Fractal Compression

Alex Munoz
Algorithms Interest Group
29 June 2016

Outline

- Introduction
- Mathematical Background
- Connecting Mathematics to Compression
- Fractal Compression
- Coherent Example

Introduction

- Thinking about fractals
- How do we measure a the Australian coastline?
- Using different sized measuring sticks:
  500 km : 13,000 km
  100 km : 15,500 km
- The CIA World Facebook gives a measurement of 25,600 km

Introduction

- Self-similarity is the central idea
- While not as powerful or widely applicable as JPEG and wavelet compression techniques, fractal compression handles niche cases very well.
- In the very least, it is of mathematical interest and possible applications are not terribly obscure.

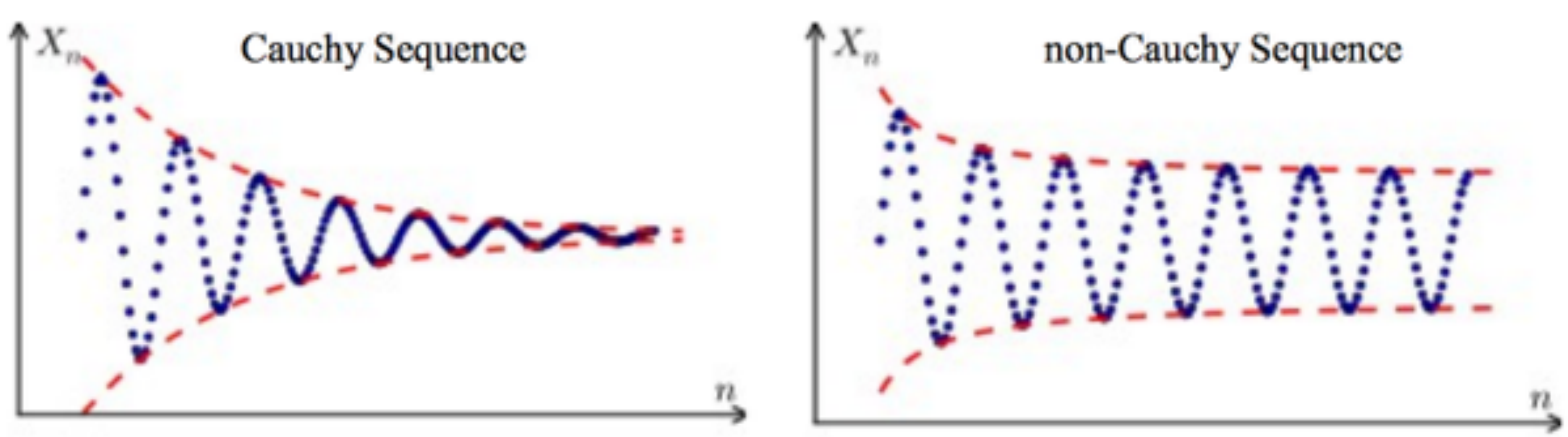
Introduction

- The intent is to use approximate redundancies in images to save space
- The logic we develop here carries over to functions and anything that we have a concept of ‘distance’ in
- Alternative uses include a method for identifying important constituents of objects e.g. a baseball is reduced to seams and leather

Mathematical Background

- Metric Spaces: a set S with a global distance function
  d(x,y) = 0 iff x=y
  d(x,y)=d(y,x)
  d(x,y)+d(y,z) => d(x,z)

- A Cauchy sequence:



Images from: http://en.wikipedia.org/wiki/Cauchy_sequence

Mathematical Background

- A mapping T: X —> X is a contraction mapping if: d(T(x),T(y)) =< c*d(x,y)

