GRAPH-THEORETICAL ANALYSIS OF THE FRACTAL TRANSFORM

Jaroslaw Domaszewicz and Vinay A. Vaishampayan

Department of Electrical Engineering
Texas A&M University
College Station, Texas, 77843, USA
409/847-9068, Fax 409/845-7161
domaszew@eemips.tamu.edu, vinay@ee.tamu.edu

ABSTRACT

A part of a fractal code is an assignment of a domain block to every range block. The assignment is used to construct the dependence graph of a fractal code. The vertices of the graph represent the range blocks. Two vertices x and y are connected by a directed edge from y to x if the range block y is overlapped, fully or partially, by the domain block assigned to the range block x. An algorithm to analyze the structure of the dependence graph is presented. The exposed structure of the graph can be used for three different purposes. The first one is convergence analysis: the affine transformations linking domain and range blocks can be classified into those that affect convergence and those that do not. The second one is decoding time reduction: certain range blocks can be reconstructed in a non-iterative way. The third one is improving upon collage coding: the affine transformations for some range blocks can be optimized based on the domain blocks extracted from the reconstructed rather than the original image.

1. INTRODUCTION

Fractal transform algorithms [1, 2, 3, 4] present a novel approach to image compression. A simple fractal transform works as follows. The original image is partitioned into non-overlapping square *range blocks* of size $B \times B$. Also, a set of *domain blocks* of size $nB \times nB$ is extracted from the original. Usually a domain block is twice as large as a range block. Next, the algorithm finds the best matching domain block for every range block. The domain blocks are not compared against a range block directly. Rather, for each domain block, an affine transformation which minimizes the distortion between the domain block and the target range block is first determined. The best matching domain block along with its optimized transformation is assigned to the target range block. The procedure is repeated for all the range blocks in the image.

The fractal code is a list of domain blocks and the corresponding transformations, assigned to all the range blocks. Once a fractal code for the original image has been determined in the encoder, it is sent to the decoder. The decoder uses an iterative procedure to obtain the reconstructed image. The reconstructed image is the limit of a sequence of images generated in the decoder by iteratively applying the fractal code to an arbitrary initial image.

In this paper we examine how the range blocks depend on one another through domain blocks. We introduce a concept of the dependence graph of a fractal code, and provide an algorithm to analyze the graph. The dependence graph can be constructed for a variety of fractal transforms, including ones that use adaptive image partitioning into range and domain blocks of different sizes. We allow a range block to be encoded without a domain block, i.e., using the fixed space only. Also, the graph is independent of the form of a transformation applied to domain blocks; a transformation need not be affine.

The structure of the dependence graph, which we expose, can be used in at least three application areas. The first one deals with the behavior of a fractal code when iterated at the decoder. We can classify the affine transformations into those which are responsible for convergence of the iterated sequence, and those which are irrelevant to this property. The second application is reducing the decoding time: not all of the range blocks have to be decoded iteratively, some of them can be decoded in a one-step calculation. Finally, one can improve upon collage coding. In collage coding, the transformations are optimized based on the domain blocks extracted from the original image. However, after the decoding, the same transformations act on the domain blocks extracted from the reconstructed image. Since the reconstructed image is different from the original one, collage coding leads to suboptimal codes. By using the structure of the dependence graph, transformations assigned to some range blocks can be optimized based on the domain blocks extracted from the reconstructed image.

Remarkably, all these different applications are based on the same properties of the dependence graph.

2. DEPENDENCE GRAPH AND DECODABLE REGIONS

Assume that a fractal code is given. A part of the fractal code is an assignment of a domain block to every range block.

Def. 1 The *dependence graph* of a fractal code is a directed graph (R, P) whose set of vertices is the set of all range blocks R and the set of edges is given by $P \triangleq \{\langle y, x\rangle \in R \times R : y \text{ is overlapped, fully or partially, by the domain block assigned to } x\}$.

