CS 6810/7810: Wavelets and Wavelet Algorithms
Assignment 6
2D Ordered HWT

Vladimir Kulyukin
Department of Computer Science
Utah State University

February 18, 2017

Learning Objectives

1. 2D Ordered HWT
2. Applying 2D Ordered HWT to Images

Introduction

In this assignment, you will implement the 2D Ordered HWT and apply it to images. For the sake of simplicity, we will assume that input images are square and their length and width are integral powers of 2.

Problem 1

Implement a class `TwoDHWT.java` with the following static method that applies the 2D Ordered HWT to the 2D signal `sig` for the number of iterations given in the second argument.

```
public static void ordFwdDWTForNumIters(double[][] sig, int num_iters) {}
```

The class `WaveletAlgos_S17_HW06.java` has several 2D signals defined. You can use these signals for debugging. We analyzed some of these signals in Lecture 8. There is a method `testOrd2DHWT(double[][] data, int num_iters)` in `WaveletAlgos_S17_HW06` to test different numbers of iterations. Note that the method `TwoDHWT.ordFwdDWTForNumIters` has a third boolean argument set to `false`. You do not have to have this argument. In my implementation, this is a debugging flag. When I want intermediate states displayed, I set this argument to `true`.

```
  public class WaveletAlgos_S17_HW06 {
     static double[][] signal_2x2_1 = {
        {9, 7},
        {5, 3}
     };

     static double[][] signal_4x4_1 = {
        {9,7,6,2},
        {5,3,4,4},
        {8,2,4,0},
        {6,0,2,2}
     };

     static double[][] signal_8x8_2 = {
        {8,  5,  4,  8,  6, 8, 10,  8},
        {8, 10, 10,  4, 10, 4,  8,  2},
        {6, 10,  2,  4,  2, 6,  1,  6},
        {0,  8,  0, 10, 10, 6,  6, 10},
        {8,  8,  8,  0,  4, 8,  6,  2},
        {10, 6,  2,  2,  6, 6,  6,  8},
        {2,  4, 10, 10, 10, 4,  6, 10},
        {10, 6, 10,  6,  6, 4,  4,  4}
    };
    public static void testOrd2DHWT(double[][] data, int num_iters) {
        final int dim = data.length;
        TwoDHWT.ordFwdDWTForNumIters(data, num_iters, false);
        System.out.println("Result Matrix");
        Utils.display2DArray(data, dim, dim);
        System.out.println();
    }
  }
```

Let us run a few tests. In the first test, we appy the 2D HWT to `signal_2x2_1`.

```
    public static void main(String[] args) {
       testOrd2DHWT(signal_2x2_1, 1);
    }
```

Here is the output.

```
Result Matrix
6.0 1.0
2.0 0.0
```

In the second test, we apply the HWT to `signal_4x4_1` for one iteration.

```
public static void main(String[] args) {
   testOrd2DHWT(signal_4x4_1, 1);
}
```

Here is the output.

```
Result Matrix
6.0 4.0 1.0 1.0
4.0 2.0 3.0 1.0
2.0 0.0 0.0 1.0
1.0 0.0 0.0 1.0
```

We now run 2 iterations of 2D HWT on the same signal.

```
public static void main(String[] args) {
   testOrd2DHWT(signal_4x4_1, 2);
}
```

Here is the output.

```
Result Matrix
4.0 1.0 1.0 1.0
1.0 0.0 3.0 1.0
2.0 0.0 0.0 1.0
1.0 0.0 0.0 1.0
```

In the third test, let us apply the 2D HWT to the `8x8` signal analyzed in lecture 8 defined as `signal_8x8_2`.

```
public static void main(String[] args) {
   testOrd2DHWT(signal_8x8_2, 1);
}
```

Here is the output after one scale.

```
Result Matrix
7.75 6.5 7.0 7.0 0.25 0.5 1.0 2.0
6.0 4.0 6.0 5.75 -3.0 -3.0 0.0 -2.25
8.0 3.0 6.0 5.5 1.0 2.0 -1.0 0.5
5.5 9.0 6.0 6.0 0.5 1.0 2.0 -1.0
-1.25 -0.5 0.0 2.0 1.25 -2.5 -2.0 -1.0
2.0 -1.0 -2.0 -2.25 1.0 2.0 -2.0 -0.25
0.0 1.0 0.0 -1.5 -1.0 2.0 -1.0 1.5
-2.5 1.0 1.0 2.0 -1.5 -1.0 1.0 -1.0
```

We now apply the 2D Ordered HWT to the signal for two iterations.

```
public static void main(String[] args) {
   testOrd2DHWT(signal_8x8_2, 2);
}
```

Here is the output.

```
Result Matrix
6.0625 6.4375 0.8125 0.0625 0.25 0.5 1.0 2.0
6.375 5.875 0.375 0.125 -3.0 -3.0 0.0 -2.25
1.0625 0.5625 -0.1875 -0.0625 1.0 2.0 -1.0 0.5
-0.875 -0.125 2.125 0.125 0.5 1.0 2.0 -1.0
-1.25 -0.5 0.0 2.0 1.25 -2.5 -2.0 -1.0
2.0 -1.0 -2.0 -2.25 1.0 2.0 -2.0 -0.25
0.0 1.0 0.0 -1.5 -1.0 2.0 -1.0 1.5
-2.5 1.0 1.0 2.0 -1.5 -1.0 1.0 -1.0
```