Figure 3: Contraction Mapping

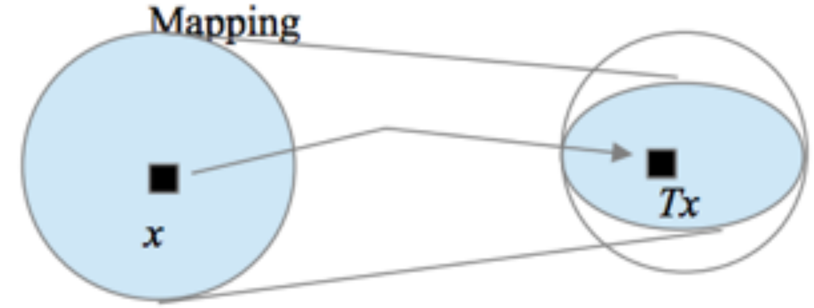
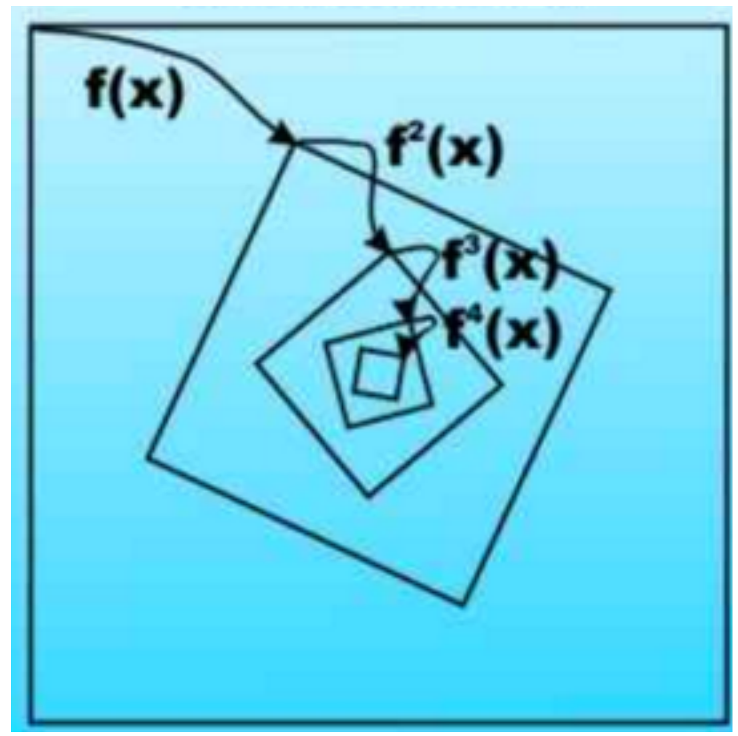


Image from: Hunter, Nachtergaele (2001) p 61

- If T: X—>X and T(x) —> x, we have a fixed point

- Contraction mapping on complete metric spaces have exactly one solution that is a fixed point

Mathematical Background



Demonstration of a fixed point as a consequence of contractive affine transformations

Mathematical Background

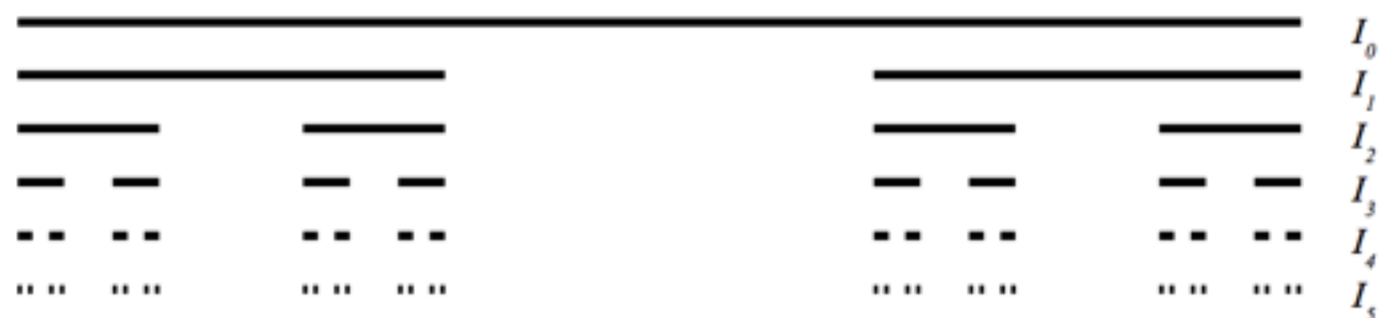
- Core idea for fractal compression: Iterated Function Systems (IFS)

- IFS: A finite set of contraction mappings w_n:X—>X on a complete metric space

- The IFS scales, translates, and rotates a finite set of functions

Mathematical Background

- Something a bit more concrete: “The Cantor Set” or “Cantor Comb”

