International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:4, No:4, 2010

Recursive Similarity Hashing of Fractal Geometry

Timothee G. Leleu

***Abstract*—**A new technique of topological multi-scale analysis is introduced. By performing a clustering recursively to build a hierarchy, and analyzing the co-scale and intra-scale similarities, an Iterated Function System can be extracted from any data set. The study of fractals shows that this method is efficient to extract self-similarities, and can find elegant solutions the inverse problem of building fractals. The theoretical aspects and practical implementations are discussed, together with examples of analyses of simple fractals.

***Keywords*—**hierarchical clustering, multi-scale analysis, Similarity hashing.

I. Introduction

A. Multi-scale analysis

MULTISCALE analysis is the mathematical modeling of phenomena from the point of view of scales. The growing interest in non-linear equations has created a need to understand how systems react to scaling. Multiplying all coefficients by a ratio in a linear equation does not change the over whole behaviour; whereas scaling a non-linear equation may lead to drastic changes: as scale increases, dynamical systems may be subject to chaotic behaviours on complex attractors.

In order to simplify the representation of these systems, models have incorporated conditions of their stability (*Hyperbolic systems* with stable and unstable manifolds[1]), or conditions of plural stability (*bifurcation theory*[2]) , their geometric properties in their phase space [3], and finally, of their statistical multi-scale behaviour[4]. Current efforts aim toward the interpretation of geometric properties at multi-scales (notably *multi-fractals*[5]).

Multiscale also refers to systems which are the result of the interaction of a large number of agents, as in *multi-physics* phenomena. To model a physical problem of higher complexity, at each scale is applied a corresponding physical model. This technique has been used extensively in weather forecast[6]. Multi-scale analysis appears to be a transversal approach to numerous scientific domains, from complex networks, with *small-world networks*[7] and *fractal machines*[8], to the extent of the unification theory, with the *invariant set postulate*[9].

In this paper, a new technique of geometrical multiscale similarities will be introduced, performing a recursive similarity hashing, based on the hierarchical clustering of data set. This method can solve the inverse problem of building fractals, and be applied more generally to any data, in order to extract co-scale similarities (redundant patterns in a given scale), and trans-scale dependencies (the variation of properties according to the scale); which provides a theoretical framework to study a vast horizon of phenomena.

Timothee G. Leleu is with The University of Tokyo (Institute of Industrial Science, University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo 153-8505 Japan, email: timothee@sat.t.u-tokyo.ac.jp)

B. Related work

Wavelet analysis, by finding the repetition of intensity in the plan, and taking into account the scale factor, have a practical application in image compression [11]. For the study of multi-fractals[12], wavelets can find a spectrum of the self-similar intensities; not taking into account the geometrical features of sub elements.

In terms of similarity analysis applied to fractal data set, similarity hashing [13] explores the frequency of all possible transformations (as translations, rotations between all pairs of elements); which provides a spectrum with a majority of redundant or hardly representative mappings coding self-similarities. All other attempts of solving the inverse problem of building fractals, ranging from the “moments method”[14], wavelet analysis[15], to genetic algorithm[16], do not provide an analysis depending on the scale; they however provide prototypes of models to understand fractals.

More recently[17], a hierarchical decomposition of data set achieved by a succession of dilatations and contractions have yielded interesting results, showing that the self-similar parts of a fractal can be identified, or clustered. The optimal method to achieve this clustering is one of the topics of this paper. Clustering techniques, as SVM-RFE[18], density clustering[19] do not provide the right answer for a recursion analysis of geometrical self-similarities. A clustering based on simple rules, and its heuristic implementation (to some extent similarly to the k-means[20] algorithm) run recursively on self-similar data, can be used to rebuild a hierarchical structure and compute a simple representation of its self-similarities.

In next section will be defined the Recursive Similarity Hashing technique, and introduced the Context-Dependant IFS, from which the conditions on the clustering in its implementation will be based. In section 4 are described the results of the analysis of fractal data.

II. Recursive Similarity Hashing

A. Motivations and basic concepts

In the field of databases, hashing is the act of extracting, usually from a large amount of data, a smaller representation, of fixed sized, in order to organize this data. The data "hashed" is then classified according to a discriminating function, which will use a "key", the *hash key*, to find the proper tables to distribute the data. Entries which are "closer" to one another are grouped in common tables. For example, a database storing clients according to their address, could use as a hash function the postal code, and as a discriminating function to put in the same table the clients with identical postal code.

Similarity hashing (definition which differs from [13]) is based on a similar idea. Applied to a target data set, the hashing is applied to all possible combinations of groups of elements, and the "key" can be used to select the combinations of groups that have a potentiality to present similarities.

