블락 외곽선의 기울기를 이용한 프랙탈 이미지 압축

박인영[0], 위영철
아주대학교 정보통신전문대학원
{parkiy7[0], ycwee}@ajou.ac.kr

Fast Fractal Image Compression Using the outer fence acceleration

In-Young Park[0], Young-Cheoul Wee
Dept. of GSIC, Ajou University

요 약

압축 방법에는 크게 손실(lossy)압축과 무손실(lossless)압축으로 나눌 수 있다. 그 중 프랙탈 이미지 압축은 lossy 압축의 한가지 방법으로서 개별적인 화소들에 대한 자료를 저장하기보다는, 영상 생성을 위한 명령이나 방식을 저장하는 방법이다. 특히 이미지의 내에 자기유사성(self-similarity)과 중복성(Redundancy)을 이용하여 관련성을 발견하고 수학적인 공식으로 표현하려는 방식이다. 그러나 이미지를 Domain과 Range로 블록화 한 후 유사한 이미지를 찾아내는 데 걸리는 시간이 상당히 길다. 여기에서는 Domain과 Range의 외곽선의 기울기의 부호를 이용하여 블록을 16가지로 클래스화 하여서, 전체의 Domain 블록을 탐색하는 데 걸리는 시간을 줄이고자 하였다. 전체 탐색을 하는 경우보다 10배 이상의 속도향상을 보였고, 이미지에 따라서는 PSNR 값의 손실도 없음을 보였다.

1. 서 론

Fractal 이란 용어는 Benoit Mandelbrot 가 다른 비율에서 자기 유사성을 가지는 객체를 나타내려고 하면서 처음 사용이 되었다. 이러한 객체들은 모든 배율에서 상세한 모습을 갖게 된다. 잘 알려진 예가 Mandelbrot Set 이다. Mandelbort는 압축을 위한 Fractal을 생각하지 않았다. 그는 구름, 나무, 산 같은 실제의 객체들을 모델링 하는데 쓰여질 수 있음을 보였다.

Michael Barnsley과 그의 동료들은 이미지 압축을 위한 Fractal 방법을 처음으로 인식하게 되었다. Barnsley는 반복함수계(Iterated Function System)을 발전시켰다. 그러나 압축과정을 완전히 자동화하는데는 실패하였고, IFS-based 압축은 실용화되지 못하였다.

Arnaud Jacquin은 이미지의 IFS를 찾는 대신에 이미지를 겹치지 않는(non-overlapping) range로 나누었고, local IFS를 찾으려는 생각을 가지고 있었다. 이것은 문제를 처리할 수 있는 작업으로 나누게 되었고, 자동화될 수 있었다. 그리고 부분반복함수계(Partitioned Iterated Function System)을 발전시켰다.[1]

2. 프랙탈 이미지 압축

2.1 프랙탈 이미지 압축의 개요

프랙탈 이미지 압축은 입력된 이미지를 블록화 하여서 최대한 자기유사성(self-similarity)가 가능한한 많이 반영되도록 하게 하는 방법이다. 각 블록마다 affine transformation을 사용하여서 블록과 블록사이의 매핑관계를 나타내게 된다.

기본적인 affine transformation 은 다음과 같다.

$$W\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

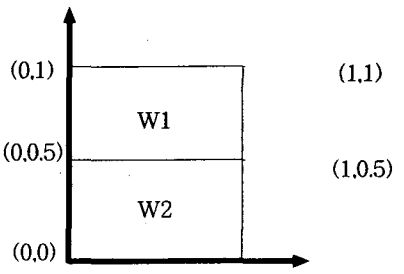


그림 1.

그림 1에서 a,b,c,d,e,f 는 다음과 같이 결정되어질 수가 있다.

$$W_1\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$W_1\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}$$

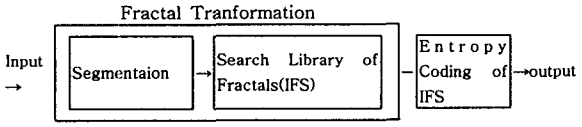
$$W_1\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

$$W_1\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

같은 방식으로 W_2를 구할 수 있다.

$$IFS = \begin{matrix} W_1: \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}+\begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \\ W_2: \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}+\begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{matrix}$$

프랙탈 압축 과정의 그림2 와 같이 요약될 수가 있다.



Fractal transformation은 많은 계산량을 필요로 하기 때문에 실시간 이미지 압축 시스템에서 가장 중요한 방해요소가 되고 있다.[2]

본 논문에서 프랙탈 이미지 압축 과정을 크게 블록분할(segmentation), 매핑획득(find mapping), 복원(Reconstruction) 으로 나누어서 진행하였다.

