Report

Image Compression using Wavelets

Under the guidance of –

Prof. G.R.M REDDY

Made By-

Abhinav Prakash 11IT02

Shivam Singh 11IT87

Vaibhav Tripathi 11IT99

**Contents**

* Introduction
* Lossy Compression
* Wavelets for Image Compression
* A haar wavelet transform
* Compression of 2D image with haar wavelet technique
* Matlab code
* Conclusion

**Introduction**

The computer is becoming more and more powerful day by day. As a result, the use of digital images is increasing rapidly. Along with this increasing use of digital images comes the serious issue of storing and transferring the huge volume of data representing the images because the uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth.

The image is actually a kind of redundant data i.e. it contains the same information from certain perspective of view. By using data compression techniques, it is possible to remove some of the redundant information contained in images. Image compression minimizes the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a certain amount of disk or memory space. It also reduces the time necessary for images to be sent over the Internet or downloaded from web pages.

Image compression is the process of reducing or compressing size of image files but still retaining important information. Compressed file makes it easy to handle and circulate on web.

## Lossy Compression:

In order to achieve higher rates of compression, we give up complete reconstruction and consider lossy compression techniques. The reason is that our eyes and ears cannot distinguish subtle changes. The compressed data is not the same as the original data, but a close approximation of it. Yields a much higher compression ratio than that of lossless compression.

In lossless compression schemes, the reconstructed image, after compression, is numerically identical to the original image. However lossless compression can only achieve a modest amount of compression. An image reconstructed following lossy compression contains degradation relative to the original .Often this is because the compression scheme completely discards redundant information. However, lossy schemes are capable of achieving much higher compression.

**Wavelets for image compression**

Wavelet transform exploits both the spatial and frequency correlation of data by dilations (or contractions) and translations of mother wavelet on the input data. It supports the multi resolution analysis of data i.e. it can be applied to different scales according to the details required, which allows progressive transmission and zooming of the image without the need of extra storage. Another encouraging feature of wavelet transform is its symmetric nature that is both the forward and the inverse transform has the same complexity, building fast compression and decompression routines. Its characteristics well suited for image compression include the ability to take into account of Human Visual System’s (HVS)characteristics, very good energy compaction capabilities, robustness under transmission, high compression ratio etc. The implementation of wavelet compression scheme is very similar to that of sub band coding scheme: the signal is decomposed using filter banks. The output of the filter banks is down-sampled, quantized, and encoded. The decoder decodes the coded representation, up-samples and recomposes the signal. Wavelet transform divides the information of an image into approximation and detail sub signals. The approximation sub signal shows the general trend of pixel values and other three detail sub signals show the vertical, horizontal and diagonal details or changes in the images. If these details are very small (threshold) then they can be set to zero without significantly changing the image. The greater the number of zeros the greater the compression ratio. If the energy retained(amount of information retained by an image after compression and decompression) is 100% then the compression is lossless as the image can be reconstructed exactly. This occurs when the threshold value is set to zero, meaning that the details have not been changed. If any value is changed then energy will be lost and thus lossy compression occurs. As more zeros are obtained, more energy is lost. Therefore, a balance between the two needs to be found out.

**A. Haar Wavelet Transform**

To understand how wavelets work, let us start with a simple example. Assume we have a 1D image with a resolution of four pixels, having values [9 7 3 5]. Haar wavelet basis can be used to represent this image by computing a wavelet transform. To do this, first the average the pixels together, pairwise, is calculated to get the new lower resolution image with pixel values [8 4]. Clearly, some information is lost in this averaging process. We need to store some detail coefficients to recover the original four pixel values from the two averaged values. In our example, 1 is chosen for the first detail coefficient, since the average computed is 1 less than 9 and 1 more than 7. This single number is used to recover the first two pixels of our original four-pixel image. Similarly, the second detail coefficient is -1, since 4 + (-1) = 3 and 4 - (-1) =5. Thus, the original image is decomposed into a lower resolution (two-pixel) version and a pair of detail coefficients.

