synthesize mixed textures, but to understand what texture is. We are currently investigating the projection of textures onto the universal class, or in other words, how much of a given texture is obtained from the universal simulation (following recent trends on separating different components of an image, e.g., [4, 10]). While the result of a universal simulation may show some visual artifacts that distinguish it from the original texture, a second iteration of the universal simulation on the result of the first will be visually indistinguishable from its input. This, the universal simulation proposed here indeed behaves like a type of projection that might lead to a better understanding of "visual randomness" in images. In

inal texture scanning process and then uses them, permuted following the universal simulation, during the reconstruction of the synthesized texture.

a sense, the universal simulation procedure does a faithful job within the constraints of the (implicit or explicit) statistical model used. The output of a universal simulation shows us how the chosen model "sees" the texture, and provides guidance to improving and refining the model. In a closely related direction, and as frequently done for texture analysis/synthesis and mixing, multiscale/multi-orientation decomposition should be incorporated more fully into our approach. This can be done either performing the universal simulation in each band separately or working with vectors, incorporating all or part of the bands as the dictionary Λ. Results in these directions will be reported elsewhere.

A Mixture Random Sampling from Multiple Universal Type Classes

We describe, in pseudo-code form, the sampling procedure that produces a mixture of given input textures. We assume that N texture samples are given, and each has been Hilbert-scanned and parsed with the LZ incremental parsing algorithm, modified as described in Section 2 to avoid the "context dilution" problem. Thus, trees $T_1, T_2, \ldots, T_N$, each corresponding to a respective texture, have been produced, and that each node $\mathbf{t}$ in each tree represents a substring $s(\mathbf{t})$ from the corresponding input sample (we assume that the substring is stored in the node, since, due to the quantization mentioned in Section 2, we may not be able to reconstruct it solely from the path leading to the node). Furthermore, we assume that positive real numbers $R_1, R_2, \ldots, R_N$ are given, such that $\sum_i R_i = 1$, and R_i is the desired proportion of texture i in the output mixture. We also assume that a desired length n of the output sequence is specified, and that the size of the i-th input sample is at least $n_i = \lceil nR_i \rceil$. For simplicity, the description omits the enhancement introduced to combat the "loss of context" problem, also discussed in Section 2. This visual enhancement is rather independent of other properties of the scheme, and when the restart depth parameter m is fixed, it does not affect the main statistical properties of the sampling algorithm. We also omit the reverse Hilbert scan procedure previously described.

Each node $\mathbf{t}$ of the trees is marked as `used` or `unused`, each stores an auxiliary variable $V(\mathbf{t})$ counting the number of currently `unused` nodes in the subtree rooted at $\mathbf{t}$, and the depth of $\mathbf{t}$ in the tree is denoted $|\mathbf{t}|$. For each tree T_i, we keep a count C_i of the number of symbols output that originated in phrases from T_i. We also keep an overall output symbol count $L = \sum_i C_i$.

```
1. Initialization. For each $i$, $1 \leq i \leq N$, set $C_i = 0$. For each node $\mathbf{t}$ of each tree $T_i$, mark $\mathbf{t}$ as unused, and set $V(\mathbf{t}) = c_{\mathbf{t}}$, the size of the subtree rooted at $\mathbf{t}$. Set $L = 0$.
2. Draw an integer $i$ with probability $P(i{=}j) = (nR_j{-}C_j)/(n{-}L)$, $1{\leq}j{\leq}N$, and set $\mathbf{t}{=}\lambda_i$, the root of $T_i$.
3. If $\mathbf{t}$ is unused:
   (a) If $C_i + |\mathbf{t}| > nR_i$, pick a node $\mathbf{v}$ of depth $\lceil nR_i \rceil - C_i$ uniformly in $T_i$, and set $\mathbf{t} = \mathbf{v}$.
   (b) Output $s(\mathbf{t})$, mark $\mathbf{t}$ as used, set $V(\mathbf{t}) = V(\mathbf{t}) - 1$, $C_i = C_i + |\mathbf{t}|$, and $L = L + |\mathbf{t}|$.
   (c) If $L \geq n$, stop. Else, go to Step 2.
4. Draw a symbol $a{\in}\Lambda$ with probability $P(a{=}b) = V(\mathbf{t}b)/V(\mathbf{t})$, $b \in \Lambda$. Set $V(\mathbf{t}) = V(\mathbf{t}){-}1$, $\mathbf{t} = \mathbf{t}a$, and go to Step 3 (note: we get to this step only with $V(\mathbf{t}){>}0$).
```