2.2. 블록분할(segmentation)

입력 이미지는 256*256의 해상도를 그레이 이미지이다. PIFS(Partitioned Iterated Function System)을 적용하기 위해서 이미지를 4*4 Non-overlapping 한 Range 블록으로 나눈다. 총 64*64개의 Range 블록으로 나뉘어 지게 된다. Domain 블록은 Range 블록 크기의 2배가 되는 8*8 이고 overlapping을 허용한다. 249*249개의 Domain 블록으로 나누어 지고, 두 블록(Range, Domain)의 유사성을 판별하기 위하여 작업이 필요하게 된다.

일반적인 exhausted 방식으로 찾아 나간다고 한다면, 하나의 Range 블록당 249*249개의 Domain 블록을 찾아나가야만 한다. 상당히 time consumed 하지 않을 수 없다.

본 논문에서는 블록의 4개의 꼭지점에서의 grey 값의 증감(기울기)를 이용하여 탐색을 해야할 Domain 블록을 16개로 클래스화 한다.

좌표에 대한 grey값의 함수를 f 라고 한다면, z=f(x,y) 이고 z의 증감 여부를 판별하게 된다.

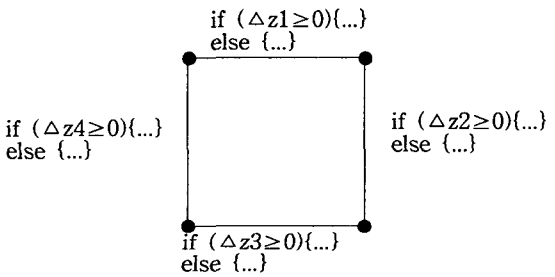


그림.2

기울기의 증감에 대한 2가지의 경우가 사각형의 4개의 변에서 알 수 있기 때문에 각각의 블록에 대해서 2^4 =16 가지로 클래스화 할 수가 있다.

각 Range 블록에 대해 먼저 클래스 매칭을 하고서 같은 클래스에 대한 Domain 블록에 대한 탐색을 하게 된다. Range 블록과 Domain 블록의 유사성을 따지기 위해서 여러 가지 거리법이 사용될 수 있는데, 여기서는 L_2-distance[3]를 사용한다.

$$d=\sqrt{\sum_{n=0}^{N-1}(x_n-y_n)^2}$$

2.3. 매핑획득(find mapping)

grey level 이 추가 될 경우에 affine tranformation 다음과 같이 나타낼 수 있다.

$$W\begin{bmatrix} x \\ y \\ z \end{bmatrix}=\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & s \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}+\begin{bmatrix} e \\ f \\ o \end{bmatrix}$$

여기서 s(contrast)와 o(brightness) 값이 필요하게 되는데, 그 이유는 유사블록을 찾은 것이 정확한 매핑이 아닌 근사적인 것이기 때문에 색상을 보정하는 부분이 필요하게 된다. s, o 값을 구하기 위해서는 Least Squares[4]를 이용하게 된다.

$a_1,...,a_n$(from D_i) , $b_1,...,b_n$(from R_i)가 주어졌을 때, 다음 식을 최소화하는 s와 o값을 구하게 된다.

$$R=\sum_{i=1}^{n}(s*a_i+o-b_i)^2$$

2.4. 복원(Reconstruction)

각각의 Range 블록에 대한 매핑 관계를 정해지고 나면, 이미지에 대한 encoding에 대한 작업이 끝나게 된다. 프랙탈 이미지 압축의 복원은 이러한 매핑 관계를 적용해 나간다. 축소변환(Contractive Transformation)에 기초하고 있기 때문에, 어떤 이미지로부터 복원을 시작하던지 간에 결과는 같게 나오게 된다.

3. 실험결과

입력이미지는 256*256 그레이 이미지이다.

사용한 PC는 CPU: P-Ⅳ 1.6GHz, Memory: 256M에서 수행하였다. 일반적인 exhausted 방식과 본 논문에서 제시한 외곽선의 기울기를 이용한 두 가지 방식으로서 시간과 PSNR[5]로서 자료를 해석하였다.