Formerly, the data studied is represented by a vector space associated to a metric (X^n, d) (it could be for example the phase space of dynamical system, in $\Re^n$ with the Euclidian distance, $n \in \aleph$ the dimension). The hash function h operates on a combination of data points, and outputs a key, which could be a scalar, or more generally a vector:

$$h : (X^n)^{n_e} \to H \subset Y^m \qquad (1)$$

Usually, the space Y is $\Re$, $m < n$, and n_e is the number of elements on which a key is defined. The term "hash function", not just morphism, is used to emphasize the role of this function: to classify the data into groups. The grouping depends on the degree of similarity of two groups $\tilde{x}_1$ and $\tilde{x}_2$, where $\tilde{x} \in (X^n)^{n_e}$ is a group of cardinal $card(\tilde{x}) = n_e$. The keys are then used to calculate a discriminating value, by applying a discriminating function D, comparing all the possible combinations of n_g keys, where n_g is the number of groups (of clusters) into which the data will be partitioned. If the combination has a potential to be a good one (under some criteria which will be detailed further), that is to say its partition into groups would contain similarities, then the discriminating value is higher:

$$D : \vec{h} \in H^{n_g} \mapsto D(\vec{h}) \in \Re \qquad (2)$$

It is important to note here that the discriminating function does not compare two groups, but the keys of a combination of groups which would be the best decomposition of the data into similar sets. The reason is subtle: if a kind a similarity function could be calculated, comparing two groups and returning a similarity indicator (a correlation), this would mean that there was *a priori* some knowledge about their similarities, in order to define this function. The discriminating function is of another kind: its definition is based on assumptions about the topology (in the space of the hash key) where similarities can be found, not about the similarities themselves. The key should (as in classical hashing) extract the relevant information to prepare the groups to be classified. The discriminating function should select the set of keys that have the best potentials.

Before moving to the real application, a practical example will illustrate the concept of similarity hashing. A librarian is asked to classify cooking books. The wrong method would be for this librarian to compare pair by pair all the books, and decide on their resemblance: all contain very different recipes, and even if some elements may be similar (two versions of an identical recipe for example), two books are hardly similar. The right method is to first decide on a hash function. For instance, select only the nationality of the author. Then the discriminating function would be to put on the same shelf books with identical culinary origins. Now that the books which are candidate to contain features of similarity have been selected, the librarian can find real similarities: fusion cuisine is similar to Californian and Japanese cuisine etc.

B. Hierarchical clustering

Before detailing the form of the hash and discriminating function, what criterion should be used to quantify the potential of containing features of similarities (the nationality of the author from the example above) ought to be determined.

The archetype of a self-similar data is a fractal. In order to remain in the most general case, nothing about the aspect of the self-similarities will be implied; rather the topological aspects of where they could be found will be discussed. A fractal has the property to contain smaller parts (subparts) identical to larger ones. In this sense, a fractal is a multi-scale invariant (whereas a multifractal is a multi-scale variant). Therefore, selecting from a scale of a fractal will provide the elements of its similarities.

In order to select from a scale, the fractal (which is a set a vector, with no organization in scale) must be decomposed into a hierarchy, with each level representing a scale.

Supposing that there was a method to select the groups with best potential to carry similarity features. Recursively grouping these groups will generate a hierarchical clustering of the set studied.

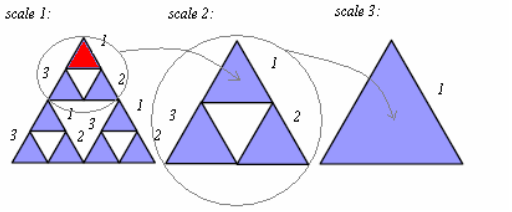

Fig. 1: hierarchical grouping of the Sierpinski gasket

In the figure 1, from the 9 smaller triangles composing a Sierpinski gasket, 3 parent groups are formed, which are similar, then a parent group again similar. This decomposition creates a classification of the points into a tree, called *hash tree*. The red triangle has the coordinate $(1,1,1)$ in the tree of figure

2. The coordinate on the tree, at a scale l of a certain data point of index i is noted $c(l,i)$.

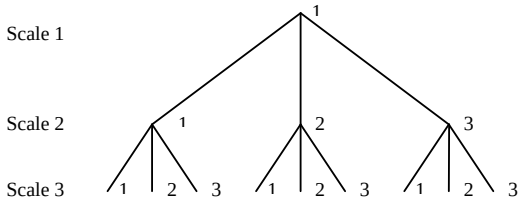

Fig. 2: tree representing the hierarchical clustering of the Sierpinski gasket

C. Conditions on the clustering, Hash and discriminative function

To identify the groups with best potential of carrying similarity features, two properties of fractals are used: (1) *Each parent group is composed of neighboring groups; (2) Groups are not overlapping, i.e. there are as far as possible from one another.*

By applying these conditions on the grouping, or clustering method, the potentiality of these clusters to contain features of self-similarity is optimized.

