MOVEMENT DETECTION IN A LIVE VIDEO WITH FRAMES COMPRESSED USING DWT

By

Kashyap Rastogi-051104
Himanshu Patidar-051123
Ashish Pankaj Singh-051030

MAY-2009

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled, "Movement Detector" submitted by Kashyap Rastogi, Himanshu Patidar and, Ashish Pankaj Singh in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communications Engineering of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

[illegible] [illegible]

[Dr. S.V. Bhooshan] **[Dr. Vinay Kumar]**

ACKNOWLEDGMENT

There were many steps to take to complete this work. Each step forward would have been impossible without the help of many people. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

First of all, we are deeply grateful to our advisor, Mr. Vinay Kumar. We really appreciate the patience he had towards our project. We acknowledge his advice and guidance throughout the year.

We would also like to express our profound gratitude to our panelists for their constant and valuable suggestions during the development of this project.

Last but not least, we express our heartiest gratitude to Almighty God, and our parents for their love and blessings to complete the project successfully.

[illegible]
[Ashish Pankaj Singh]

Himanshu
[Himanshu Patidar]

[illegible]
[Kashyap Rastogi]

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

In the present world scenario, where security breaches are getting common to see, A Movement Detector can be used for security purposes in high security zones. The system would be developed in MATLAB(Matrix Laboratory), using the Image Processing Toolbox. This security system would detect any movement in the area covered by the Camera installed.

CHAPTER 1

INTRODUCTION

The movement detector would be developed using Image Processing Toolbox of MATLAB. The initial step would be the live feeding of the video into MATLAB. Then, frames would be extracted from the running video. Further these frames would be compressed in order to reduce the memory usage. To make the system more efficient, we need to device a compression algorithm which fulfills our needs.

Finally, the motion will be detected by finding the absolute difference between two successive frames. If at any time, frame difference is found to be non-zero, then a movement must have occurred.

1.1 Terminologies:

Digital Video – A 'real' visual scene is continuous both spatially and temporally. In order to represent and process a visual scene digitally, it is vital to sample the real scene spatially or temporally. A digital video is the representation of a spatio-temporally sampled video scene in digital form.

Video Image - It is a projection of a 3-D scene onto a 2-D plane. A 3-D scene consisting of a number of objects each with depth, texture and, illumination is projected onto a plane to form a 2-D representation of the scene. This 2-D representation contains varying texture and illumination but, no depth.

Video Sampling –

A video has to be sampled spatially, as well as temporally. To obtain a 2D sampled image, a camera focuses a 2 D projection of the video scene onto a sensor, such as an array of charge coupled devices (CCD array). The image so obtained may be continuous with respect to the x and y coordinates and also in amplitude.

Digitizing the co-ordinates values is called Sampling, while digitizing the amplitude values is known as Quantization.

The two process involved in video sampling are spatial sampling, and temporal sampling.

Spatial Sampling - The output of a CCD array is varying electrical that represents a video image. Sampling this signal at a point in time produces a sampled image or frame that has defined values at a set of sampling points.

A common format for a sampled image is a rectangle with a sampling points positioned on a square or rectangular grid. Sampling occurs at each of intersection points on the grid and the sampled image may be reconstructed by representing each sample as a square picture element or a pixel. The visual quality of the image is influenced by the no. of sampling points.

Temporal Sampling – A moving video image is captured by taking a rectangular 'snapshot' of the signal at a periodic time intervals. Playing back the series of frames produces the appearance of motion.

Higher is the temporal sampling, smoother is the motion in the video scene but this requires more samples to be captured and stored. Higher fps (frames per second) can be achieved on the expense of a higher data rate.

1.2 RGB color model

The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue. The main purpose of the RGB color model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers. Computer monitors reproduce every color using a combination of red (R), green (G), and blue (B) light. R, G, and B each have 256 possible depths, making it possible for your computer to generate more than 16 million different possible RGB colors.

Each RGB color is defined using three numbers, also known as values: the first represents red, the second represents green, and the third represents blue. The values in each channel range from 0 (the darkest shade) to 255 (the brightest shade): While the deepest shade of black would have an RGB value of 0 0 0, white would have an RGB value of 255 255 255, bright red would be 255 0 0, and so on.