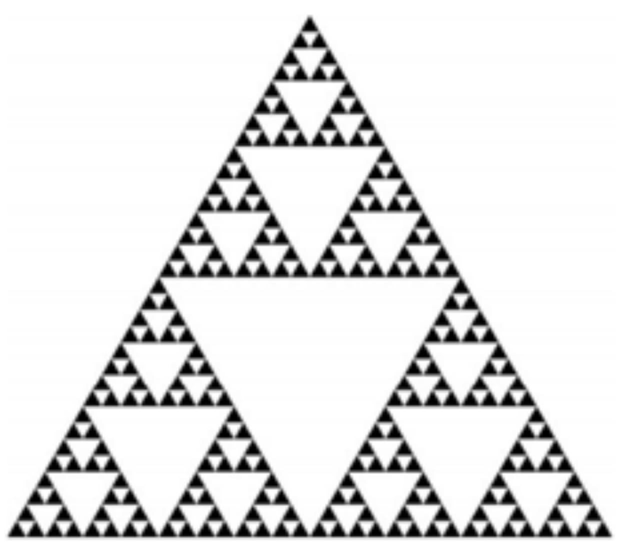


- $f_1 = x/3$ and $f_2 = x/3 + 2/3$ —>

$$
\begin{aligned}
I_1 &= f_1(I_0) \cup f_2(I_0) \\
I_2 &= f_1(I_1) \cup f_2(I_1) \\
&\vdots \\
I_n &= f_1(I_{n-1}) \cup f_2(I_{n-1})
\end{aligned}
$$

Mathematical Background

- IFS of affine transformations: $w_i(x) = w_i \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} = A_i x + t_i$



Sierpinski Triangle

Fern

Table 1: IFS code for a Sierpinski triangle

w	*a*	*b*	*c*	*d*	*e*	*f*	*p*
1	0.5	0	0	0.5	1	1	0.33
2	0.5	0	0	0.5	1	50	0.33
3	0.5	0	0	0.5	50	50	0.34

Table 2: IFS code for a Fern

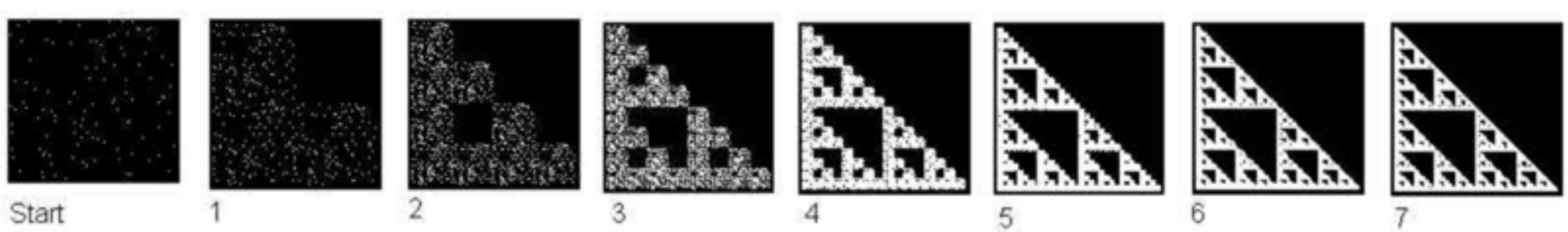
w	*a*	*b*	*c*	*d*	*e*	*f*	*p*
1	0	0	0	0.16	0	0	0.01
2	0.85	0.04	-0.04	0.85	0	1.6	0.85
3	0.2	-0.26	0.23	0.22	0	1.6	0.07
4	-0.15	0.28	0.26	0.24	0	0.44	0.07

Connection

- Given an IFS, there is a unique attractor which is the union of all of our transformations such that: $d(L,A) < e/(1-c)$
- Ultimately, this limit is what allows the use of IFS as a way to approximate images

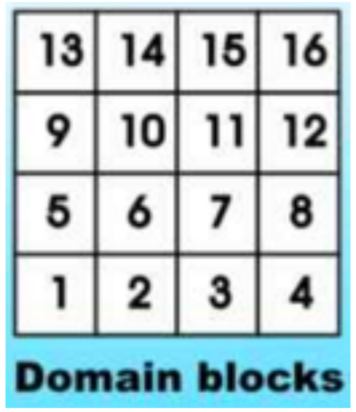
Connection

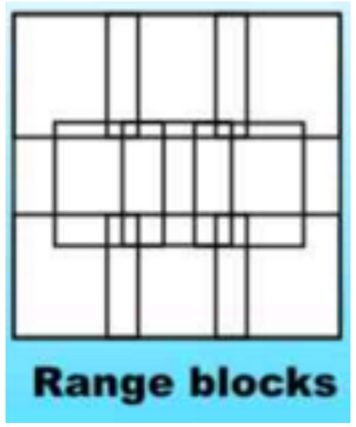
- We try to find an IFS that will generate a given image along with the IFS coefficients
- All that necessary is an image of the desired resolution and then iterating on it with the IFS will return our image