Finally, we apply the 2D Ordered HWT to the signal for three iterations.

```
public static void main(String[] args) {
   testOrd2DHWT(signal_8x8_2, 3);
}
```

Here is the output.

```
Result Matrix
6.1875 0.03125 0.8125 0.0625 0.25 0.5 1.0 2.0
0.0625 -0.21875 0.375 0.125 -3.0 -3.0 0.0 -2.25
1.0625 0.5625 -0.1875 -0.0625 1.0 2.0 -1.0 0.5
-0.875 -0.125 2.125 0.125 0.5 1.0 2.0 -1.0
-1.25 -0.5 0.0 2.0 1.25 -2.5 -2.0 -1.0
2.0 -1.0 -2.0 -2.25 1.0 2.0 -2.0 -0.25
0.0 1.0 0.0 -1.5 -1.0 2.0 -1.0 1.5
-2.5 1.0 1.0 2.0 -1.5 -1.0 1.0 -1.0
```

Problem 2

Implement the method in `WaveletAlgos_S17_HW06.java`:

`pubic static void createScaledImageOf2DHWT(input_img_path, output_img_path, num_iters)`

This method applies the specified number of iterations of the 2D Ordered HWT to the image in `input_img_path` and saves the result in the image in `output_img_path`. Consider the image in figure 1.

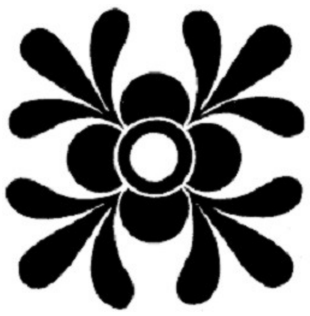

Figure 1: Ornament 1

Then the following `main()` generates the images displayed in figures 2, 3, 4, and 5.

```
public static main(String[] args) {
  createScaledImageOf2DHWT("ornament_01.jpg", "ornament_01_1_scale.jpg", 1);
  createScaledImageOf2DHWT("ornament_01.jpg", "ornament_01_2_scales.jpg", 2);
  createScaledImageOf2DHWT("ornament_01.jpg", "ornament_01_3_scales.jpg", 3);
  createScaledImageOf2DHWT("ornament_01.jpg", "ornament_01_4_scales.jpg", 4);
}
```

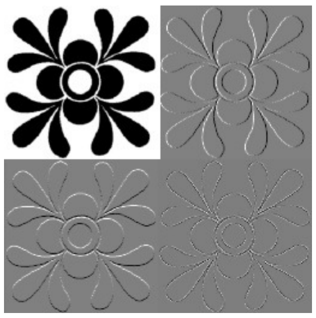

Figure 2: Ornament 1: 1 scale of 2D HWT saved in ornament_01_1_scale.jpg

The top left square matrix in figure 2 is the coarser (or downsampled) representation of the original signal after 1 iteration. The top right square matrix in each image is the horizontal wavelets. The bottom left matrix is the vertical wavelets. The bottom right matrix is the horizontal wavelets.
If 2 iterations/scales are applied to the image in figure 1, the 2nd scale is applied to the top left matrix. This top left matrix is replaced by four smaller matrices, as shown in figure 3, where the top left matrix is the coarser image, the top right matrix contains the horizontal wavelets after the 2nd scale, the bottom left matrix contains the vertical wavelets after the 2nd scale, and the bottom right matrix contains the diagonal wavelets after the 2nd scale.

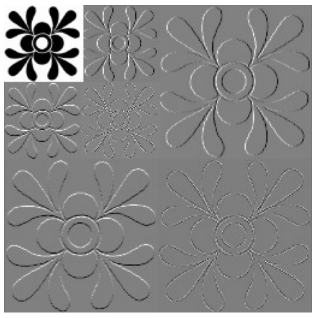

Figure 3: Ornament 1: 2 scales of 2D HWT saved ornament_01_2_scales.jpg

The same recursive pattern continues in figures 4 and 5.

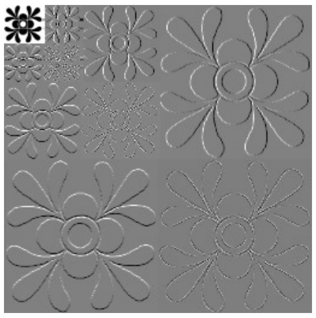

Figure 4: Ornament 1: 3 scales of 2D HWT saved in ornament_01_3_scales.jpg

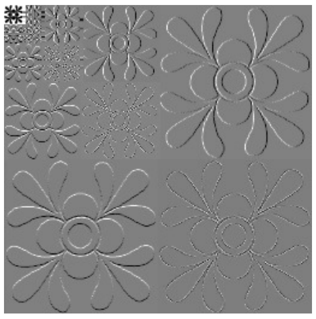

Figure 5: Ornament 1: 4 scales of 2D HWT in ornament_01_4_scales.jpg

Displaying the top left images is straightforward in that they are smaller versions of the original image. All we need to do is to convert the doubles into integers. Displaying wavelets is a bit trickier in that some of them will be negative and no image format, to the best of my knowledge, can handle negative real pixel values. What I did is in the above figures is to scale the wavelets to lie in `[0, 255]`.

What To Submit

Submit your `WaveletAlgos_S17_HW06.java` and `TwoDHWT.java` or their Py equivalents via Canvas.

Happy Hacking!