Compat-Map For Real-time Texture Synthesis and Rendering

Linling Zhang, Simon Fenney
Imagination Technologies

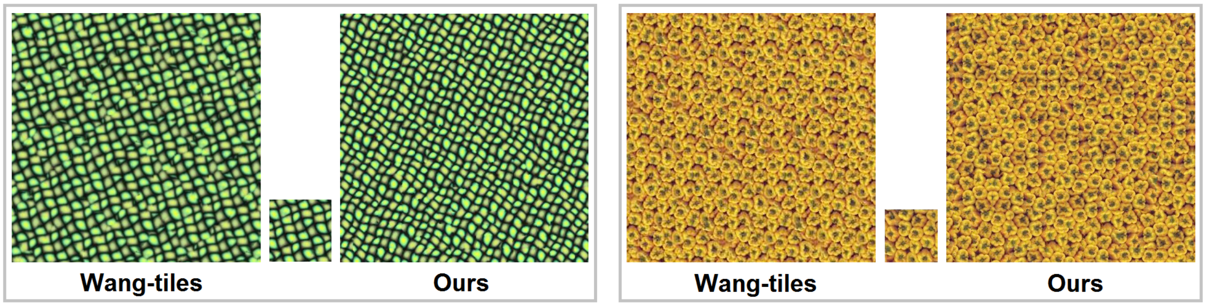

Figure 1: *Our Mini-map based texture synthesis achieves real-time performance similar to Wang-tiles. Our method avoids diamond artifects (left) or missing important patterns (right, the yellow pepper in shadow), which is comparatively common with Wang-tiles method.*

1 Introduction

We propose a Compat-Map based real-time texture synthesis method which is for influencing Graphics Processing Unit (GPU) Design. With low complexity of calculation and low request of bandwidth and storage, our texture synthesis is a practical algorithm which achieves both high quality results and real-time performance. Representing the synthesized texture as an index table, ***Concurrent Random Access*** is another essential advantage, which significantly improves the performance of graphic chips.

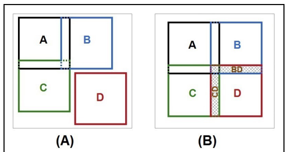

Figure 2: *Basic matching scheme for Mini-map.*

41	47	238	156	175	139	159	158
169	222	176	92	139	251	156	146
159	24	183	175	239	202	175	153
140	13	190	139	168	207	139	98
233	208	159	243	175	160	80	192
224	216	157	250	139	168	156	200
221	223	140	99	0	147	146	207
162	75	163	134	7	154	188	159

Figure 3: *An Index-table of a synthesized texture.*

2 Patch matching with Mini-map

our method is composed of three procedures: the Compat-Map construction followed by the index map generation and patch tiling.

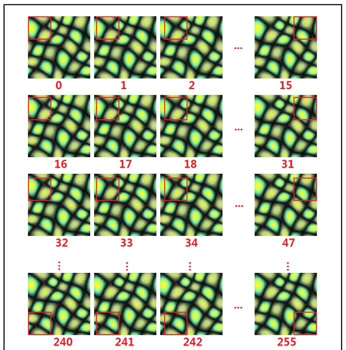

Figure 4: *Fixed set of blocks (squares with red edge) from a small sample image.*

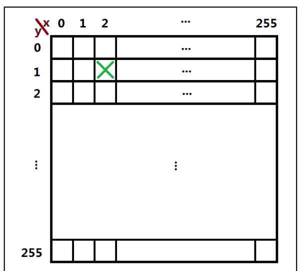

Figure 5: *Mini-map structure. The element located at the green cross should be the ID of the best matched block D if the ID of block B is 2 and the ID of block C is 1.*

Our Compat-Map is developed using the best-matched block for each arbitrary *two* image blocks by measuring the overlap (the shadows in Figure 2(B)) in the placement of random blocks from the sample image. It should be noticed that block A as in Figure 2(B) has little impact on the selection of block D. Thus, to find a best matched block D, we just need to evaluate colour difference for shadowed BD and CD. Only notations of block B, C, D need to be preserved for the matching of patch ABC and D, if the sample blocks belong to a fixed block set. To produce the fixed block set, the simplest way is uniformly picking up blocks from the sample image. Figure 4 shows an example with 256 sample blocks.

Thus, we propose a Compat-Map as in Figure 5. It is a two dimensional array, using the number of sample blocks (256 for example) as the fixed length for both dimensions. The x dimension indicates IDs of sample blocks placed at B and y dimension indicates IDs of sample blocks place at C. The IDs of best matched blocks for each B and C combination are filled into the Compat-Map.

With a prepared Compat-Map, we fill in the top and leftmost lines with random block IDs(the shaded elements in Figure 3). All empty cells left in the Index-table can be easily filled in a raster order, by reference to the prepared Mini-map. For example, the element marked by a red ellipse should refer to Mini-map[176][24]=183.

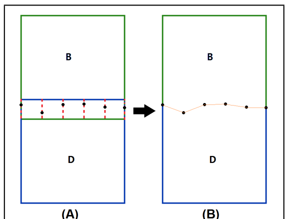

Figure 6: *Sample based minimum cutting.*

Replacing the IDs in the Index-table with corresponding sample image blocks, we can get the synthesized texture. But we can see obvious block effects caused by the regular shape of sample blocks and their edge colour difference. To eliminate the block effect, we propose to use a sample based minimum cutting, the result of which is almost as good as the minimum cutting.

3 Results and Conclusion

Our Compat-Map based texture synthesis generates high-quality textures (Figure 1) and achieves real-time performance (Table 1) even on low-cost/mobile devices.

Synthesied Size	Analysis Time	Synthesis Time
280 × 280	0.82s	0.002s
460 × 460	0.82s	0.007s
910 × 910	0.82s	0.03s

Table 1: *Performance tested on a mobile tablet with a core of A7@1.2Ghz, using* 256×256 *Mini-map, from* 128×128 *sample image.*

The overall performance can be further improved with GPU. Different from existing methods, permitting concurrency random access by use of Index-table is a huge bonus for graphic chip design.