Antipole Clustering For Fast Texture Synthesis

Battiato S., Pulvirenti A., Reforgiato Recupero D.
Dipartimento di Matematica ed Informatica - Università di Catania
Viale A. Doria 6 - 95125- Catania Italy
{battiato, apulvirenti, diegoref}@dmi.unict.it

ABSTRACT

This paper describes a new method for analysis/synthesis of textures using a non-parametric multi-resolution approach able to reproduce efficiently the generative stochastic process of a wide class of real texture images. This is realized through a new data structure the Antipole Tree and a suitable research strategy able to outperform both the classical linear full-search heuristic and the TSVQ (Tree Structure Vector Quantization) acceleration used in previous related works. Experimental results performed on an exhaustive set of textures [VisTex] show the effectiveness of the proposed approach.

Keywords

Texture Synthesis, Pyramid Decomposition, Antipole Clustering, Nearest-neighbor.

1. INTRODUCTION

The exciting world of "texture", with its different application and results (texture classification, discrimination, retrieval, mapping and/or rendering) represent only a partial view of the various lines of research and application fields. Among others, to be able to realize fast and effective algorithm for texture synthesis, with high performance both in term of real time generation and perceived quality is a fascinating goal. Two different strategies or lines of research have been followed in the literature. The more ambitious one tries to "learn", with a proper set of filters, the underlying stochastic model ([Cro83]) of an input texture; the synthesis is then obtained by a suitable sampling. The main drawback of these methodologies is the computational complexity that tends to be impractical for real-time applications [Wu00]. More efficient techniques tend to properly match texture features ([Por00]), measured at different resolution levels ([Bur83]): a series of heuristics are used without explicitly derive a real mathematical model. In [Heg95] and [Deb97] impressive results using marginal histograms of image pyramids and maintaining cross-scale dependencies are obtained (see also [Bat00a], [Bat01b]).

WSCG POSTERS proceedings
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.
Copyright UNION Agency – Science Press

More recently [Efr99] and [Wey00] pointed out a series of simple but effective techniques showing excellent results on large class of textures. In particular the work presented in [Wey00] has been furtherly generalized in [Her01] to realize a computational framework where analogies between pairs of images can be deduced. Other techniques such as those presented in [Xu01b] combine together smart patch merging. This paper describes a series of possible solutions trying to improve existing algorithmic solutions by making use of advanced approximated search data structures. The procedural approach described in [Wey00] applies a multiresolution technique tracking neighborhood dependence level by level. The synthesis is realized using a classical sampling strategy over the data collected in the analysis phase. The entire process is then accelerated using a TSVQ (Tree Structure Vector Quantization) [Ger92]. We claim that the overall computation time needed to perform a full-search sampling strategy can be avoided using suitable advanced data structures and searching strategies. In our approach, image pixels are grouped into clusters of bounded radius by an efficient clustering algorithm: the Antipole Tree Clustering [Can02b], which belongs to the class of the "bisector trees" [Cha01]. The clustering probability model of spatial neighborhoods derived from a texture was introduced for the first time by [Pop93]. Our clustering algorithm works in such a way that "far" elements lie in different clusters. The algorithm is able to find such a pair (A, B) (called Antipole) in linear time. Finally, elements of the sets are partitioned according to their proximity to one of the two Antipole endpoints (A, B). This top-down recursive splitting procedure will produce a binary tree whose leaves are the final clusters. Once the Antipole Tree data structure is built an efficient nearest neighbor search algorithm is developed and used in our framework. Experimental results show the improvement with respect to the sampling process described in [Wey00] in term of computation time with respect to the full-search strategy while maintaining the same final quality. The paper is structured as follows. The next Section introduces the Antipole clustering strategy and describes an

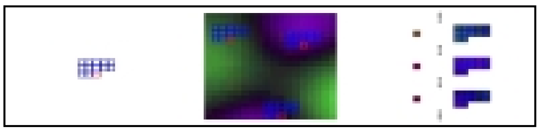


Figure 1 - Typical shape of the neighbourhood used in the analysis/synthesis process (on the left). Each synthesized pixel, as showed in the magnified texture (in the middle), is determined by suitable analyzing its neighbourhood.

algorithm for the nearest neighbor problem comparing performances of different techniques. Section 3 shows the experiments results obtained with the proposed solution while a conclusions Section tracks direction for future works and research.

