A FAST BOX COUNTING ALGORITHM FOR DETERMINING THE FRACTAL DIMENSION OF SAMPLED CONTINUOUS FUNCTIONS

Hendrik F.V. Boshoff

Department of Electrical and Electronic Engineering,
University of Stellenbosch, Stellenbosch

Abstract

An adaptation is given of the box counting algorithm used for fractal dimension determination, to samples of a continuous signal. An iterative scheme is combined with an exploitation of continuity to yield a fast method. This allows us to reduce the algorithm's complexity from quadratic to linear. The method is applied to speech segments; in this context it is faster than morphological filtering and Hurst analysis, and has comparable performance. The new algorithm is also suitable to be parallellized.

The algorithm is applied to calculation of the dimension of fricative phonemes, and automatic segmentation of speech into periodic/noiselike segments. It is suggested that this dimension value may be used instead of zero crossing rate in some applications.

1 Introduction

Attempts to determine fractal dimension of speech segments, have to date been based on morphological filtering [1, 2] or Hurst analysis [3]. These methods are robust and reliable, but complex. Box counting, conceptually the simplest method, has been used to analyse signals originating in a turbulent combustion flow experiment [5].

Application of the box counting method was previously hampered, because the mathematical texts had stressed the limit of vanishing box size [4, pp 172–177]. The simplifications made possible by taking the limit are invalid in the case of discretized, sampled signals. The question of the relative scaling of time and amplitude also clouds the issue of dimension in the case of signals, the traces of which are self-affine, not self-similar. As long as the method is correctly applied however, *local* box counting dimension is a good estimation of the fractal dimension of a signal's trace [6].

2 Adapting and optimizing box counting

Box counting is well-known and trivial, but slow. The following can be noted when restricting attention to samples of continuous functions, regularly used to model signals:

- Assume that box counting must approximate the *smallest* number of squares necessary to cover a geometric set.
- The restriction to *functions* only, means we can count boxes in (and between) 'columns'; a function never folds back on itself.
- Assuming *continuity* of the function, allows us to retain only the highest and lowest value in each column. Every box in between will be entered by the function.
- Starting with a number of samples which is a power of two, allows the number of columns to be halved *recursively*. Part of the function may be mirrored to fill up extra places, without affecting the calculated dimension value.
- Allowing *fractions* of boxes vertically, we can perform integer operations throughout, with a single division at each scale. The error caused by this, works in the opposite direction to that caused by the rigidity of the grid origin and the blocks' form.
- The time origin may be varied and an average taken.

These considerations are illustrated in Figure 1, for the first three box scales on a 32 sample signal.

Pseudo code is given in Figure 2. The basic box counting algorithm is of quadratic complexity. Making the given assumptions, the complexity of the algorithm is reduced to become linear with signal length.

TH0482-0/92/0000.0043 $1.00

3 Applications

3.1 Dimension of fricative phonemes

Fig. 3 shows the result of applying the given algorithm to a speech phoneme.

Dimension histograms for fricative phonemes determined in this way are shown in Fig. 4. Comparison with results obtained on the same data by Hurst analysis [3] or morphological filtering [7, 8], shows that they are substantially the same. The computation times of our implementations suggest that a large (roughly 5 times) speedup can be obtained by box counting over the other two.

3.2 Automatic segmentation

Automatic segmentation of speech into periodic /noiselike segments can be done, using dimension values determined in a short sliding window. An example segmentation of a phrase is shown in Fig. 5. This simple division may be used as a basis for speech coding experiments based on an IFS approach [8].

4 Conclusion

The adaptation of the box counting algorithm has been applied with success to speech signals. The complexity of the calculation compares with that of zero crossing rate, and in some applications it may be advantagous to replace zero crossing rate with local fractal dimension. The algorithm is applicable to any continuous signal, and because of its speed, should be useful in other research areas than speech.

References

[1] Maragos, P & F-K Sun, "Measuring the fractal dimension of signals: Morphological covers and iterative optimization," Harvard Robotics Lab Technical Report no 91-14, October 1991, to appear in the IEEE Transactions on Signal Processing.

[2] Maragos, P, 'Fractal aspects of speech signals: dimension and interpolation," Proc. ICASSP '91, Toronto, May 1991, pp 417–420.

[3] Boshoff, HFV & M Grotepass, "The fractal dimension of fricative speech sounds," Comsig '91, Indaba Centre, Pretoria, August 1991, pp 12–16.

[4] Barnsley, MF, *Fractals everywhere*, Academic Press, San Diego, 1988.

[5] Strahle, WC, "Turbulent combustion data analysis using fractals," AIAA Journal, Vol 29 No 3, March 1991, pp 409–417.

[6] Mandelbrot, BB, "Self-affine fractals and fractal dimension," Physica Scripta, Vol 32, 1985, pp 257–260.