The hash function of this Recursive Similarity Hashing is to compute the center of masses $x_g \in X^n$ of all groups $\tilde{x} \in (X^n)^{n_e}$:

$$h(\tilde{x}) = x_g = (1/n_e).\sum_{x \subset \tilde{x}} x \quad (3)$$

The discriminative function is the calculation of the Root Mean Square (RMS) of the distances between centers for a configuration of grouping; all the data points are grouped by n_e tuplets, for all possible combinations of tuplets of size n_e each:

$$D(\vec{h}) = RMS_{h(\tilde{x}_i),h(\tilde{x}_j)\in H^{n_g},i\neq j}(d(x_{g_i},x_{g_j})) \quad (4)$$

The term $\vec{h} = \left\{h(\tilde{x}_i) = x_{g_i}\right\}_{i\in\{1,..,n_g\}} \in H^{n_g}$ is one of the possible combinations of n_g keys.

$$RMS_{h(\tilde{x}_i),h(\tilde{x}_j)\in H^{n_g},i\neq j}(d(x_{g_i},x_{g_j})) = \sqrt{\frac{\sum_{h(\tilde{x}_i),h(\tilde{x}_j)\in H^{n_g},i\neq j} d(x_{g_i},x_{g_j})^2}{n_g}}. \quad (5)$$

It is the RMS of the distances between the centers of masses for all possible combinations of groups.

D. Center of Masses Optimization

To accomplish the type of clustering described in the previous section, a technique called *Center of Mass Optimization* (CMO) is used. It consists in selecting n_g groups $\tilde{\tilde{x}} = \left\{\tilde{x}_i \in (X^n)^{n_e}\right\}_{i\in\{1,..,n_g\}}$ for which the centers of masses $\vec{h}(\tilde{\tilde{x}}) \in H^{n_g}$ of all the group $\tilde{x} \in (X^n)^{n_e}$ of n_e elements are the furthest from one another:

$$\mathrm{K} = \arg\max_{\tilde{\tilde{x}}}(D(\vec{h}(\tilde{\tilde{x}}))), \quad (6)$$

The term $\arg\max$ is the argument of the maximum, i.e., the groups for which the discriminative function is maximal. The condition "is maximal" in the definition of K means that the n_g groups respecting this condition are selected (and thus $\mathrm{K} \in (X^n)^{n_g}$).

In real applications, it is very unlikely that two combinations obtain an identical value from the discriminating function (due to noise in the observation at least). Therefore, the choice of the argument of the maximum is always unique in practical cases. The choice of n_e and n_g will be discussed in the next section.

E. Recursive clustering

The process of selecting groups via CMO clustering is then recursively applied on the children groups found, to generate the next generation of groups. Step by step, an organization in hierarchy is constructed.

The relation between parent groups (represented by the center of masses of the cluster) and children can be modeled by a weighted (directed from parent to child) graph $G = (E,V)$, where the edges E are the positions of the center of masses (or vectors themselves in the cases on singletons); the vertices V represent the relation parent-child. The weight function $w: V \rightarrow w(V)$ returns in simple cases the distance between the parent and children nodes:

$$w(V(E_1,E_2)) = d(E_1,E_2) \quad (7)$$

In more general cases, the weight function can return a *cost function*: in complex networks, the cost can be the band switch between the nodes (servers) etc. It is in fact a correlation

indicator between the parent and child node.

The graph built has the shape of a Cayley Tree: starting from a point (the ancestor node), children nodes spread across the space, at each generation as far as possible from one another.

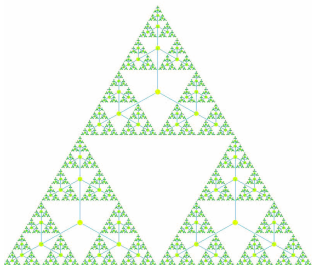

Fig. 3: Cayley tree representing the hierarchy of the Sierpinski Gasket (the nodes are yellow dots, the vertices green lines)

In the figure 3, all n elements can be equally distributed between groups at each scale (or generation), each containing n_e elements. In this particular case, due to the topology of the Sierpinski gasket, the number of elements is $n = 3^{n_l}$, where n_l is the number of generations (in figure 3, $n_l = 5$); the choice $n_e = 3$ distributes all elements. In other cases, the data must be truncated in order to be formatted into a power of n_e. The case of not equally distributed groups will not be discussed here in further details.

