Simple Wavelets with Convolution

Daniel Beatty

TexasTech University

Lubbock,Texas

April 11,2003

Contents

```
$j$    $[0, N)$
    n=i-j
    if ($n$    $[0, M)$)
        $y_i$ + = $x$
```

L is the limit on the number resolutions that signal can have based on the wavelet type.

The first Ωersion is simple in concept, but provides a few more possibilities for error and confusion. Regardless of the case, the four components have the following definitions:

3.1.1 1-D to 2-D Method

Both rows 1-D and 2-D and columns 1-D and 2-D transform are performed similarly. The obvious di erence is the indexing of rows and columns.

Given 1-D wavelet transform

source matrix

Algorithm: (Row Transform)

```
$i$  $rows$
• $j$  $columns$
• $S[j] \leftarrow source[i][j]$
• $S$  $^{W}$ $R$
```

This principle of this algorithm is simple. Only three intuitive steps are

- n = k - l
- if ($n \quad columns$)

  $yA_k = W_{i,n} \quad hA_l$

  $yD_k = W_{i,n} \quad hD_l$

3. Transfer back to W

   $W_j = yA|yD$

Note: W_i names the row vectors and W_j names the column vectors, and $W_{i,j}$ is the element from the ith row and jth column.

3.3 Computational Cost:

The cost of this algorithm is computed first for each row and each column. This value is used to compute the cost of the matrix. The cost of computing the matrix is used to compute the cost of the multi-resolution steps. Per row the cost is 3k, where k is the number of columns. Per column the cost is 3l, where l is the number of rows. For the whole matrix, one resolution costs 6kl operations to compute the wavelet transform. Per resolution, the rows and columns shrink by 2^i for each resolution, i, performed. The limit of this cost equals 12*fo*

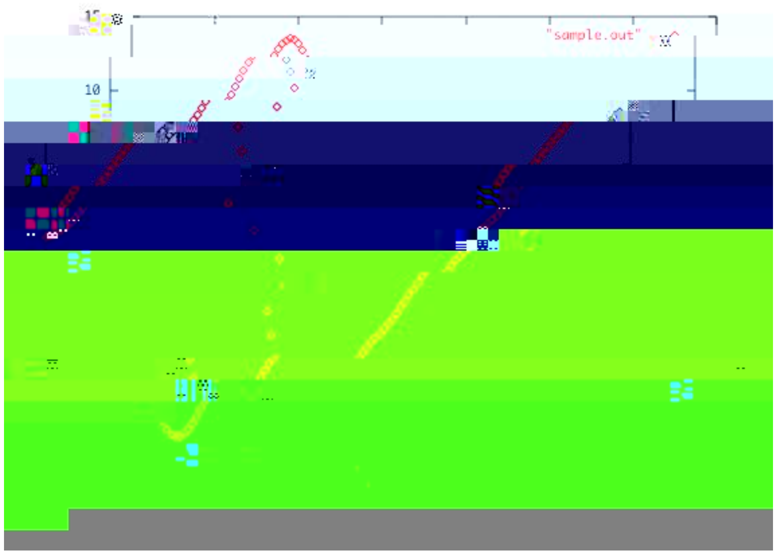


Figure 2: Recovered function. The x-axis is the array index (index n). The y value is simply the value y[n]. The function was recovered from an even indexed wavelet transform.

5 Results: 2-D Wavelet Transform

A simple (om)-om picture shomws the di erence correct indexing pr-duces in the

wavelet transform and its271(7n)1(dex8ctur)ee ure Thtormdeth(simple)-drmcture 27(v)1(e)erm i th