The dependence graph is illustrated in Fig. 1. We write yPx, and say that x depends on y, to denote $\langle y, x\rangle \in P$. We also use the following notation. If $x \in R$ and $A \subset R$,

0-7803-2431-5/95 $4.00

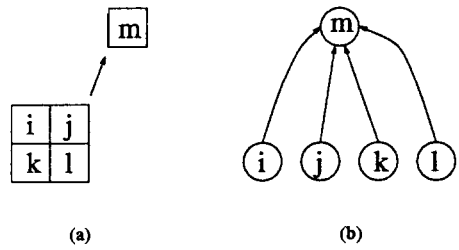

Figure 1: The dependence graph. (a) A domain block overlapping the range blocks i, j, k, and l is assigned to the range block m. (b) The corresponding fragment of the dependence graph.

then $P(x) \triangleq \{y \in R : yPx\}$ and $P(A) \triangleq \bigcup_{x \in A} P(x)$. For example, the set $P(A)$ is the set of all range blocks which are overlapped by a domain block assigned to any of the range blocks in the set A.

Def. 2 A non-empty set of range blocks $A \subset R$ is called a *decodable set* if and only if $P(A) \subset A$. A set of range blocks $A \subset R$ is called a *minimal decodable set* if it is decodable and does not contain a proper decodable subset.

A set of range blocks is decodable if all the domain blocks assigned to the range blocks in the set are totally overlapped by these range blocks. Hence, if $A \subset R$ is a decodable set, one can decode (i.e., find the reconstructed version of) the range blocks in A without decoding or referencing the remaining part of the image.

One can show (see also [5]) that if $A \subset R$ is a minimal decodable set, and $x \in A$ is such that $P(x) \neq \emptyset$, then x belongs to a cycle of vertices from A. If $P(x) = \emptyset$ (i.e., x is encoded using the fixed space only), then $\{x\}$ itself is a minimal decodable set.

Assume that a decodable set is given. We now show how to grow a bigger decodable set, related to the given one in a special way.

Def. 3 Let $A \subset R$. Then $N(A) \triangleq \{x \in R : P(x) \neq \emptyset \text{ and } P(x) \subset A\}$, and $N^*(A) \triangleq \bigcup_{n=0}^{\infty} N^{\circ n}(A)$.

The set $N(A)$ is the set of all range blocks whose assigned domain blocks are totally overlapped by the range blocks in A. Now assume that A is decodable. We immediately have $A \subset N(A)$. It follows that $N(A)$ is decodable as well. Also, since $N(\cdot)$ is monotone, we get $N^{\circ n}(A) \subset N^{\circ(n+1)}(A)$ for all $n \geq 0$, i.e., the sequence in the definition is an increasing sequence of decodable sets. Note that $N(\emptyset) = N^*(\emptyset) = \emptyset$.

Assume that the part of the reconstructed image supported by the range blocks in a decodable set A is known. It is possible to extend the support of the reconstructed part to $N(A)$ in a *one-step* (non-iterative) calculation. For each range block in $N(A)$ (not already in A), it suffices to extract the reconstructed domain block from the part supported by A, and to perform the assigned transformation. After decoding the range blocks in $N(A)$, it is possible to decode in one step those in $N(N(A))$. One can continue this way until all the range blocks in $N^*(A)$ have been decoded. Hence $N^*(A)$ is the union of A and the set of all range blocks whose reconstructed version can be found non-iteratively once the reconstructed version of each range block in A is known.

It can be shown that if $A \subset R$ is a decodable set, and $x \in N^*(A) \setminus A$, then x does not belong to a cycle.

3. THE ANALYSIS ALGORITHM

This section introduces an analysis algorithm which exposes the structure of a dependence graph. Assume the dependence graph of a fractal code is given. As a result of the algorithm, the set of vertices (i.e., the set of all range blocks R) is partitioned into a collection of disjoint sets $\{R_0, R_1, \ldots, R_n\}$. The number of sets $n+1$ depends on the graph being analyzed.