2. ANTIPOLE STRATEGY AND TSVQ

The Wei-Levoy algorithm [Wey00] uses the locality and stationarity properties of the textures to synthesize an image by a raster scan order. Each pixel in the input sample is mapped into the pixel of the nearest neighborhood vector (see Fig. 1). The input consists of an example texture patch together with a random noise image having the desired size of the output image. The algorithm modifies this random noise to make it looks like the given example. Since this process is computationally expensive, multi-resolution pyramids and quantization acceleration are used (see [Ash01], [Bil01], [Wey02]). In particular a speed-up is obtained using TSVQ [Ger92], which takes as input a set of training vectors and generates a binary tree of codebooks having a depth specified by the user, which will be representative of the dataset. First the process finds a centroid *c* of the training vector and uses it as root of the tree. After that, the same centroid *c* and a properly perturbated centroid are chosen as children of the root. The process proceeds recursively until the specified depth is reached. In [Wey02] the approximation introduced considers a variable number of training codebooks allowing also a limited backtracking in the tree traversal to trade-off between computation time and final image quality. In our approach a positive cluster radius σ, is used to guarantee that pixels with a similar neighborhood lie in the same cluster. The Antipole clustering of bounded radius [Can02b] is performed by a top-down procedure starting from a given finite set of points *S* which checks if a given splitting condition is satisfied. If this is not the case then splitting is not performed and the given subset is a cluster. The computation of an approximate centroid [Can02] having an approximated distance less than σ from every point in the cluster is hence performed. Otherwise, a suitable pair of points (A, B) of *S* called Antipole is generated and the set is partitioned by assigning each point of the splitting subset to the closest endpoint of the Antipole (A, B).

```
NEAREST-NEIGHOBOR(Tree $T$, Object $Q$, Threshold $t$,
                  OutputObj $OUT$)

if ($T.Leaf = FALSE$) then
    $D_A \leftarrow$ CHECK($Q$, $T.A_l$, $t$, $OUT$);
    $D_B \leftarrow$ CHECK($Q$, $T.A_r$, $t$, $OUT$);
    $Q.D_v \leftarrow Q.D_v \cup \{D_A, D_B\}$
    if($D_A \leq D_B$) then
        NN-VISIT($T.left$, $T.right$, $Q$, $t$, $OUT$,
                 $T.Rad_l$, $T.Rad_r$, $D_A$, $D_B$);
    else
        NN-VISIT($T.right$, $T.left$, $Q$, $t$, $OUT$,
                 $T.Rad_r$, $T.Rad_l$, $D_B$, $D_A$);
    end if;
else
    VISITCLUSTER($T.Cluster$, $Q$, $t$, $OUT$);
end if;
END NEAREST-NEIGHBOR.
```

```
NN-VISIT(Tree $A$, Tree $B$, Object $Q$, Threshold $t$,
         OutputObj $OUT$, Radius $Rad_A$,
         Radius $Rad_B$, Distance $D_A$, Distance $D_B$)

 if ($D_A < t + Rad_A$) then
    NEAREST-NEIGHOBOR($A$, $Q$ $t$, $OUT$);
 end if;
 if ($D_B < t + Rad_B$) then
    NEAREST-NEIGHOBOR($B$, $Q$ $t$, $OUT$);
    end if;
end if;
END NN-VISIT.
```

```
VISITCLUSTER(Cluster $Cluster$, Object $Q$,
             Threshold $t$)

$Q.D_v \leftarrow Q.D_v \cup \{$ CHECK($Q$, $Cluster.C$ $t$, $OUT$) $\}$;
for each $O \in Cluster.C_{List}$ do
    if TRIANGULARITY($O$, $Q$, $OUT$, $t$) $= FALSE$ then
        CHECK($Q$, $O$, $t$, $OUT$);
    end if;
end for each;
END VISITCLUSTER.
```

```
CHECK(Object $Q$, Object $O$, Threshold $t$, OutputObj $OUT$)

$D_O \leftarrow dist(Q, O)$;
if ($D_O < t$) then
    $t \leftarrow D_O$;
    OUT $= O$;
end if;
return $D_O$;
END CHECK.
```