F. Context-Dependant IFS

The decomposition in a hierarchy of fractals, achieved by Recursive Similarity Hashing, is the reverse process of building fractals in a hierarchy. A Context-Dependant Iterated Function System[22] is an operator building, from a starting point, a fractal organized in a hierarchy, or equivalently, an associated Cayley tree. This novel method differs from traditional Iterated Function Systems used to build fractals[10] by taking in consideration the context of the construction of a fractal set.

Formerly, the definition of a Context-Dependant IFS Ψ is the following:

$$\Psi = \left\{ f'_{i,j} \right\}_{i=\{1,\ldots,N\}, j \subset M^k = \{1,\ldots,N\}^k, k \in K \subset \aleph}, \quad (8)$$

with N being the number of initial mappings $f'_{i,j}$. This operator is applied to a starting point $x_0 \in X^n$ to build a sequence of growing cardinal defined by the relation:

$$x_{l+1}^{\,i \cup j} = x_l^{\,j} + f'_{i,j}(x_l^{\,j}), l \in \aleph \quad (9)$$

The index i represents the index of the mapping used to construct the vector $x_{l+1}^{\,i \cup j}$, j is the set composed of the indexes of the k previous transformations iteratively composed to form the vector $x_l^{\,j}$. The sequence built is thus indexed by their context in the recursion. The index k represents the memory of the system. In the case of $K = \aleph$, the IFS keeps memory of all the context, and consequently Ψ is composed of an infinity of mappings. The upper apostrophe is used to differentiate the mappings of the Context-Dependant IFS, to the ones of the usual IFS.

The *fractalian* is the union of all vectors built after l iterations, and the *leaves* is the union of these from a certain generation $l = n_l$. The fractalian converges towards the fractal when n_l tends to infinity, under certain conditions of convergence (the mappings must have a *decreasing progression*, see [22] for more details). A Cayley tree is associated to the Context-Dependant IFS: the vectors are the edges, and the vertices link vectors which are related by their context. Vectors of generations l and $l+1$ are in a Parent-Child relation if they share the same context j.

The interesting properties of such constructions are plural: (1) the associated Cayley tree, or fractalian, can preserve the organization in a hierarchy of the fractal under conditions which make the fractalian "invertible"; and thus the fractalians can be inverted in order to compute rigorously the Context-Dependant IFS associated; (2) the error with which the *fractalian* built describes a fractal structure is detailed precisely.

A Context-Dependant IFS building a Sierpinski gasket (shown in figure 4) is the following (with a complex notation representing the space $\Re^2$):

$$\begin{aligned} \overline{f'_1(n)} &= (1/2)^n . e^{i\pi/2} \\ \overline{f'_2(n)} &= (1/2)^n . e^{-i\pi/6} \\ \overline{f'_2(n)} &= (1/2)^n . e^{-i\pi(5/6)} \end{aligned} \quad (10)$$

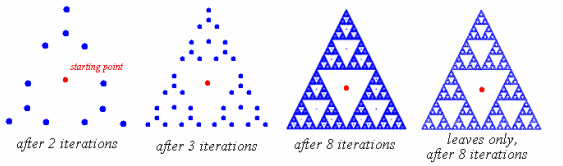

Fig. 4: fractalian of the Sierpinski gasket built with the Context-Dependent IFS of relation (13)

The conditions used for the clustering of the Recursive Similarity Hashing are in fact derived from the conditions which make fractalians "invertible". The latter conditions are: (i) the children vectors are closer and closer to their parent as their generation increases; (ii) children from common parent are closer to each other than to children from another parent. These conditions are exactly equivalent to the conditions (1) and (2) described previously.

The conditions on the clustering of the Recursive Similarity Hashing aim to extract a Context-Dependant IFS from the target data set. In this sense, Context-Dependant IFS and Similarity Hashing are complementary; the former being the construction process, and the latter the formalization. If the target data respects (1) and (2) rigorously an associated Context-Dependant IFS can be extracted with ease. In more difficult cases, the best compromise between these two conditions will provide the best hierarchy from which similarities can be recognized, in order to decide of the most representative Context-Dependant IFS.

G. Pyramidal tensor

Once the organization in a hierarchy is extracted (which is itself an interesting indicator of similarity), proper similarities can be identified. From this point, the forms of the similarities have to be restricted according to the problem. In order to find a simplifying representation of the target data, based on the analysis of its self-similarities, the next step is to analyze which transformations (mappings) relate a parent node to a child node, in two situations: for all pairs of parent to child mappings from an identical scale (or generation for the associated Context-Dependant IFS), the *co-scale* similarities; for all pairs of parent to child across all generations, the *trans-scale* similarities.