We explain the algorithm in terms of minimizing the decoding time at the fractal decoder. As hinted in the previous section, not all of the range blocks have to be processed iteratively at the decoder. For example, if A is a decodable set, then one can perform iterative decoding on A, and find the reconstructed version of the range blocks in $N(A) \setminus A$ in one step.

Let us explore the limits of this idea. We would like to decode non-iteratively as many range blocks as possible. If there are range blocks encoded using the fixed space only, we decode these first, obviously without iterating. Let $R_{-1} = \{x \in R : P(x) = \emptyset\}$ be the set of all such range blocks. Once the reconstructed version of the range blocks in R_{-1} is known, we can decode non-iteratively all the range blocks in $N^*(R_{-1})$, but not others. Hence we define the first set as $R_0 = R_{-1} \cup N^*(R_{-1})$.

Some iterative decoding is now necessary. We already know the reconstructed version of the range blocks in R_0, so the dependence of other range blocks on these is not critical. Hence we remove all the vertices in R_0 (as well as all the edges leaving or entering these vertices) from the dependence graph. We have to choose some decodable set of the *modified* dependence graph in order to find the reconstructed version of additional domain blocks. We want the set to be as small as possible so as to minimize the number of range blocks decoded iteratively. All range blocks belonging to minimal decodable sets of the modified dependence graph have to be decoded iteratively. Hence, the union of all minimal decodable sets of the modified graph is the next set, R_1. Iterative decoding is performed on R_1.

After that, non-iterative decoding can be done for all the range blocks in the set $N^*(R_1) \setminus R_1$ (here $N^*(\cdot)$ refers to the modified dependence graph), but not for others. Hence we set $R_2 = N^*(R_1) \setminus R_1$. Non-iterative decoding is performed on R_2.

We now remove from the modified graph all the additional vertices that have been decoded (i.e., $R_1 \cup R_2$). As in the case of R_1, we set the next set, R_3, to be the union of all minimal decodable sets of the graph thus obtained. (Notice that some range blocks in R_3 must depend on some range blocks in $R_0 \cup R_1 \cup R_2$.) Iterative decoding is performed on R_3.

The range blocks in the set $R_4 = N^*(R_3) \setminus R_3$ are again decoded non-iteratively. We continue this alternating between iterative and non-iterative decoding, until all

```
1   R_{-1} ← {x ∈ R : P(x) = ∅},  C ← R_{-1},  n ← −1
2   while (C ≠ R)
3         n ← n + 1
4         if (n even)
5               R_n ← N*(R_{n−1}) \ R_{n−1}
6               remove(R_{n−1} ∪ R_n)
7         else
8               R_n ← ⋃{A : A ∈ AllMinDec()}
9         C ← C ∪ R_n
10  R_0 ← R_0 ∪ R_{-1}
11  return {R_0, R_1, ..., R_n}
```

Figure 2: The analysis algorithm

the range blocks have been classified and decoded.

We call the odd-numbered sets (R_1, R_3, etc.) the *looped* sets, and the even-numbered ones the *straight* sets. Combining the observations made in the previous section, we conclude that the set of all range blocks decoded iteratively, $\bigcup_{i \text{ odd}} R_i$, is equal to the set of all range blocks that belong to a cycle. Similarly, the set of all range blocks decoded in a single step, $\bigcup_{i \text{ even}} R_i$, is equal to the set of all range blocks that do not belong to a cycle.

The analysis algorithm is presented in Fig. 2. The algorithm AllMinDec() returns the collection of all minimal decodable sets of a graph (see the Appendix). The operation remove(A) removes from a graph all vertices contained in A, as well as all edges leaving or entering these vertices. Note that the operations like AllMinDec() or $N(\cdot)$ act mostly on the modified graphs, which are different from the original dependence graph due to vertex removal. For simplicity, this is not directly indicated in the notation. A contrived graph and the results of running the algorithm with the graph as the input are presented in Fig. 3.