[7] Botha, TF, "Fraktale en spraak," Final year B Ing Project, University of Stellenbosch, 1991.

[8] Boshoff, HFV "Fractals applied to speech processing: Dimension of phonemes and coding of /a/ and /s/," Fractals in Engineering '92, École Polytechnique, Montréal, June 3–5 1992.

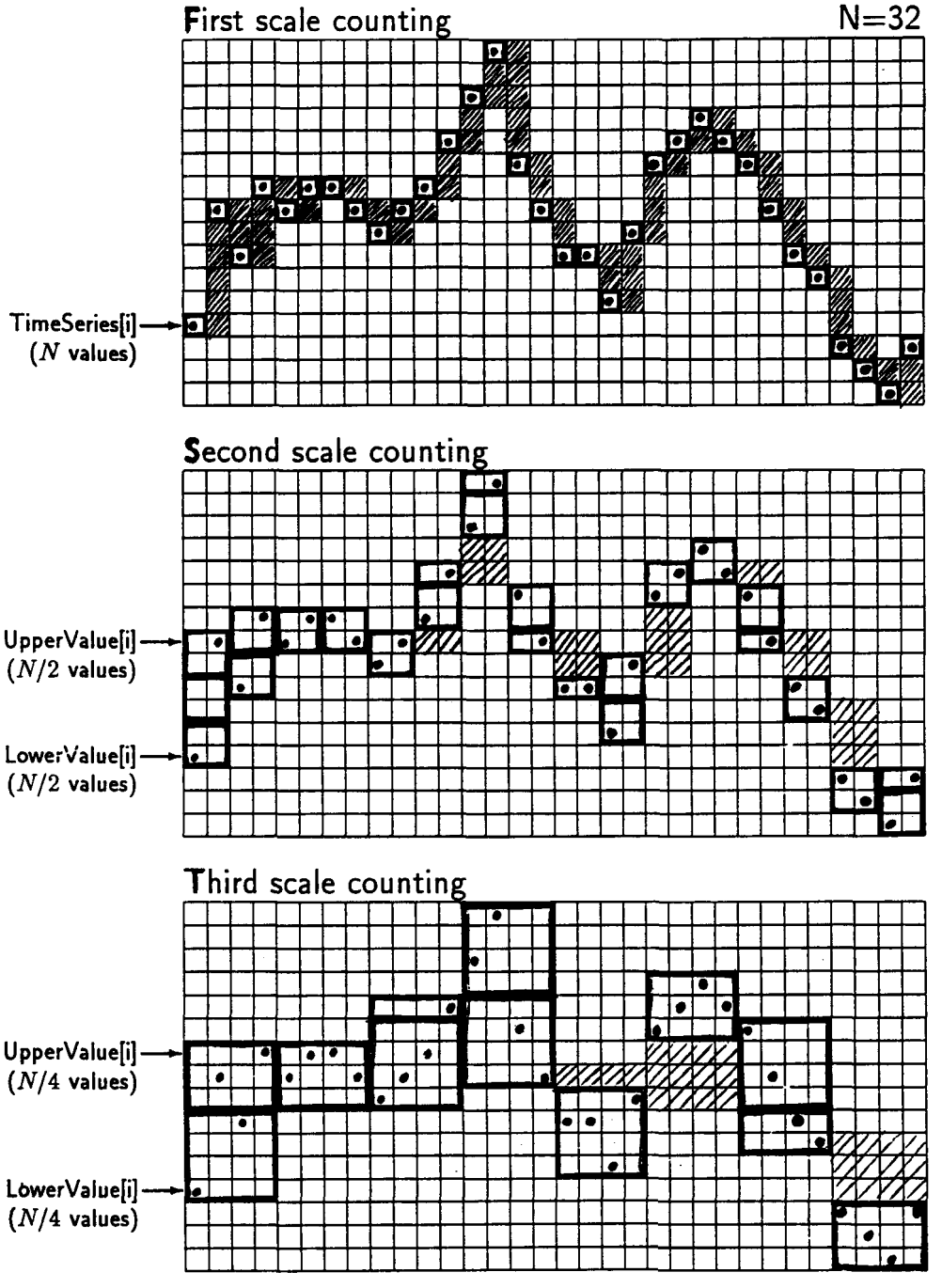

Figure 1: Adaptation of box counting: First three steps

```
BlockSize[1] := 1;                  {Sampling discretization }
NumBox[1] := 1;                     {One block in first column}
for i := 0 to N-2 do begin          {Count N-1 more columns}
  NumBox[1] := NumBox[1] + 1 +      {Point occupies 1 block}
    abs(Samples[i]-Samples[i+1]); {Function assumed continuous}
end;                                {Beware of integer overflow!}

NumCols := N/2;
for i := 0 to NumCols-1 do begin {Prepare for recursion}
  UpperValue[i] := max(Samples[2*i],Samples[2*i+1]);
  LowerValue[i] := min(Samples[2*i],Samples[2*i+1]);
end;
UpperValue[Numcols] := UpperValue[NumCols-1];
LowerValue[Numcols] := LowerValue[NumCols-1];
{Create bogus column for easier loop later}

for scale := 2 to MaxScale do begin
  Dummy := 0;
  BlockSize[scale] := 2*BlockSize[scale-1];
  for i := 0 to NumCols-1 do begin {Count inside NumCols columns}
    Dummy := Dummy + UpperValue[i] - LowerValue[i] + 1;
    {Add blocks only if no overlap exists between columns:
        (bogus column contributes nothing)}
    if UpperValue[i] < LowerValue[i+1] then
        Dummy := Dummy + LowerValue[i+1] - UpperValue[i];
    if LowerValue[i] > UpperValue[i+1] then
        Dummy := Dummy + LowerValue[i] - UpperValue[i+1];
  end; {for i}
  NumBox[scale] := round(Dummy/BlockSize(scale));
  {Convert from smallest to present vertical scale only once.
   Error arising works against that from rigidity of boxes.}

  if scale <> MaxScale do begin {Prepare for recursion}
    NumCols := NumCols/2;
    for i := 0 to NumCols-1 do begin
      UpperValue[i] := max(UpperValue[2*i],UpperValue[2*i+1]);
      LowerValue[i] := min(LowerValue[2*i],LowerValue[2*i+1]);
    end;
    UpperValue[Numcols] := UpperValue[NumCols-1];
    LowerValue[Numcols] := LowerValue[NumCols-1];
    {Create bogus column for easier loop later}
  end; {if}
end; {for scale}
```