References

[1] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Texture mixing and texture movie synthesis using statistical learning," *IEEE Transactions on Visualization and Computer Graphics* **7(2)**, pp. 120–135, 2001.

[2] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Transactions on Graphics* **2**, pp. 217-236, 1983.

[3] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," *IEEE International Conference on Computer Vision, Corfu, Greece*, pp. 1033–1038, Sept. 1999.

[4] J. M. Francos, A. Z. Meiri, and B. Porat, "A unified texture model based on a 2-D Wold like decomposition," *IEEE Trans. Signal Process.* **41**, pp. 2665-2678, August 1993.

[5] C. Gotsman and M. Lindenbaum, "On the metric properties of discrete space-filling curves," *IEEE Transactions on Image Processing* **5**, pp. 794-797, 1996.

[6] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," *ACM SIGGRAPH*, pp. 229-238, 1995.

[7] V. Interrante, "Harnessing natural textures for multivariate visualization," *IEEE Computer Graphics and Applications*, November/December 2000.

[8] A. Lempel and J. Ziv, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory* **24**, pp 530-536, September 1978.

[9] N. Merhav and M. J. Weinberger, "On universal simulation of information sources using training data," *IEEE Trans. Inform. Theory* **50**, pp. 5-20, January 2004.

[10] Y. Meyer, *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations*, AMS University Lecture Series **22**, 2002.

[11] A. Perez, S. Kamata, and E. Kawaguchi, "Peano scanning of arbitrary size images," *Proceedings of IEEE International Conference of Pattern Recognition*, pp. 565-568, 1992.

[12] J. Rissanen, "A universal data compression system," *IEEE Trans. Inform. Theory* **IT-29**, pp. 656-664, Sept. 1983.

[13] G. Seroussi, "On universal types and simulation of individual sequences," *Theoretical Informatics: 6th Latin American Symposium,* Lecture Notes in Computer Science, **2976**, M. Farach-Colton (ed.), Springer-Verlag, Berlin, 2004, pp. 312–321.

[14] E. Simoncelli, W. Freeman, E. Adelson, and D. Heeger, "Shiftable multiscale transforms," *IEEE Transactions on Information Theory* **38**, pp.587-605, 1992.

[15] E. Simoncelli and J. Portilla, "Texture characterization via joint statistics of wavelet coefficient magnitudes," *Proc. 5th IEEE Int'l Conf. on Image Processing*, 1998.

[16] S. C. Zhu, Y. N. Wu, and D. Mumford, "FRAME: Filters, random field and maximum entropy: Towards a unified theory for texture modeling," *International Journal of Computer Vision* **27(2)**, pp.1-20, March/April 1998.