Fractals (mono-fractals) should exhibit *co-scale* invariance and *trans-scale* contractions, while multi-fractals should be more generally *trans-scale* uniformly variant. By analyzing both types of similarities, the Recursive Similarity Hashing is truly a multiscale analysis.

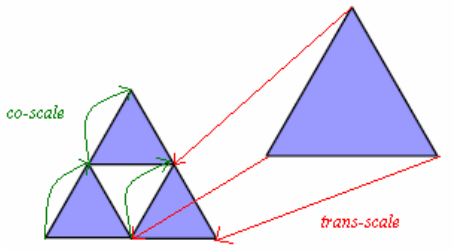

Fig. 5: co-scale and trans-scale similarities of the hierachized Sierpinski Gasket fracalian

The formalization of the Context-Dependant IFS focuses on mappings $f'_{i,j}$ which are the shifts from the position of a parent to the position of its child. From the relation (12), the position of a vector of generation n_l can be expressed as the sum of all the shifts impacted by the previous generations:

$$x_{n_l}^{\,j(n_l)} = \sum_{l=0}^{n_l-1} \underbrace{f'_{i(l),j(l)}(x_l^{\,j(l)})}_{shift} \qquad (11)$$

In order to study, from the organization in a hierarchy found by the recursive clustering, the shifts impacted at each generation, a *pyramidal* tensor P of size n_l is defined as:

$$\mathrm{P}(x_l^{\,j(l)}) = f'_{i(i),j(j)}(x_l^{\,j(l)}), l = \{1,...,n_l\} \qquad (12)$$

It is tensor containing all the shifts (which are vectors) at all the generations $l = \{1,...,n_l\}$. The term *pyramidal* is used, because there are fewer shifts at a generation $l-1$ than at the next one l. Indeed, at $l=1$, there are N shifts, at $l=2$, N^2 shifts (cf. the fractalian built after 2 iterations on the figure 4 has $N^2 = 4^2 = 16$ shifts), etc. A graphical interpretation of this tensor is given in figure 6.

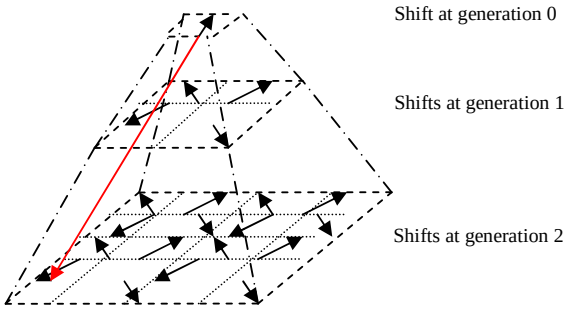

Fig. 6: pyramidal tensor of the shifts impacted on vectors of generations 0, 1, and 2 (with $N=4$)

This formalization exhibit clearly the multiscale behaviour of the Context-Dependant IFS: a vector $x_{n_l}^{\,j(n_l)}$ from the leaves of a fractalian are built by the composition of the shifts from the previous generations, the ones which share the same context $j(l), l = \{1,...,n_l - l\}$, that is to say all the shift encountered by starting from the top of the pyramid through the shifts of same context (see the red arrow of figure 6).

From the organization in a hierarchy found by Recursive Similarity Hashing, a similar pyramidal tensor can be constructed, by decomposing the positions of the target data points in all the shifts of the centers of masses of the groups, at each scale, in which the points are included. Formerly:

$$x_g(c(l,i)) - x(i) = \mathrm{P}(x_l^{\,c(l,i)}), l = \{1,...,n_l\} \qquad (13)$$

The index i is the index of the data point x, $x_g(c(l,i))$ represents the center of masses of the group of coordinate $c(l,i)$ in the tree created by hierarchical clustering of the data (see section 2.2).

By comparing formula (16) and (17), the equivalence between a Recursive Similarity Hashing and a Context-Dependant IFS appears clearly. The hierarchical clustering of the data creates a pyramidal tensor from which the shifts of the Context-Dependant IFS can be defined.

H. Reduction of the pyramidal tensor

The final step in finding a *simple* representation (the Context-Dependant IFS) from the data is to simplify the number of parameters of the pyramidal tensor. Initially, from a data set of size $n = n_e^{n_l}$, there should be $\sum_{l=`}^{n_l} n_e^{\,l}$ different shifts in the pyramidal tensor (which is more than the initial data of the systems studied). The process of finding a model is to simplify the tensor, by statistical assumptions, so that there are as few parameters as possible.