Figure 2: Pseudo code for adapted box counting algorithm

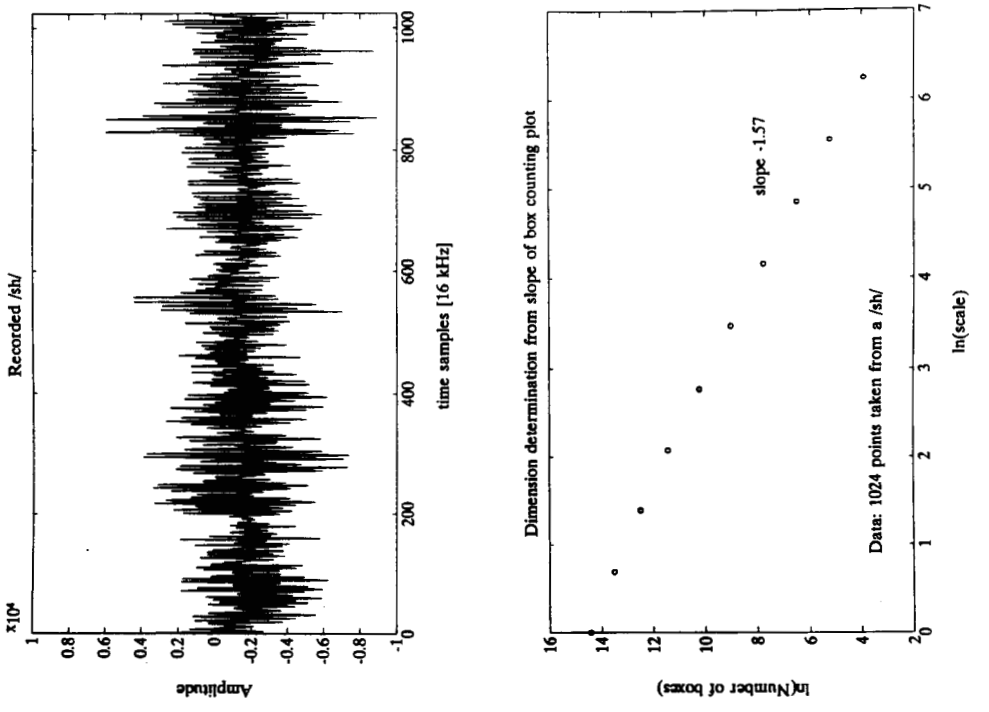

Figure 3: Dimension of a single phoneme

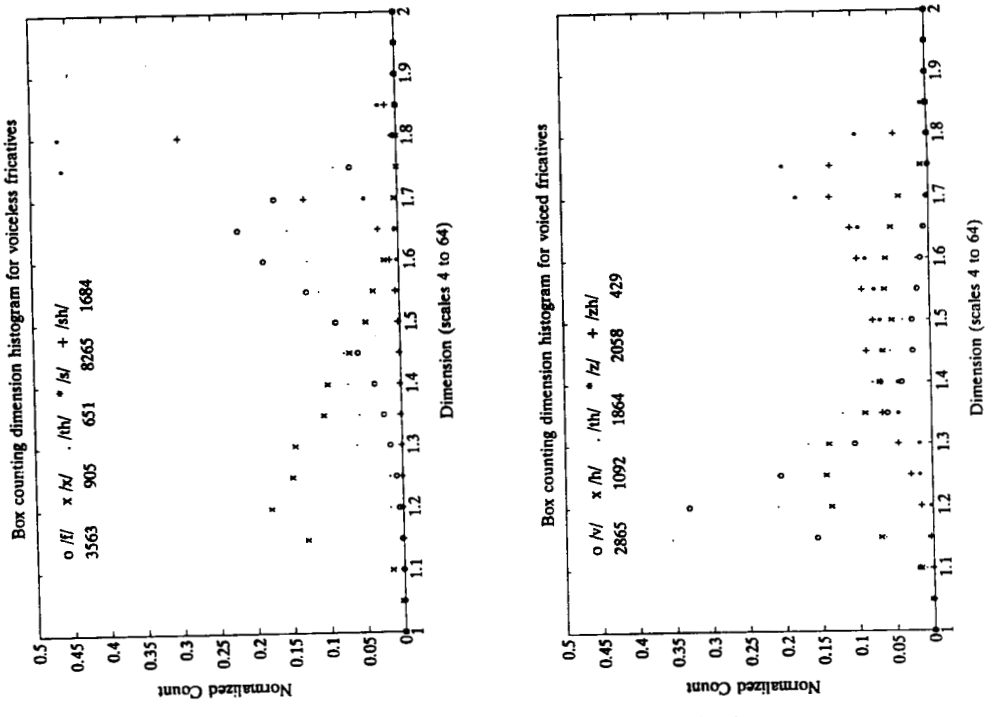

Figure 4: Histograms of box counting dimension of fricatives.