PSNR은 두 개의 이미지의 왜곡된 정도를 나타내는 수치로 공식은 다음과 같다.

$$PSNR(db)=10\log_{10}\frac{x^2}{\sigma^2}$$

3.1. Boat Image

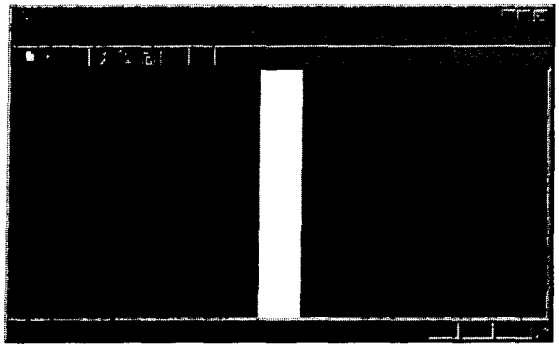


그림3. exhausted 방식의 원본 이미지(좌)와 복원된 이미지(우)

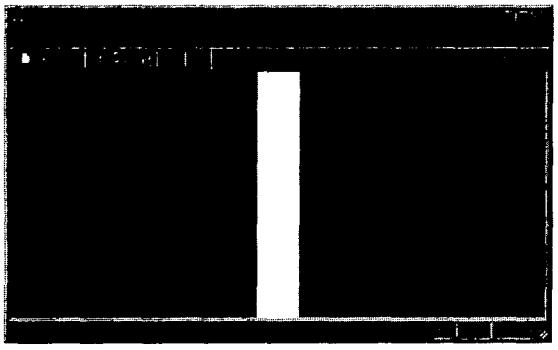


그림4. 외곽선의 기울기 방식의 원본 이미지(좌)와 복원된 이미지(우)

	PSNR(db)	Time(sec)
exhausted 방식	37.338991	160.020000
외곽선의 기울기 방식	36.991370	14.691000

3.2. Lenna Image

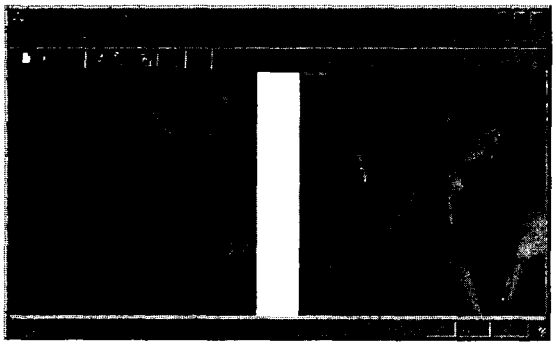


그림5. exhausted 방식의 원본 이미지(좌)와 복원된 이미지(우)

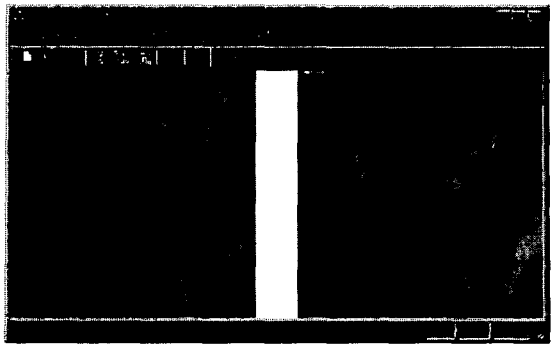


그림6. 외곽선의 기울기 방식의 원본 이미지(좌)와 복원된 이미지(우)

	PSNR(db)	Time(sec)
exhausted 방식	36.369891	159.980000
외곽선의 기울기 방식	36.369891	15.392000

4. 결론

결과에서 알 수 있는 바와 같이 속도면에 10배 이상이 빨라지게 되었고, 특히 Lenna 이미지 에서는 exhausted 방식과 질적인 차이의 저하가 없음을 알게 되었다.

본 실험에서는 contrast와 brightness의 임계값을 주지 않고 실험을 하였다. 임계값을 주어서 복구된 이미지가 어떻게 변화하는지 대한 연구가 이루어져야 하겠다.

본 논문에서 블록의 꼭지점을 이용하여 외곽선의 기울기의 증감을 파악하여서 블록을 클래스화 하였는데, 대각선이나 아니면 수학적인 도구를 이용하여 PSNR의 희생을 최소화하면서 압축 속도를 향상할 수 있는 다각적인 방법을 모색해 봐야 할 것이다.

5. 참고문헌

[1] Mark Nelson, Jean-Loup Gailly, "The Data Compression Book", pp.457-458, 1996

[2] Jianmin Jiang, "Image compression with fractals", Fractals in Signal and Image Processing, IEE Colloquium on, pp.7/1 -7/3, 1995

[3] M.de Berg, M.van Kreveld, M.Overmars, O.Schwarzkopf, "Computational Geometry", pp.146,160,314, 1991

[4] Yuval Fisher, "Fractal Image Compression", Siggraph '92 course notes, pp14

[5] Khalid Sayood, "Introduction to Data Compression", pp184, 1996