In order to do so, *co-scale* frequency analysis (which can now be defined rigorously as the frequency analysis of the shifts of an identical generation l), and *trans-scale* frequency analysis (on the shifts sharing an identical context) is conducted.

Contrary to other multi-scale analysis, it is possible to refine the precision of this frequency analysis for each scale.

Before moving to the implementation itself, an illustration of what are *co-scale* and *trans-scale* similarities could be the following: if one should study the behaviors of families through generations, one should focus on two aspects: the role of a member of the family for all families of a given generation (for example, the role of the father in archaic latin society, tha *pater*); and the changing of the role of this member through the generations (from the *pater* to the Japanese good-father).

Searching for a *co-scale* similarity is equivalent to looking at the local behaviors, which should be identical whatever the context; *trans-scale* similarities are the variations (if any) of these similarities with the context, or scale. The latter should be continuous in real data, since there are no rupture between the behaviors from two close scales; while there could be between two very different scales. The over whole scheme of the Recursive Similarity Hashing is presented in figure 7.

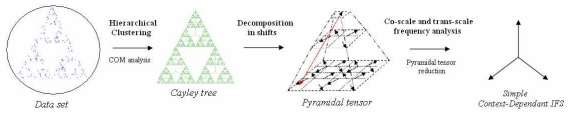

Fig. 7: Recursive Similarity Hashing process flow

III. Implementation and Applications

Our implementation of a prototype for the Recursive Similarity Hashing was done in C++. The difficulty of its practical development is that at all steps in the process the structures should be trees; and therefore, the method to skim across them should be recursive.

A. Agglomerative or divisive hierarchical clustering

The choice of the clustering technique is crucial, since it is the first step that will determine the next ones. Prior attempts to cluster fractals, based on the fractal dimension to define the cluster[23], the definition of balances boxes[24] (boxes containing an equal cardinal of elements) were not considering the conditions (1) and (2) described previously, vector machines, or simply the grid of a fractal compression[25], are not suited for such a clustering.

There are two ways to proceed, by an *agglomerative* or *divisive* approach. The agglomerative starts from the data points, and groups them step by step to their nearest neighbors, so that the groups formed are as far as possible from one another (according to condition (2)). This naïve approach is efficient if the data is well formatted (if for example it has been generated by an Context-Dependant IFS respecting conditions (i) and (ii)); however, and this is a common problem, early mistake in the clustering creates problem in higher clusters. Isolated points appear which induce a large deviance on the condition (2) at higher generation (see figure 8).

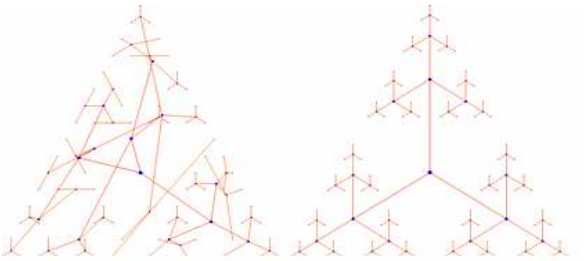

Fig. 8: Cayley tree result of the hierarchical clustering: if the initial clustering is false (left); otherwise (right)

The divisive approach is less subject to error in the early steps of the recursion.

B. Implementation of CMO

The problem to be solved by the Center of Masses Optimization described in section 2.4 is in a way similar to the k-means clustering[26]; however, instead of selecting the combination of groups for which the distance to the center of masses is minimal, groups which are as far as possible from the others are chosen. This is in ad equation with the condition (2) of section 2.3 described previously. Moreover, the number of elements per group is fixed (whereas it is the number of groups in k-means), and the clustering should be done recursively on the groups obtained.

The combinatronics problem behind the CMO is computationally very expensive. Similarly to K-means, the heuristic implementation of CMO is designed to start with a random combination of vectors, and iteratively correct the groups chosen; which reduces the complexity of the algorithm.

The complexity of the algorithm is not the purpose of this paper, and therefore CMO can be simply understood as a selection all possible combinations of vectors to form all possible groups, and then all possible combinations of groups to calculate the discriminative function.

C. Implementation the reduction of pyramidal tensor

To reduce the number of parameters of the pyramidal tensor P, it is assumed that the drifts (which are vectors) are simple affine transformations. Therefore, the contraction ratio C_r and angle α between two drifts can be defined. By constructing a frequency spectrum of the contraction ratios and angles, for

each generation (*co-scale*) and for the all the drifts sharing the same context (*trans-scale*), the self-similarities at multi-scales can be analyzed.