Repeating this process recursively on the averages gives the full decomposition shown in Table I:

|  |  |  |
| --- | --- | --- |
| **Resolution** | **Average** | **Detail Coefficient** |
| 4 | [9,7,3,5] |  |
| 2 | [8,4] | [1,-1] |
| 1 | [6] | [2] |

Thus, for the one-dimensional Haar basis, the wavelet transform of the original four-pixel image is given by [6 2 1 -1]. We call the way used to compute the wavelet transform by recursively averaging and differencing coefficients, filter bank. We can reconstruct the image to any resolution by recursively adding and subtracting the detail coefficients from the lower resolution versions.

**B.Compression of 2D image with Haar Wavelet Technique**

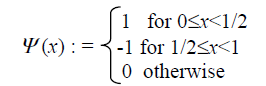
It has been shown in previous section how one dimensional image can be treated as sequences of coefficients. Alternatively, we can think of images as piecewise constant functions on the half-open interval [0, 1). To do so, the concept of a vector space is used. A one-pixel image is just a function that is constant over the entire interval [0, 1). Let V0be the vector space of all these functions. A two pixel image has two constant pieces over the intervals [0, 1/2) and [1/2, 1).We call the space containing all these functions V1. If we continue in this manner, the space Vj will include all piecewise-constant functions defined on the interval [0, 1) with constant pieces over each of 2j equal subintervals. We can now think of every one-dimensional image with 2j pixels as an element, or vector, in Vj. Note that because these vectors are all functions defined on the unit interval, every vector in Vj is also contained in Vj+1. For example, we can always describe a piecewise constant function with two intervals as a piecewise-constant function with four intervals, with each interval in the first function corresponding to a pair of intervals in the second. Thus, the spaces Vj are nested; that is, V 0⊂ V 1⊂ V 2⊂ …… This nested set of spaces Vj is a necessary ingredient for the mathematical theory of multi resolution analysis [6]. It guarantees that every member of V0 can be represented exactly as a member of higher resolution space V1.The converse, however, is not true: not every function G(x) inV1 can be represented exactly in lower resolution space V0; in general there is some lost detail [11].Now we define a basis for each vector space V j. The basis functions for the spaces V j are called scaling functions, and are usually denoted by the symbol φ. A simple basis for Vj is given by the set of scaled and translated box functions ;

φij (x) : = φ (2jx – i) i = 0, 1, 2…..2j -1 where

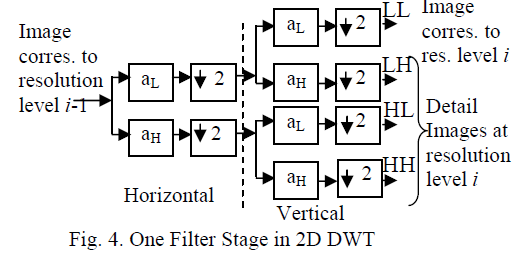


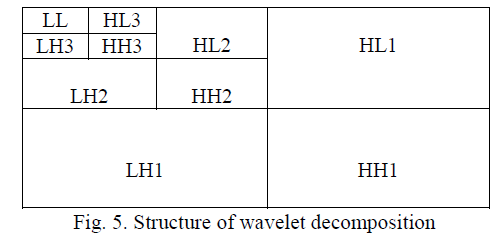
The wavelets corresponding to the box basis are known as the Haar wavelets, given by-