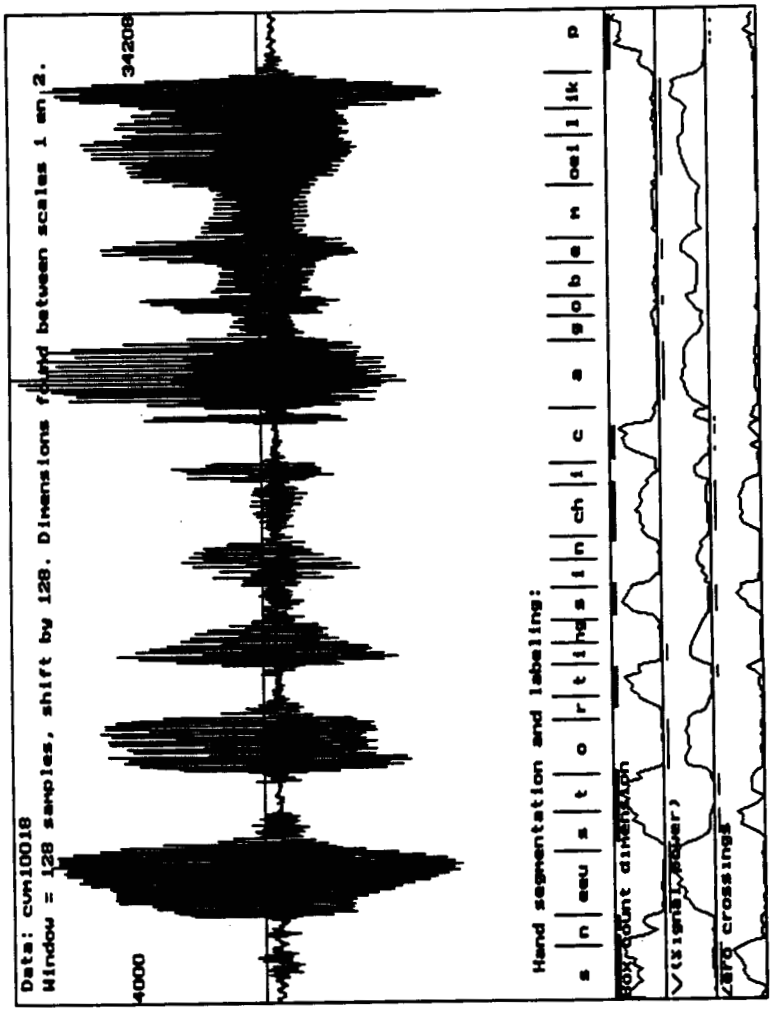

Figure 5: Speech phrase automatically segmented using box counting dimension. Vertical lines indicate hand segmenting and horisontal bars box counting dimension segments.