The analysis algorithm has been implemented. For a simple fractal transform and the "Lena" image, there are four sets—R_1, R_2, R_3, and R_4. They contain 2800, 1280, 4, and 12 range blocks, respectively.

4. APPLICATIONS

4.1. Speeding-up the Fractal Decoder

The possibility of speeding up the decoding process due to the structure of the dependence graph is explained in the previous sections. Note that a fractal code restricted to each looped set specifies a dynamical system. Sample values in the range blocks from the set form the state of the system. However, while for R_1 the dynamical system is autonomous, the systems corresponding to R_3, R_5, etc., have "inputs" (sample values in some range blocks in the lower-numbered sets). The decoding algorithm proposed in this paper has an additional advantage that these dynamical systems are iterated only after their inputs have been made constant. This is not the case in the standard decoder.

To achieve further speedup, one should break down the iterative decoding of a looped set into a sequence of steps. Each step should consist of the iterative decoding of a single minimal decodable set contained in the looped set. This

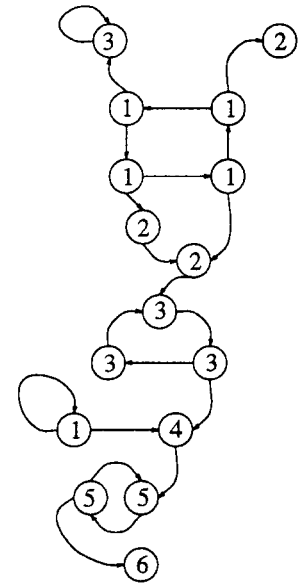

Figure 3: A graph whose set of vertices has been partitioned by the analysis algorithm. A vertex with an index n belongs to R_n.

is because a fractal code, when iterated on different minimal decodable sets, may produce sequences converging at different rates.

Obviously, the penalty for the decoder speedup is some bit rate overhead which is needed to add a description of the structure of the dependence graph to a fractal code.

4.2. Convergence Analysis

The traditional decoder generates a sequence of images by iterating a fractal code. One can easily see that convergence of the sequence depends solely on the parameters of the affine transformations associated with the range blocks from the looped sets. Moreover, the interaction between the parameters is limited to a single minimal decodable set. The parameters corresponding to the range blocks from straight sets do not affect convergence in any way. For example, one can use arbitrary values for the scaling coefficient and still obtain a fractal transform which behaves like a contractive map. The exposed structure of the dependence graph seems to be a natural and fundamental concept for understanding convergence properties.

4.3. A New Encoding Algorithm

Finally, one can use the graph-theoretical approach to improve upon collage coding. We briefly describe the new encoding procedure, which we call the *re-encoding algorithm*. An initial fractal code is obtained using collage coding. For simplicity assume that every range block has an associated domain block, i.e., $R_0 = \emptyset$. One then executes the analysis algorithm to partition the set of range blocks. Next, iterative decoding is performed on the decodable set R_1. This way we know the reconstructed versions of the domain blocks covered by the range blocks in R_1. Range blocks

in R_2 are now re-encoded (i.e., the affine transformations are re-optimized) using the *reconstructed* versions of the domain blocks. We continue alternating iterative decoding and re-encoding. As a result, all range blocks in the straight sets are re-encoded. Preliminary results for the re-encoding algorithm have been obtained. For a simple fractal transform algorithm and the "Lena" image, a slight improvement of PSNR from 26.99 to 27.20dB is observed.

5. SUMMARY

A concept of the dependence graph of a fractal code is introduced. The graph reflects how domain blocks are assigned to range blocks. Partitioning the set of range blocks according to the structure of the dependence graph helps understand convergence properties and improve the existing encoding and decoding algorithms.

6. APPENDIX

In the Appendix, we consider an efficient implementation of the analysis algorithm. The only non-trivial operation is AllMinDec(). We begin by characterizing the sets returned by AllMinDec().