Figure 2 – Pseudo code of the NEAREST-NEIGHBOR, VISITCLUSTER **and** CHECK **procedure.**

```
APPROXIMATE-NEAREST-NEIGHBOR(Tree $T$, Object $Q$)

if ($T.Leaf = FALSE$) then
    $D_A \leftarrow$ DIST($Q$, $T.A_l$,);
    $D_B \leftarrow$ DIST($Q$, $T.A_r$,);
    if($D_A \leq D_B$) then
        APPROXIMATE-NEAREST-NEIGHBOR($T.left$, $Q$);
    else
        APPROXIMATE-NEAREST-NEIGHBOR($T.right$ $Q$);
    end if;
else
        return $T.Cluster.Centroid$;
end if;
END APPROXIMATE-NEAREST-NEIGHBOR.
```

Figure 3 – APPROXIMATE NEAREST NEIGHBOR **search via Antipole Tree.**

The Antipole algorithm has been successfully applied in a Mobile Wireless Network Problem [Fer02]. Once the data structure is built a suitable nearest neighbor algorithm can be designed around it. Fig. 2 contains the pseudo-code relative to the nearest neighbor search performed on the corresponding Antipole Tree built by the Antipole Clustering strategy. The search, starting from the root, proceeds by following the path in the tree, which guarantees that the nearest cluster centroid is found pruning the impossible branches. A backtracking search explores the remaining branches of the tree to obtain a correct answer. At each step the distance between the query object and the nearest neighbor is bounded by a threshold *t*. At the beginning of the process *t* is initialized to +∞. During the search, an overall speed-up is obtained by using the triangle inequality. In order to do that, during the Antipole data structure construction, for each introduced object *O* a vector O_{DV} of all distances from *O* to each Antipole element in the unique path followed by *O* from the root to its final cluster is maintained. A similar Q_{DV} is generated also for each query *Q*. The procedure TRIANGULARITY will check if exists an element $1 \leq i \leq min(|Q_{Dv}|, |O_{DV}|)$ such that $t < Q_{DV}[i] - O_{DV}[i]$. If the condition is verified then the object *O* will be discarded. On the other hand if such a condition is not verified we need to compute the distance between the object *O* and the query *Q* in order to decide if the element can be in the output set. Notice that the Antipole indexing organize the data in such a way that linear scanning during the search may be avoided. This results in a faster nearest neighbor search procedure with respect to the linear Nearest Neighbor search. As suggested in [Wey02] a suitable acceleration of the Antipole Tree Search described above can be obtained by introducing an approximated nearest neighbor search (see Fig. 3). This modified approximating procedure follows the path in the tree, which minimizes the distance from the query, returning the centroid of the cluster contained in the leaf node. This approximated nearest neighbor search is fully competitive with the TSVQ acceleration producing in many cases better results.

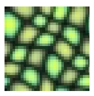
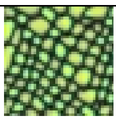
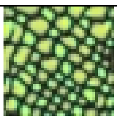
		
(a)	(b)	(c)

Figure 4 - Synthesized images obtained by classical full search (b) and Antipole Tree Search (c) from input images (a) using {5x5,1} neighborhood.