Fractal Compression

- In practice, you don't work on the entire image, you partition it and find like sections within the image
- Take a set of "domain" and "range" blocks and search for contractive transformations





Fractal Compression

- It is wise to restrict your class of transformations:

Transform	Type
0	identity
1	reflection in the y axis
2	reflection in the x axis
3	180 degree rotation
4	reflection in 45 degree line
5	90 degree rotation
6	270 degree rotation
7	reflection in -45 degree line

- Pseudocode for this geometry:

```
Divide image into range blocks
Divide image into domain blocks
FOR each domain block
    -Calculate effects of each transform
    on each range block
    -Find combination of transformations that
      map closest to image in the domain block
    -Record range block and transformation
```

Fractal Compression

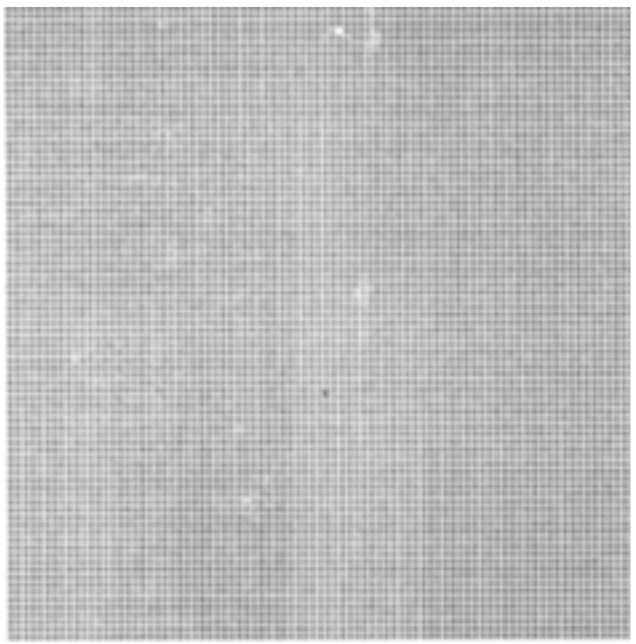
- The resulting fractal compressed image is the list of range blocks positions and transformations
- To reconstruct the image, iterate the entire set of transformations on the range blocks
- The Collage theorem guarantees that the attractor of this set of iterations is close to the original image

Fractal Compression

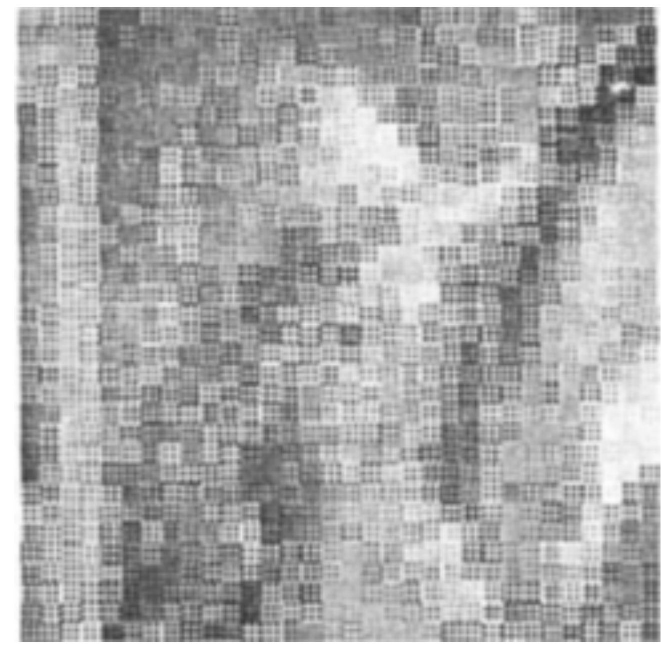
- If the set of transformations does not reach a cut-off for distance, the domain block is partitioned into 4 equally sized square children
- Extension to gray-scale or color
- Grayscale can be roughly interpreted as a depth in the image

Coherent Example

Start



1 Iteration



2 Iterations

10 Iterations

Advantages

- Images with a lot of affine redundancy are stored very compactly: Sierpinski triangle (1.2 KB) —> (18 Bytes)!

- Fractal methods are not scale dependent — compress to any resolution you like

- Fourier methods work poorly with discontinuities in images (Think Gibbs phenomenon), but fractal methods don't care

Resources

- https://www.youtube.com/watch?v=Lte3xpmH2_g
- http://www.i-programmer.info/babbages-bag/482-fractal-image-compression.html?start=1
- https://stepheneporter.files.wordpress.com/2013/02/colloquium-paper.pdf