IV. Results and Discussion

To illustrate the efficiency of Recursive Similarity Hashing on real fractals, results on a tree fractal with 3 branches are detailed here. Through the paper, results on the Sierpinski Gasket were also given. The experiment is based on the Recursive Similarity Hashing of 9 points of the fractal, built so that all the points of a common generation are drawn (this implicitly means that the fractal has not been built with the Chaos game, but with by the growth approach[22]). In the two cases, the data is minimal: the minimum number of points is drawn so that the IFS can be determined from them (9 points to find the 3 mappings of the Sierpinski gasket and tree with 3 branches). In both cases, a least two generations (in the sense of the Context-Dependant IFS) are needed to understand the trans-scale similarities. By doing so: (a) groups containing features of similarities can be found, recursively, so that a hierarchical organization of the fractal can be extracted by the CMO analysis (see figure 9 for the result of the CMO analysis, figure 10 for the associated Cayley tree of the organization in a hierarchy), (b) the pyramidal tensor computed, (c) a simple Context-Dependant IFS that builds the fractal (see figure 11) can be extracted.

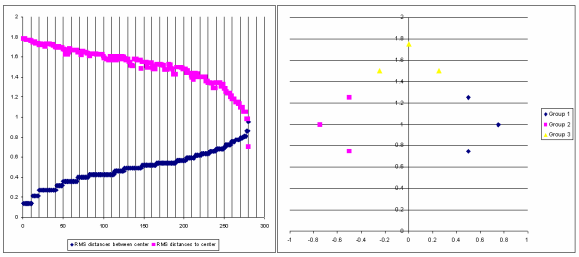

Fig. 9: CMO analysis on 9 points of a tree with 3 branches fractal (choice of the clusters (cf. figure 9) on the right, 3 clusters determined by CMO on the left)

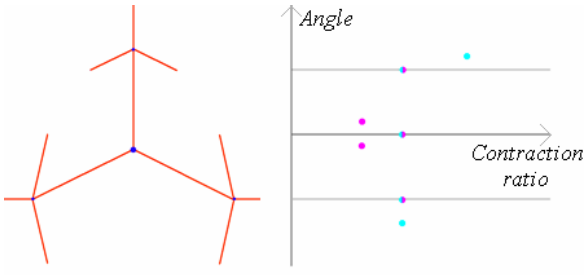

Fig. 10: Associated Cayley tree constructed with the result of the hierarchical clustering (right) and frequency spectrum in angle and contraction ratio of the mappings between clusters (left)

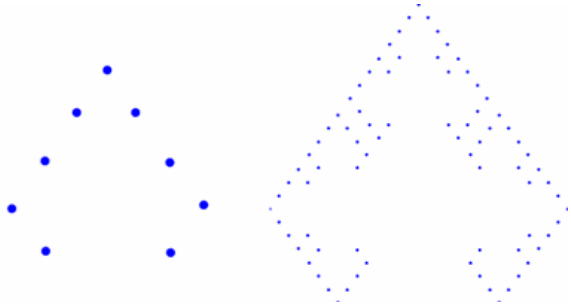

Fig. 11: Reconstructed tree with 3-branches fractal after 2 iterations (right) and 4 iterations (left)

The Recursive Similarity Hashing has found a Context-Dependant IFS, different from the one used to build the fractal, which can reconstruct the original fractal with in theory no error. This examples illustrate how can Recursive Similarity Hashing can “invert” many usual fractals, and find a Context-Dependant IFS to rebuild them.

V. Conclusion

Recursive Similarity Hashing and the Context-Dependant Iterated Function Systems are complementary tools to study self-similarities at multi-scale. By decomposing an object into its scales, these methods have the promising properties to be able to find self-similarities from fractal-data, and more generally from any data exhibiting self-similarities. It has been shown that, contrary to other methods to solve the inverse problem of building fractals, this novel approach finds a rigorous inversion, which puts into a simpler relationship the spaces where fractals and Context-Dependant IFS lie.

Possible real world applications are numerous (from growth phenomena, to information coding). Moreover, the similarity of Context-Dependant IFS with dynamical systems is a topic to be explored more in details in further research.

Acknowledgment

This work was supported by the Global Center for Excellence program (GCOE), Secure Life Electronics and the school of Engineering of the University of Tokyo.