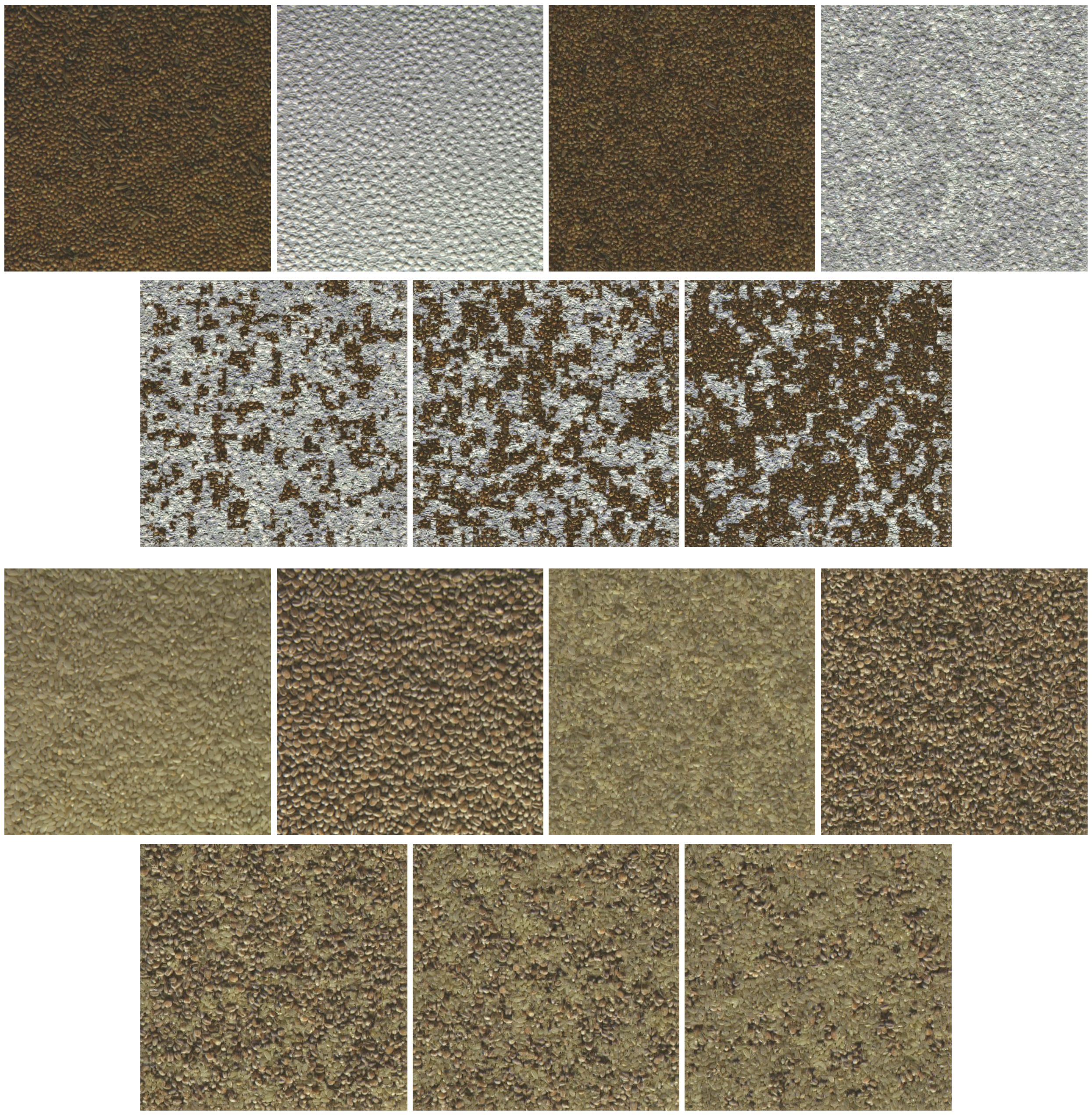


Figure 1: *Examples of texture mixture. The first row shows two original textures followed by their universal simulation results (which correspond to percentiles of 100-0 and 0-100 in the mixture). The second row show results of the mixture for percentiles of 30-70, 50-50, and 70-30 respectively. This is repeated for a second set of textures in the last two rows. (This is a color figure.)*