Thus, the DWT for an image as a 2D signal will be obtained from 1D DWT. We get the scaling function and wavelet function for 2D by multiplying two 1D functions. The scaling function is obtained by multiplying two 1D scaling functions :φ(x,y)=φ(x)φ(y). The wavelet functions are obtained by multiplying two wavelet functions or wavelet and scaling function for 1D. For the 2D case, there exist three wavelet functions that scan details in horizontal Ψ(1)(x,y)= φ(x)Ψ(y),vertical Ψ(2)(x,y)= Ψ(x)φ(y) and diagonal directions: Ψ(3)(x,y)=Ψ(x) Ψ(y). This may be represented as a four channel perfect re construction filter bank as shown in Fig. 4. Now, each filter is 2D with the subscript indicating the type of filter (HPF or LPF) for separable horizontal and vertical components. By using these filters in one stage, an image is decomposed into four bands. There exist three types of detail images for each resolution: horizontal (HL), vertical (LH), and diagonal (HH).The operations can be repeated on the low low (LL) band using the second stage of identical filter bank. Thus, a typical2D DWT, used in image compression, generates the hierarchical structure shown below.

****



The transformation of the 2D image is a 2D generalization of the 1D wavelet transformed already discussed. It applies the1D wavelet transform to each row of pixel values. This operation provides us an average value along with detail coefficients for each row. Next, these transformed rows are treated as if they were themselves an image and apply the 1Dtransform to each column. The resulting values are all detail coefficients except a single overall average co-efficient. In order to complete the transformation, this process is repeated recursively only on the quadrant containing averages. Now let us see how the 2D Haar wavelet transformation is performed. The image is comprised of pixels represented by numbers [12]. Consider the 8×8 image taken from a specific portion of a typical image shown in Fig. 6. The matrix (a 2Darray) representing this image is shown in Fig. 7.

Now we perform the operation of averaging anddifferencing to arrive at a new matrix representing the sameimage in a more concise manner. Let us look how theoperation is done. Consider the first row of the Fig. 7.

Averaging:

(64+2)/2=33, (3+61)/2=32, (60+6)/2=33,

(7+57)/2=32

Differencing:

64–33 =31, 3–32= –29, 60–33=27 and

7–32= –25

****

Fig. 6: A 8×8 image

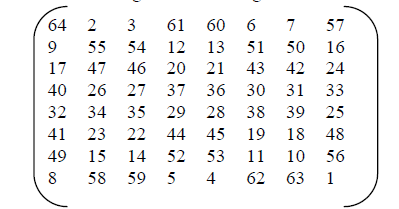


Fig. 7 : 2D array representing the Fig. 6

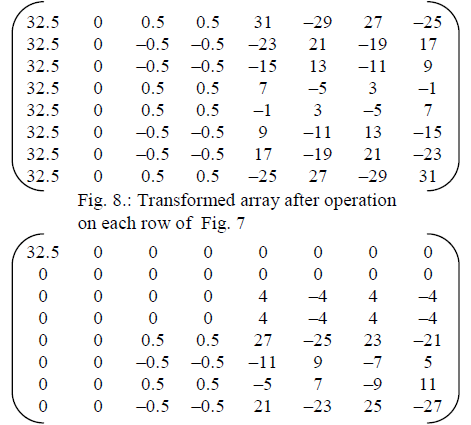
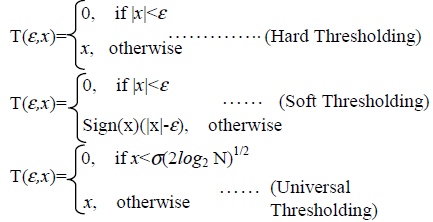


Fig. 9.: Final Transformed Matrix after one step

It can be seen that in the final transformed matrix, we find a lot of entries zero. From this transformed matrix, the original matrix can be easily calculated just by the reverse operation of averaging and differencing i.e. the original image can be reconstructed from the transformed image without the loss of information. Thus, it yields a lossless compression of the image. However, to achieve more degree of compression, we have to think of the lossy compression. In this case, a nonnegative threshold value say ε is set. Then any detail coefficient in the transformed data whose magnitude is less than or equal to ε is set to zero. It will increase the number of0’s in the transformed matrix and thus the level of compression is increased. So, ε =0 is used for a lossless compression. If the lossy compression is used, the approximations of the original image can be built up. The setting of the threshold value is very important as there is at rade off between the value of ε and the quality of the compressed image. The different thresholding methods we have used are: hard thresholding, soft thresholding and universal thresholding. These thresholding methods are defined as follows:



where σ is the standard deviation of the wavelet coefficients and N is the number of wavelet coefficients. Loosely saying, the compression ratio of the image is calculated by- the number of nonzero elements in original matrix : the number of nonzero elements in updated transformed matrix [13].