First, we want to generally describe the results of the analysis algorithm in terms of strongly connected components (see [6]) of the dependence graph. The following result is easy once we realize that all elements of a strongly connected component which contains more than one element belong to a cycle.

Lemma 1 *$\bigcup_{i \text{ even}} R_i$ is the union of all strongly connected components which contain exactly one element, and the element does not have a self-loop. $\bigcup_{i \text{ odd}} R_i$ is the union of all strongly connected components which either contain two or more elements, or contain exactly one element, and the element has a self-loop.*

The next lemma establishes a simple relationship between minimal decodable sets and strongly connected components.

Lemma 2 *A set $A \subset R$ is minimal decodable, if and only if it is decodable, and it is a strongly connected component of the dependence graph.*

Proof. $\Longrightarrow$ Let A be minimal decodable. Pick $x \in A$, and let $B = \{y \in A : x \text{ is reachable from } y\}$. Note that $x \in B$, so B is non-empty. Pick $y \in B$, and let $z \in P(y)$. The set A is decodable, so $z \in A$. Since y is reachable from z, and x is reachable from y, we conclude that x is reachable from z. Hence $z \in B$, and so B is decodable. Since A is minimal decodable, we have $B = A$. The element x was chosen arbitrarily, and so any two elements in A are mutually reachable. Hence A is a subset of some strongly connected component C. Since A is decodable, any element in A is not reachable from any element in $C \setminus A$. Hence we must have $C = A$, and A is a strongly connected component. $\Longleftarrow$ Let $B \subset A$ be a decodable subset of A. Then any element in B is not reachable from any element in $A \setminus B$. Since A is a strongly connected component, we must have $A \setminus B = \emptyset$, and so A is minimal decodable. □

Assume that a graph is given, and let $C_1, C_2, \ldots, C_k$ be some of the strongly connected components of the graph. Assume that the operation remove$(C_1 \cup C_2 \cup \cdots \cup C_k)$ has been performed on the graph. Let C_{k+1} be a strongly connected component of the *modified* graph. Then C_{k+1} is also a strongly connected component of the original graph. This observation, along with Lemma 1 and Lemma 2, leads to the following characterization of the sets returned by AllMinDec().

Lemma 3 *Let A be a set calculated in step 8 of the analysis algorithm (i.e., a minimal decodable set of a modified dependence graph). Then A is a strongly connected component of the original dependence graph.*

The analysis algorithm can now be efficiently implemented as follows. First, we use any standard algorithm to find all the strongly connected components of the dependence graph. For each strongly connected component, we use Lemma 1 to determine whether it belongs to a straight set or a looped set. Second, we find the component graph (i.e., the graph in which there is one vertex for each strongly connected component of the original graph). Finally, we run a simplified analysis algorithm on the component graph. In the simplified algorithm, the operation $N^*(\cdot)$ accumulates only those components that belong to straight sets in the original dependence graph. Since the component graph is acyclic, the operation AllMinDec() reduces to finding all vertices x such that $P(x) = \emptyset$. In the simplified analysis algorithm, all steps are quite simple.

7. REFERENCES

[1] Arnaud E. Jacquin. Fractal image coding: A review. *Proceedings of the IEEE*, 81(10):1451–1465, October 1993.

[2] Yuval Fisher, editor. *Fractal Image Compression.* Springer-Verlag, New York, 1995.

[3] Skjalg Lepsøy, Geir E. Øien, and Tor A. Ramstad. Attractor image compression with a fast non-iterative decoding algorithm. In *Proc. ICASSP'93*, pages V–337 – V–340, 1993.

[4] Bernd Hürtgen and Thomas Hain. On the convergence of fractal transforms. In *Proc. ICASSP'94*, pages V–561 – V–564, 1994.

[5] Jaroslaw Domaszewicz. *Encoding Algorithms for Fractal Compression.* PhD thesis, Texas A&M University, to be published.

[6] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, MA, 1990.