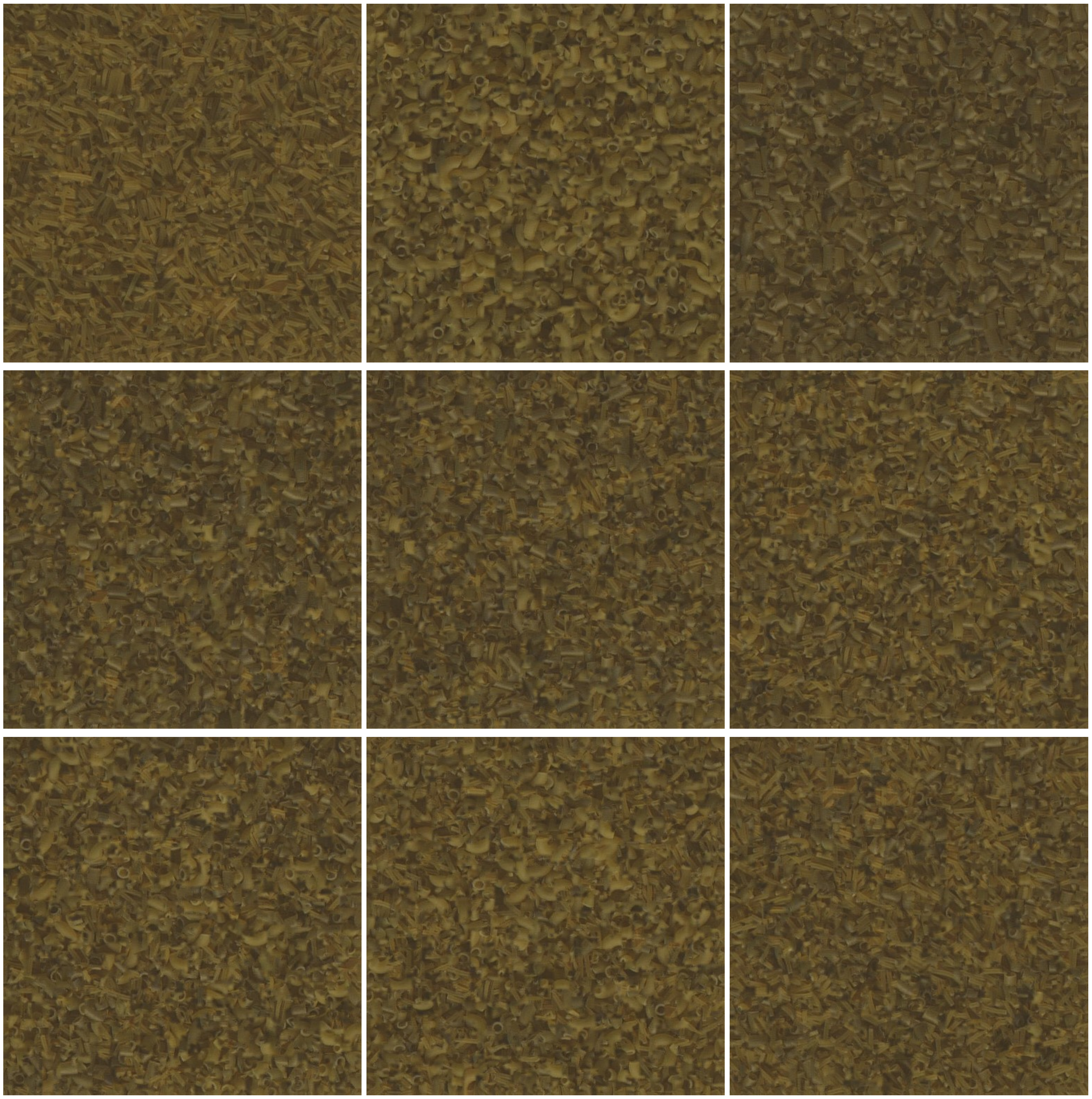


Figure 2: *Mixing three textures. The respective percentiles, from let to right and top to bottom, are 100-0-0, 0-100-0, 0-0-100, 10-30-60, 25-25-50, 33-33-33, 25-50-25, 30-60-10, and 60-10-30, respectively. (This is a color figure.)*