In summary, the main steps of the 2D image compression using Haar Wavelet as the basis functions are: (a) Start with the matrix P representing the original image, (b) Compute the transformed matrix T by the operation averaging and differencing (First for each row, then for each column) (c)Choose a threshold method and apply that to find the new matrix say D (e) Use D to compute the compression ratio and others values and to reconstruct the original image as well. Now we see the effect of one step averaging and differencing of an image. The Fig. 11 (a) is the original image and the Fig. 11 (b) is the transformed image after applying the one step averaging and differencing. The more steps produce more decomposition.

**Code For Row And Column Transformation**

/\*row transformation\*/

for(i=0;i<row;i++){w=col;

do{ k=0;

/\*averaging\*/ for(j=0;j<w/2;j++)

a[j]=((mat[i][j+j]+mat[i][j+j+1])/2);

/\*differencing\*/ for(j=w/2;j<w;j++,k++)

a[j]=mat[i][j-w/2+k]-a[k];

for(j=0;j<row;j++) mat[i][j]=a[j];

w=w/2;

}while(w!=1);

}

/\*column transformation\*/

for(i=0;i<col;i++){ w=row;

do{k=0;

/\*averaging\*/ for(j=0;j<w/2;j++)

a[j]=((mat[j+j][i]+mat[j+j+1][i])/2);

/\*differencing\*/for(j=w/2;j<w;j++,k++)

a[j]=mat[j-w/2+k][i]-a[k];

for(j=0;j<w;j++) mat[j][i]=a[j];

w=w/2;

}while(w!=1);

}



1-level decomposition



2-level decomposition

**Matlab code :**

function varargout = imageprocessing(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @imageprocessing\_OpeningFcn, ...

'gui\_OutputFcn', @imageprocessing\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function imageprocessing\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = imageprocessing\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function pushbutton1\_Callback(hObject, eventdata, handles)

global X;

global xd;

global rc;

siz = size(X);

%to convert RGB image into gray scale

if(siz(3) == 3)

X = rgb2gray(X);

end;

% inputting the decomposition level and name of the wavelet

n=4;

wname = 'haar';

x = double(X);

NbColors = 255;

map = gray(NbColors);

x = uint8(x);

% A wavelet decomposition of the image

[c,s] = wavedec2(x,n,wname);

% wdcbm2 for selecting level dependent thresholds

alpha = 1.5; m = 2.7\*prod(s(1,:));

[thr,nkeep] = wdcbm2(c,s,alpha,m)

% Compression

[xd,cxd,sxd,perf0,perfl2] = wdencmp('lvd',c,s,wname,n,thr,'h');

disp('Compression Ratio');

disp(perf0);

% Decompression

R = waverec2(c,s,wname);

rc = uint8(R);

% Plot original and compressed images.

subplot(221), image(x);

colormap(map);

title('Original image')

subplot(222), image(xd);

colormap(map);

title('Compressed image')

% Displaying the results

xlab1 = ['2-norm rec.: ',num2str(perfl2)];

xlab2 = [' % -- zero cfs: ',num2str(perf0), ' %'];

xlabel([xlab1 xlab2]);

subplot(223), image(rc);

colormap(map);

title('Reconstructed image');

%Computing the image size

disp('Original Image');

imwrite(x,'original.tif');

imfinfo('original.tif')

disp('Compressed Image');

imwrite(xd,'compressed.tif');

imfinfo('compressed.tif')

disp('Decompressed Image');

imwrite(rc,'decompressed.tif');

imfinfo('decompressed.tif')