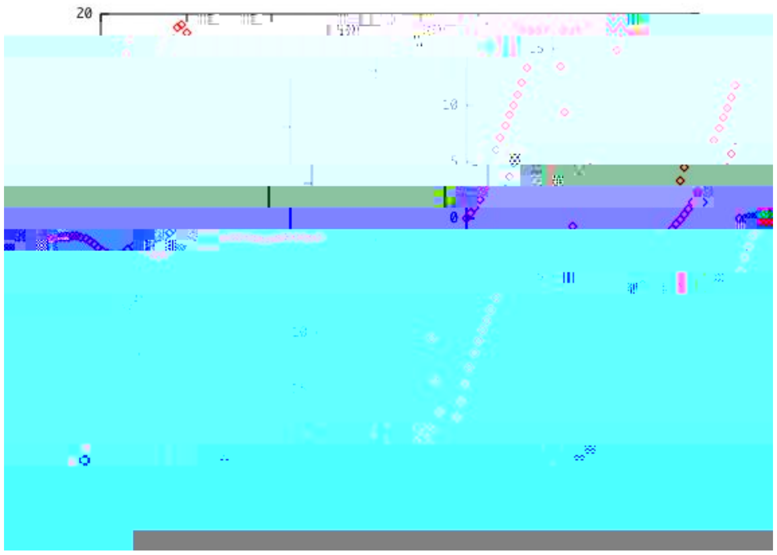


Figure 3: Recovered function. The x-axis is the array index (index n). The y

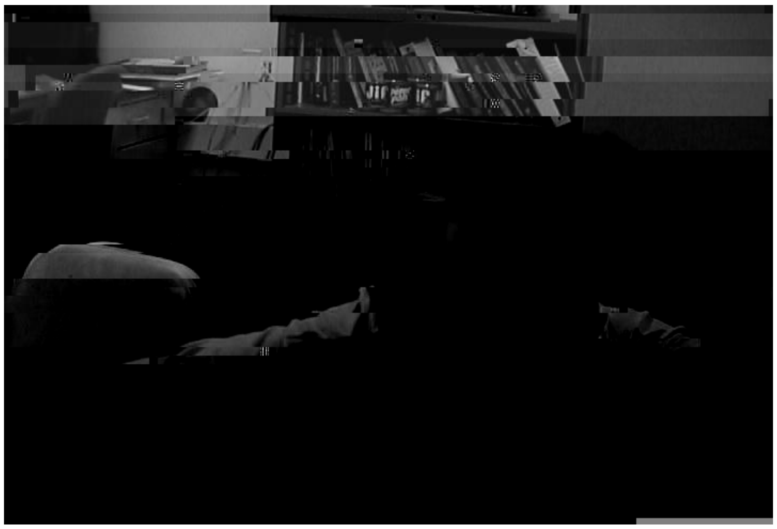


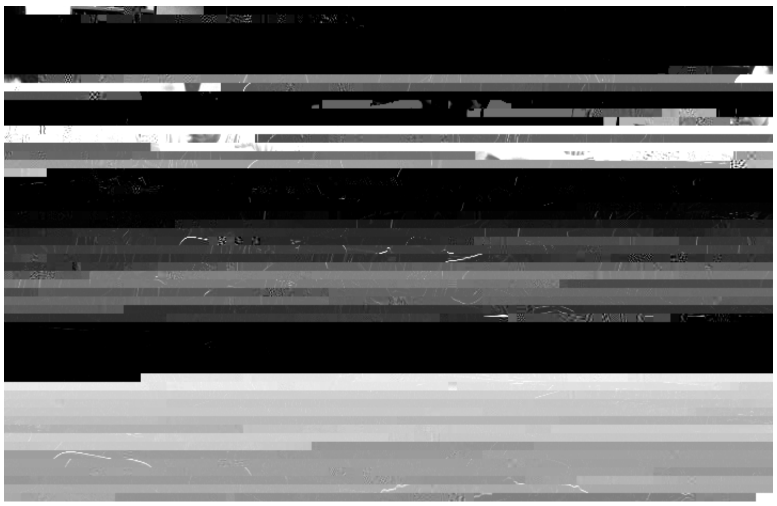
Figure 4: Original Image. This image is the original image.

indexed (Figure 6). As a result of an error in indexing, ringing is seen on edges in this method(Figure 8) for a case in point. Caution is incredibly important when matching both forward and reverse (ing, sincehinge

to the actual ordering can be obscure and tricky.

A correct result is shown in Figure ??. In this case,e indexing was matched up and ringing is not presen It is clear that the recovered image and the original

Figure 7: Wavelet Transform Image. This image is divided in to average, horizontal, vertical and diagonal components, using the vector-matrix version.