3. EXPERIMENTAL RESULTS

This Section reports all the experimental results obtained with the proposed approach with respect to the work of [Wey00]. Figure 4 shows a synthesized texture obtained respectively with the linear full-search strategy and by Antipole strategy. As reported in the previous Section the Antipole strategy speeds-up the process, without data loss. The notation $\{R_1 x C_1, 1\}, \ldots \{R_i x C_i, k_j\} \ldots, \{R_m x C_m, k_n\}$ indicates multiresolution *n* levels each with neighbor size $R_i x C_i$ at the top level merged with the previous k_j-1 levels each one having neighborhood size R_i -2 x C_i-2,…., R_i -2*(k_j-1) x C_i-2*(k_j-1). For example the expression {7x7,1}{9x9,2} means: synthesise 2 levels multiresolution with the first level neighbour size 7x7 and the second level neighbour 9x9 merged with the previous level with neighbour size 7x7. The algorithm was implemented in ANSI C and all experiments were carried out on a PC PIII 900Mhz on Linux OS, using the [VisTex] database. Each running time as reported in [Wey00] is referred to the synthesis process obtained starting from a random equalized noise image. The timing comparison has been realized, using different neighborhood, single level and multiresolution. In all cases the running time of the Antipole strategy was better than the classical full search.

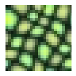
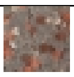
Images	Neighboor	Full	Antipole	Gain (%)
 Texture	{5x5,1}	728	44	93,96
	{7x7,1}	1386	69	95,02
	{5x5,1}{5x5,2}	1260	84	93,33
	{7x7,1}{7x7,2}	2580	206	92,02
Flowers	{3x3,1}	221	38	82,81
	{9x9,1}	2820	427	84,86
	{5x5,1}{5x5,2}	1260	205	83,73
	{5x5,1}{7x7,2}	2520	428	83,02
 Money	{5x5,1}	728	179	75,41
	{9x9,1}	2820	1145	59,40
	{3x3,1}{5x5,2}	1221	374	69,37

Table 1 - Running time comparison, in seconds, between full search (third column) and Antipole data structure (fourth column). The second column describes the size of each neighbor at each level. Percentage gain is reported on the last column

Table 1, reports some results, showing the average time and the corresponding percentage gain obtained over several textures. The results presented in Table 1 show that the Antipole Clustering running time over distinct textures may be different. This means that the underlying vector space distribution generated during the synthesis process also affects the performance of the proposed method. The running time of the Antipole Tree reported in Table 1 includes the building time of the tree which takes only a few seconds. The TSVQ acceleration used by ([Wey00], [Wey02]), is able to run two orders of magnitude faster than the full search. It works well, over a large dataset of texture, but introduces some approximation. As shown in [Ash01], the textures that are composed of various small objects do not give output images of good quality (e.g. leaves, flowers, etc.). In many cases only the full neighborhood search guarantees satisfactory results. Fig. 5 shows a series of examples with comparisons with the TSVQ acceleration. The TSVQ tree is constructed using the maximum number of codewords as suggested in [Wey00][1]. Our proposed acceleration technique seems to be more robust, as shown in Fig. 5. The acceleration allows a further time gain ranging from 70% to 90% with respect to the exact Antipole search (see: http://alpha.dmi.unict.it/~texture/ for more details).

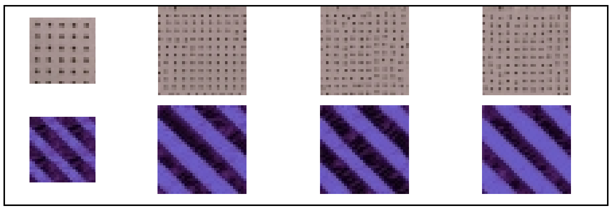


Figure 5 - A comparison between the Antipole exact search (second column), approximate Antipole search (third column), TSVQ search (last column).

4. CONCLUSIONS

In this paper we proposed a novel approach for texture synthesis using the Antipole Tree Data structure. Future work includes the study of new approximation strategies and indexing methods together with the introduction of new heuristics and sampling strategies to solve some shortcomings of the current method ([Ash01], [Cha01], [Her01]).

ACKNOLEDGMENTS

A special thanks goes to Prof. A. Ferro and Prof. G. Gallo for useful discussions and suggestions.

