Informational Texture Synthesis

Sylvain Lefebvre, Li-Yi Wei, Connelly Barnes

Informational Texture Synthesis

Sylvain Lefebvre, Li-Yi Wei, Connelly Barnes

February 13, 2018

Abstract

This paper presents a novel technique to embed messages into textures synthesized from tiles with boundary constraints. The messages are encoded in the choices that the synthesis algorithm normally makes at random during the synthesis process. Images embedding messages are not distinguishable from images that are not embedding any message. The random sequence seed used during synthesis can act as a password.

1 Change log

This is an ongoing project research, the draft will be updated often as we progress.

- 2018-02-13 Updated the draft with some previous work.
- 2018-02-08 Initial document.

2 Introduction

This document is a work in progress and to keep it short and to the point we skip introductory matters and focus instead of the main principle.

The synthesis algorithm is based on the *wave function collapse* algorithm `https://github.com/mxgmn/WaveFunctionCollapse` which is itself an improvement over model synthesis [5] (see also references from the web page). We strongly encourage the interested reader to first get familiar with this algorithm.

3 Previous work

[1] proposes to encode data in synthesized textures by constraining the color of pixels spread with a specific pattern. Synthesis is then performed to fill-in gaps around these points. [2] encodes both the message and the exemplar into the final image. The exemplar is stored in pre-positioned square patches which locations are produced from a secret key. Synthesis is performed around these using image quilting [4]. Whenever a patch can be added, multiple candidates

are considered. The candidates are ranked by how well they fit neighboring tiles in the overlap region, and the rank of the selected candidate is used to encode data. [3] also relies on image quilting for synthesis around pre-positioned square patches (tiles) which locations are produced by a secret key. The choice of which tile appears at each location encodes data. The authors note that approaches using tiles are more robust to e.g. JPEG compression than those using pixels.

These approaches focus on synthesizing unstructured, stochastic patterns. Our technique synthesizes a different type of texture patterns which are described by tiles with assembly constraints [5]. Instead of pre-determining the location of a set of tiles, we hide the data into the (pseudo-)random sequence of choices encountered during synthesis.

4 Synthesis

In the remainder it is to be understood that 'random' means 'pseudo-random using a sequential, deterministic generator'. Therefore, any random sequence produced during the algorithm can be exactly reproduced using the same start seed.

The core principle is to layout square tiles to form a large texture. Each tile can be seen as a label (the tile id). Tiles cannot be freely placed: constraints define how tiles can be neighboring (e.g. tiles A and B may only be placed with A to the left of B). This lets artists paint pattern pieces onto the tiles which, once assembled while enforcing all constraints form a consistent pattern e.g. a network of tubes.

Our variant of the algorithm goes as follows:

1. Initialize an array S of size $W \times H$, where each cell contains a vector of T booleans, with T the number of tiles. All vectors are initialized to *true*. Each boolean represents the possibility that a particular tile appears at a particular location. Initially everything is possible.

2. Visit each location of S in some *fixed* order. For instance, one can use scanline ordering (left, right then top, bottom).

3. At each location, consider the vector of possibilities. If a choice exists (e.g. more than one tile is possible), we have an opportunity to embed a bit of information. A naive approach would select the first choice to represent '0' and any other choice to represent '1'. Once the choice is made, set all other entries in the vector to false.

4. Propagate the constraints to kill all choices in neighboring cells. This step is unchanged compared to the original algorithm.

5. Repeat until all cells of S have been visited. After this, a choice has been made everywhere and we can produce the final image by assembling the tiles.

This process can encounter a contradiction at step 3: in such cases no possibilities exist in the visited cell, due the propagation of constraints from previously visited tiles. If this happens, the algorithm is launched again with a different random sequence.

5 Decoding

The decoding algorithm starts by reconstructing the labels from the input image. This can be done by simple image comparison if the image is unaltered, or with some more elaborate vision based algorithm if the input comes from a photograph.

The decoder runs synthesis again. However, when a choice is possible, it uses the input to determine which choice was made during the initial synthesis. This decodes one bit of information each time (still assuming the same naive approach). At the end the bit stream is fully recovered.

If at some point the decoder finds a discrepancy (a set of choices that does not contain the tile selected in the input), then it can determine that synthesis failed and was re-run. It generates the proper state for the random generator and restarts.

6 Encoding information in choices

The initial proposal of encoding '0' in the first choice and '1' in any other is not a good strategy. Indeed, an empty message is represented by a zero bit stream: selecting consistently the first choice will strongly bias the synthesis process and could quickly run in contradictions. Instead, we map '0' to half the choices and '1' to the other half. Which half corresponds to '0' or '1' in randomly determined at each cell. This approach is still limited to 1 bit per choice. Of course, when multiple choices are available the encoder can exploit this as additional storage space, in the manner of a data compressor (e.g. arithmetic coding).

7 Implementation

A proof of concept is available at `http://members.loria.fr/Sylvain.Lefebvre/infotexsyn/`. Code will be released when ready, but it should be simple to implement after reading this document.

8 Discussion

Tilesets This algorithm works with any tileset that provides choices (e.g. a tileset with a single tile provides none). Some tilesets will allow to encode larger messages than others in a same space. Whether and why a tileset allows for more information space in an intriguing theoretical question that we intend

to explore. This can be measured experimentally by counting the number of choices offered by the tileset over a large number of synthesis.

Messages Another intriguing question is whether pathological messages exist, that would consistently lead to a contradiction regardless of the initial random seed. Such message would be impossible to encode.

Image size As it is difficult to ensure a message will fit, and since the synthesis is fast, we devise the following approach. First, the algorithm determines the size in bits of the message and uses previously computed average capabilities of the tileset to estimate the size of S (assuming some aspect ratio, e.g. 1:1). If the process terminates before all bits are encoded, S is enlarged and the process resumes. If on the contrary a lot of space is left empty, S could be reduced for a tighter fit. Note that messages are always padded with zeros until synthesis completes.

Steganography While this is a form of steganography, it is unclear yet whether our approach is a good or bad steganography approach. The extent of the novelty is also an unknown quantity at this time.

References

[1] Hirofumi Otori and Shigeru Kuriyama. Data-embeddable texture synthesis. In *International Symposium on Smart Graphics*, pages 146–157, 2007.

[2] Kuo-Chen Wu and Chung-Ming Wang. Steganography using reversible texture synthesis. *IEEE Transactions on Image Processing*, 24(1):130–139, 2015.

[3] Zhenxing Qian, Hang Zhou, Weiming Zhang, and Xinpeng Zhang. Robust steganography using texture synthesis. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, pages 25–33. 2017.

[4] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of SIGGRAPH*, pages 341–346. ACM, 2001.

[5] Paul Merrell. Example-based model synthesis. In *I3D*, pages 105–112. ACM, 2007.