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(hObject, eventdata, handles)

global rc;

global X;

global xd;

subplot(221);

imhist(rc);

subplot(222);

imhist(xd);

subplot(223);

imhist(X);

% --------------------------------------------------------------------

function File\_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------

function About\_Callback(hObject, eventdata, handles)

about;

% --------------------------------------------------------------------

function Help\_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------

function New\_Callback(hObject, eventdata, handles)

global X;

[filename, pathname] = uigetfile('\*.m', 'Pick a MATLAB code file');

if isequal(filename,0) || isequal(pathname,0)

disp('User pressed cancel')

else

X=imread(filename);

imshow(X);

end

% --------------------------------------------------------------------

function Exit\_Callback(hObject, eventdata, handles)

delete(get(0,'Children'))

function axes1\_CreateFcn(hObject, eventdata, handles)

im4=imread('C:\Program Files\MATLAB\R2012a\bin\project\image2.jpg');

imshow(im4);

im3=imread('C:\Program Files\MATLAB\R2012a\bin\project\im3.png');

imshow(im3);

function axes3\_CreateFcn(hObject, eventdata, handles)

im1=imread('C:\Program Files\MATLAB\R2012a\bin\project\im1.jpeg');

imshow(im1);

%%%%new gui for about

function varargout = about(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @about\_OpeningFcn, ...

'gui\_OutputFcn', @about\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before about is made visible.

function about\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = about\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

im5=imread('C:\Program Files\MATLAB\R2012a\bin\project\shi.jpg');

imshow(im5);

**Conclusion**

A picture can say more than a thousand words. However, storing an image can cost more than a million words. This is not always a problem because now computers are capable enough to handle large amounts of data. However, it is often desirable to use the limited resources more efficiently. For instance, digital cameras often have a totally unsatisfactory amount of memory and the internet can be very slow. In these cases, the importance of the compression of image is greatly felt. The rapid increase in the range and use of electronic imaging justifies attention for systematic design of an image compression system and for providing the image quality needed in different applications. Wavelet can be effectively used for this purpose. A low complex 2D image compression method using Haar wavelets as the basis functions along with the quality measurement of the compressed images have been presented here. As for the further work, the tradeoff between the value of the threshold ε and the image quality can be studied and also fixing the correct threshold value is also of great interest. Furthermore, finding out the exact number of transformation level required in case of application specific image compression can be studied. Also, more thorough comparison of various still image quality measurement algorithms may be conducted. Though many published algorithms left a few parameters unspecified, here good estimates of them for implementation have been provided. All these metrics, including ours, did very well in estimating the perceptual error, so that it is difficult to conclude any decisive advantage of one algorithm over another.

**References:**

* Colm Mulcahy, Image compression using the Haar Wavelet transforms, Internal Report.
* Robert L. Cook and Tony DeRose, Wavelet Noise, ACM Transactions on Graphics, July 2005, Volume 24 Number 3, Proc. Of ACM SIGGRAPH 2005, pp. 803-811.
* Gersho, A. and Gray, R. M., Vector Quantization and Signal Compression, Kluwer Academic Publishers, 1991.
* [www.mathworks.in](http://www.mathworks.in)
* Wikipedia
* [www.google.com](http://www.google.com)
* [www.youtube.com](http://www.youtube.com)
* [www.debugmode.com/**image**cmp/](http://www.debugmode.com/imagecmp/)
* Eric J. Stollnitz, Tony D. Derose and David H. Salesin, Wavelets for Computer Graphics- Theory and Applications Book, Morgan Kaufmann Publishers, Inc. San Francisco, California.
* Pennebaker, W. B. and Mitchell, J. L. JPEG, Still Image Data Compression Standards, Van Nostrand Reinhold, 1993.