References

[1] S. Wiggins, *Introduction to Applied Nonlinear Dynamical Systems and Chaos*, (Texts in Applied Mathematics 2, Springer-Verlag)
[2] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical systems and Bifurcations of Vector Fields* (Springer, 1983)
[3] Pavliotis GA, Stuart AM, *Multiscale Methods Averaging and Homogenization* (New York , Springer, 2007 , ISBN: 9780387738284)
[4] F. Takens, *Detecting strange attractors in turbulence* (Dynamical Systems and Turbulence, Lecture Notes in Mathematics, vol. 898. Springer-Verlag. pp. 366–381)
[5] H. E. Stanley and P. *Meakin, Multifractal Phenomena in Physics and Chemistry, Nature 335*, 405-409 (1988).
[6] Tim Palmer, Paul Williams, *Stochastic Physics and Climate Modelling* (Royal Society Publishing, 2008, ISBN 9780854036950)

[7] L. A. N. Amaral, A. Scala, M. Barthelemy, and H. E. Stanley, Classes of Behavior of Small-World Networks (*Proc. Natl. Acad. Sci. 97*, 11149-11152 (2000))
[8] G. Binnig et al., Will machines start to think like humans? (*Europhysics News* (2002) Vol. 33 No. 2)
[9] Palmer, T. N., The Invariant Set Postulate: a new geometric framework for the foundations of quantum theory and the role played by gravity (*Proceedings of the Royal Society a Mathematical Physical and Engineering Sciences 465*: 3165. doi:10.1098/rspa.2009.0080)
[10] Michael F. Barnsley, *Fractal Everywhere* (second edition, Hawley Rising)
[11] Paul S. Addison, *The Illustrated Wavelet Transform Handbook* (Institute of Physics, 2002, ISBN 0-7503-0692-0)
[12] P. Abry, P. Gonçalvès & J. Lévy-Véhel, *Scaling Fractals And Wavelets* (iSTE Publishing Company, 2005)
[13] John C. Hart, Wayne O. Cochran, Patrick J. Flynn, Similarity Hashing: A Computer Vision Solution to the Inverse Problem of Linear Fractals (Washington State University, 2008)
[14] C.R Handy and G. Mantica, Inverse problems in fractal construction: moment method solution *(Physica D 43* (1990) 17-36)
[15] R. Rinaldo and A. Zakhor, Inverse and Approximation Problem for Two-Dimensional Fractal sets (*IEEE trans. on image processing*, Vol.3, No. 6)
[16] R. Shonkwiler, F.Mendivil, A.Deliu, Genetic Algorithms for the 1-D Fractal Inverse Problem (Georgia Institute of Technology)
[17] Timothee Leleu, Akito Sakurai, Recurrent self-similarties and machine learning: the inverse problem of buiding fractals (*Proceedings of Mendel 2009*)
[18] Xin Zhou and David P. Tuck, MSVM-RFE: extensions of SVM-RFE for multiclass gene selection on DNA microarray data (*Bioinformatics 2007* 23(9):1106-1114)
[19] Jörg Sander, Martin Ester, Hans-Peter Kriege, Hans-Peter Kriegel, Xiaowei Xu, Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Application (*Data Mining and Knowledge Discovery archive*, Volume 2 , Issue 2 , ISSN:1384-5810 (June 1998))
[20] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, A Local Search Approximation Algorithm for k-Means Clustering (*Computational Geometry: Theory and Applications*, 28 (2004), 89-112.))
[21] Sloan's A008277, The On-Line Encyclopedia of Integer Sequences
[22] Timothee G. Leleu, Building "invertible" fractals: Introduction to Context-Dependant Iterated Function Systems, Proc. 2010 International Joint Conference on Neural Networks (IJCNN 2010)
[23] D Barbará, P Chen, Using the fractal dimension to cluster datasets (*Proceedings of the sixth ACM SIGKDD*, 2000)
[24] CA Duncan, MT Goodrich, SG Kobourov, Balanced aspect ratio trees and their use for drawing very large graphs (*Lecture Notes in Computer*, Springer, 1998)
[25] Michael F. Barnsley and Lyman P. Hurd, *Fractal Image Compression*, ISBN 0-86720-457-5
[26] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, An efficient k-means clustering algorithm: Analysis and implementation (*IEEE Trans. Pattern Analysis and Machine Intelligence*, *24* (2002), 881-892)
[27] Chris Ding and Xiaofeng He, K-means Clustering via Principal Component Analysis (*Proc. of Int'l Conf. Machine Learning (ICML 2004)*, pp 225-232. July 2004)
[28] T. Vicsek, Fractal Growth Phenomena, 2nd ed. (*World Scientific*, Singapore 1991).
[29] H. A. Makse, J. S. de Andrade, M. Batty, S. Havlin, and H. E. Stanley, Modeling Urban Growth Patterns with Correlated Percolation (*Phys. Rev. E*, 1 December, 1998)