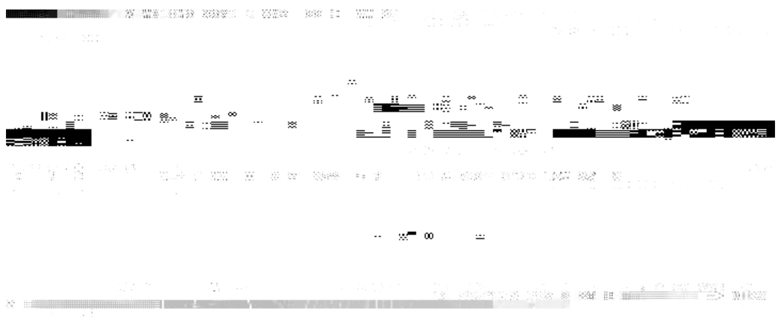


Figure 10: Recovered Image - Wrong Order (Multi-Resolution). This image shows a 2-D wavelet transform after it was recovered out of order. Obviously, the distortion is hideous.

Figure 11: Recovered Image - 2% threshold (Multi-Resolution). This image had

Figure 14: Recovered Image - 1% threshold (Multi-Resolution). This image had

function.   The algorithm is as follows

```
$S$ = $input$
$l$ = is the size of S (or the input)
  $i$    [0, $l$)
      $R_A$ = $S$    $H_A$
      $R_D$ = $S$    $H_D$
      T = join ($R_A$,$R_D$)
      F = evenSplit (T)
```

Output:

```
    myVector Result
```

Algorithm

```
    result.deallocate
    $s_l = size(left)$
    $s_r = size(right)$
    $s = s_l$
```

A.4.3 Algorithm:

```
$l_1 = s.size$
    $l_2 = r.size$
    if ($l_1 < l_2$) and (start ¡ end)
        $i \quad [start, end]$
            $r_i = s_i$
```

A.5 Procedure: Extract

Purpose: To produce a new or replace a myVector whose length is that of the

A.5.2 Output:

The output of the wavelet Xform ¿ extraction procedure is simply:

```
myVector R
```

$$R_{2\,(l/2)-1} = (A_{l/2} - D_{l/2})^{\square\,\underline{\quad\quad}}$$

that the objective of this class is not produce image translators for each image type imaginable.

B.1 Method: Column Wavelet Transform

B.2 Method: Row Wavelet Transform

Like the column wavelet transform, the row wavelet transform takes a source

matrix and returns a result399(pa)-317(m)-1a(tr)1ix. Tthelyawfic(an)28tan10(c)-1ens rle

sthei99(tem)-1msat399(pa)-35(fo)1pratedanrosa399(ota)-341(c)-1(o99(umns).)-752(Second)-1(,)-318tthe)-34(la

the nnde

ac

Just of note, this class lacks for the moment an inverse transform function. Not that such a thing does not exist mathematically. It simply was not implemented at the time of this document.

B.5 Method: Row Wavelet Inverse Transform

The row wavelet inverse transform uses the one-dimensional form to transform each row of the matrix. The algorithm is as follows:

1. *i* [0, *kiψfψuncψψa As aixd atomluese timeψdThhihinixthatdeψdTψSeFTFψfψdiTFψ/iψdTJψv*

!

B.9 Method: Self Column Inverse Wavelet Transform

The code name for this method is selfColumnInverseXform, and it takes three arguments. This particular method performs a column inverse wavelet transform on a particular column, designated j. The return value is placed in R, in the correct column.

B.9.1 Given

Two references are given. One to the source matrix, and the other to the result matrix. The third item is an integer, j. This integer identifies the column to be transformed.

B.9.2 Notation

Two symbols are used to simplify the writing of the algorithm.

- k is number of rows erofita1(y)1□dsrem.

□ k y t of thsre symoth-28(er)-323(of)-332(v)330□□d[ng of therre hm.

k□ y tpxkm.

•
k□ the resma1(ormed.)]TJ/F30-□2.9607.963□f0-□37.211□d[3B.9.t (or)1hmm.

←F∞ ←∞ ←T←F∞ ∞T|←T JF←∞∞∞ ∮ ∞ ∞