5. REFERENCES

[Ash01] M. Ashikhmin, *Synthesizing natural textures,* ACM Symposium on Interactive 3D Graphics, pp. 217–226, 2001;

[Bat00a] S. Battiato, G. Gallo, *Multi-resolution Clustering of Texture Images* – Texture Analysis in Machine Vision, - Vol. 40, pp.41-51, World Scientific, 2000;

[Bat01b] S. Battiato, G. Gallo, S. Nicotra – *Glyph Representation of Directional Texture Properties* – J. of WSCG, Vol. 10, No.1-3, pp. 48-54, 2002;

[Bil01] P. Billault, *Texture Synthesis Algorithms*, Image Signal Depart. de Math. Appliquees, Technical report 2001 ;

[Bro66] P. Brodatz, *Textures: A Photographic Album for Artists & Designers*, Dover, New York, 1966;

[Bur83] P.J. Burt, E.H. Adelson, *The Laplacian Pyramid as a Compact Image Code*, IEEE Trans. on Comm., Vol. 31, pp.532-540, 1983;

[Can02] D. Cantone, G. Cincotti, A. Ferro, A. Pulvirenti, *An Efficient Approximate Algorithm for The 1-Median Problem in Metric Spaces*, Tech. Rep. Univ. of Catania, 2002;

[Can02b] D. Cantone, A. Ferro, T. Maugeri, A. Pulvirenti, D. Shasha, *Antipole Indexing to support Range Search on Dynamic Metric Space*, Tech. Rep. Univ. of Catania 2002;

[Cha01] E. Chávez, G. Navarro, R. A. Baeza-Yates, J. L. Marroquín: *Searching in metric spaces*. ACM Comp. Surveys 33(3): pp. 273-321 (2001);

[Cro83] G.R. Cross, A.K. Jain, *Markov Random Field texture models*, IEEE, *PAMI*, Vol. 5, pp. 25-39, 1983;

[Deb97] J.S. De Bonet, *Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images*, In Computer Graphics, pp. 361-368, ACM SIGGRAPH, 1997;

[Efr99] A. Efros, T. Leung, *Texture Synthesis by a Non-parametric Sampling*, IEEE Proc. of ICCV, pp.1033-38, 1999;

[Fer02] A. Ferro, G. Pigola, A. Pulvirenti, D. Shasha, *Fast Clustering and Minimum Weight Matching Algorithms for Very Large Mobile Backbone Wireless Networks*, Int. J. of Found. of Comp. Science - Special Issue on Wireless Networks, 2002 to appear;

[Ger92] A. Gersho, R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Ac. Publ., 1992.

[Heg95] D.J. Heeger, J.R. Bergen, *Pyramid-Based Texture Analysis/Synthesis*, In Computer Graphics, pp. 229-238, ACM SIGGRAPH, 1995;

[Her01] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, D.H. Salesin, *Image Analogies*, In Proc. of ACM-SIGGRAPH, pp 327-340, 2001;

[Pop93] K. Popat, R. Picard, *Novel cluster-based probability model for texture synthesis, classification, and compression*, In Visual Comm. and Image Processing, pp. 756-768, 1993;

[Por00] J. Portilla, E.P. Simoncelli, *A Parametric Texture Model based on Joint Statistics of Complex Wavelet Coefficients*, Int. J. of Comp. Vision, Vol. 40, No. 1,. 2000;

[VisTex] Texture Synthesis: VisTex Texture, http://graphics. stanford.edu/projects/texture/demo/synthesis_VisTex_192.html;

[Wey00] L.Y.Wei, M. Levoy, *Fast texture synthesis using tree-structured vector quantization*, In Proc. of ACM-SIGGRAPH 2000, pp 479–488, 2000;

[Wey02] L.Y.Wei, *Texture synthesis by Fixed Neighborhood Searching*, Ph.D.Diss.Stanf. Un. 2002;

[Wu00] Y.N. Wu, S.C. Zhu, X.W. Liu, *Equivalence of Julesz Ensemble and FRAME Models*, Int'l Journ. of Comp. Vision, Vol. 38, No. 30 pp.245–261, 2000;

[Xu01b] Y. Xu, S.C. Zhu ,B. Guo, H.Y. Shum, *Asymptotically Admissible Texture Synthesis*, In Proc. of Int'l Workshop. on Sta.t and Comp. Theories of Vision, 2001;

[1] We are grateful to University of Washington Data Compression Laboratory (http://rcs.ee.washington.edu/COMPRESSION/code/tsvq/) to make available a freeware source code of the TSVQ.