Algorithms

Coursework 2: Wavelet Image Compression

Image and signal compression is one of the most important applications of wavelets. In this assessed exercise, you are asked to implement a Haar wavelet image compression algorithm. Before starting this exercise make sure that you have read the lecture notes on compression and the tutorial notes on wavelets. Also, on CATE you will find a review article from Eric J. Stollnitz, Tony D. DeRose and David H. Salesin, discussing wavelet transforms in two dimensions and their application to image compression.

You are asked to complete the skeleton class HaarWavelet with the following functionalities:

1. Wavelet decomposition of images (a.k.a. the wavelet transform)

2. Compression of the wavelet coefficients by discarding small coefficients

3. Reconstruction (inverse wavelet transform) of images from the compressed wavelet representation.

Please note that this exercise should be done in the labs using the Linux operating system.

Provided Files

1. Images:

    a. queenstower.pgm
    b. jobs.pgm

2. haar.cpp

    Main function. Providing a program that takes an image, apply haar wavelet compression with given tolerance rate between 0 and 1 and write the result image to the file. Compile with the following command:

    ```
    g++ haar.cpp HaarWavelet.cpp Image.cpp –I . –o haar
    ```

    Run `./haar` without argument to get the help message.

3. Image.h and Image.cpp

    The class Image inherits vector<double>, can be used as a vector of double if needed.

    List of Methods:

    ```
    // init/reset the image size.
    void init (int rows, int cols)

    // load from a PGM file
    int load(string fileName)

    // save to a PGM file
    int save(string fileName)

    // return the specified row
    vector <double> extractRow(int colNum) const

    // return the specified column
    ```

```
vector <double> extractCol(int colNum) const

// get number of rows
unsigned int numRows() const

// get number of columns
unsigned int numCols() const

// get the specified value at (row, column)
double getData(unsigned int row, unsigned int col)
const

//set value at (row, column) to val
void setData(unsigned int row, unsigned int col, double val)
```

4. HaarWavelet.h and HaarWavelet.cpp

   The class HaarWavelet inherits Image. The Haar wavelet coefficients should be stored in the data field inherited from Image. It implements the following member functions:

   a. A constructor which takes an image and a value for decomposition type and computes the wavelet transformation for the image. Valid values of decomposition types are `HaarWavelet::STANDARD` and `HaarWavelet::NONSTANDARD`. The field `m_type` stores the type of decomposition. It is set to `HaarWavelet::NONE` when it is empty.

   b. ForwardTransform(Image &out): This computes the wavelet coefficients from the image.

   c. ReverseTransform(Image &out): This reconstructs the image from the wavelet coefficients.

   d. Helper function sortIndex, that produces a vector of indices for the data, so that abs(data[index[i]]) is ranked from large to small.

TODO:

Your task is to complete the missing functions in HaarWavelet.h and HaarWavelet.cpp:

1. standardDecomposition (Image& out)
   // standard 2D decomposition

2. standardReconstruction (Image& out)
   // standard 2D reconstruction

   Tip: to access the i-th element, using (*this)[i], eg:

   ```
   (*this)[i] = 100;
   double x = (*this)[i];
   ```

3. void discardDetail(double tol);
   // Discard coefficients representing small detail.

   The parameter `tol` specifies the tolerance rate for discarding image details . This value should be between 0(discard nothing) and 1(discard all).

   Tip: the following algorithm can be used

   a. Sum the absolute value of coefficients, times tol to get the threshold
   b. Sort the absolute value of coefficients from large to small.

c. Start from the coefficient with the small absolute value, discarding them by setting them to zero, until the absolute value of discarded coefficient sum up to the threshold.

4. nonStandardDecomposition (Image& out)
   // non-standard 2D decomposition

5. nonStandardReconstruction (Image& out)
   // non-standard 2D reconstruction

Experiments

After completing the function, you have to compile the project and use the program to generate compressed images. For the each image given to you, you have to generate six different compressed versions combining standard and nonstandard transformation with tolerance rates 0.3, 0.5 and 0.7.

Hand-in List

Source code:

- HaarWavelet.h
- HaarWavelet.cpp

Transformed and reconstructed images

- queenstowerStandard3.pgm
- queenstowerStandard5.pgm
- queenstowerStandard7.pgm

- queenstowerNonStandard3.pgm
- queenstowerNonStandard5.pgm
- queenstowerNonStandard7.pgm

- jobsStandard3.pgm
- jobsStandard5.pgm
- jobsStandard7.pgm

- jobsNonStandard3.pgm
- jobsNonStandard5.pgm
- jobsNonStandard7.pgm