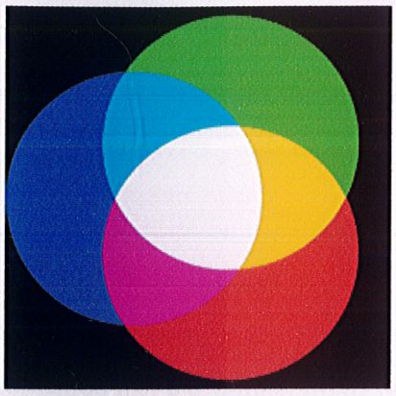


Figure 1.1 A visual representation of RGB model.

CHAPTER 2

IMAGE ACQUISITION PROCEDURE

In order to acquire images from a live video stream we have to first find the certain parameters about the image acquisition device (webcam). These parameters can be found from certain steps:

2.1 Retrieve Hardware Information

In this step, we get lots of information that the toolbox needs to uniquely identify the image acquisition device you want to access. We use the 'imaqhwinfo' function to retrieve each item.

2.1.1 Adaptor name

An adaptor is the software that the toolbox uses to communicate with an image acquisition device via its device driver.

- Determining the Adaptor Name

To determine the name of the adaptor, enter the imaqhwinfo function at the MATLAB prompt without any arguments.

imaqhwinfo

In the data returned by imaqhwinfo, the InstalledAdaptors field lists the adaptors that are available on our computer. In our case, imaqhwinfo found two adaptors available on the computer: 'coreco' and 'winvideo'

2.1.2 Device ID

The device ID is a number that the adaptor assigns to uniquely identify each image acquisition device with which it can communicate.

- Determining the Device ID

To find the device ID of a particular image acquisition device, enter the imaqhwinfo function at the MATLAB prompt, specifying the name of the adaptor as the only argument.

```
info = imaqhwinfo('winvideo')
```

2.1.3 Video format

The video format specifies the image resolution (width and height) and other aspects of the video stream. Image acquisition devices typically support multiple video formats.

- Determining the Supported Video Formats

To determine which video formats an image acquisition device supports, look in the Devicelnfo field of the data returned by imaqhwinfo. The Devicelnfo field is a structure array where each structure provides information about a particular device. To view the device information for a particular device, we can use the device ID as a reference into the structure array.

To get the list of the video formats supported by a device, look at SupportedFormats field in the device information structure. The SupportedFormats field is a cell array of strings where each string is the name of a video format supported by the device.

```
dev_info = imaqhwinfo('winvideo',1)

dev_info =

          DefaultFormat: 'RGB24_640x480'
          DeviceFileSupported: 0
          DeviceName: 'Integrated Camera'
         DeviceID: 1
         ObjectConstructor: 'videoinput('winvideo', 1)'
         SupportedFormats: {1x13 cell}
```

2.2 Create a Video Input Object

- Video Input Object

A video input object is an object that the toolbox uses to represent the connection between MATLAB and an image acquisition device.

Syntax

```
obj = videoinput(adaptorname)
obj = videoinput(adaptorname,deviceID)
obj = videoinput(adaptorname,deviceID,format)
```

2.3 Configure Object Properties

After creating the video input object and previewing the video stream, we want to modify characteristics of the image or other aspects of the acquisition process.

By typing get(getselectedsource(videoinputobject)) function on the matlab prompt we can view the characteristics of the video input object.

- Setting Object Properties

To set the value of a video input object property or a video source object property, we can use the set function or you can reference the object property as you would a field in a structure, using dot notation.

To implement continuous image acquisition, the example sets the TriggerRepeat property to Inf. To set this property using the set function, enter this code at the MATLAB prompt.

```
set(vid,'TriggerRepeat',Inf);
```

If the value of TriggerRepeat function is set to Inf then it will continue until a stop function is issued.

To help the application keep up with the incoming video stream while processing data, the example sets the FrameGrabInterval property to n (any integer value e.g. 5). This specifies that the object acquire every nth frame in the video stream.

```
vid.FrameGrabInterval = 5;
```

To set the value of a video source object property, we must first use the getselectedsource function to retrieve the object.

```
vid_src = getselectedsource(vid);
set(vid_src,'Tag','motion detector');
```

2.4 Acquire Image Data

After we create the video input object and configure its properties, now we can acquire data.

To start the code we should use some additional statements like figure or start etc.

```
- To create a figure window
  figure;
- To start acquiring frames
  start(vid)
 - To calculate difference image and display it.
   while(vid.FramesAcquired<=100)
      data = getdata(vid,2);
      diff_im = imabsdiff(data(:,:,:,1),data(:,:,:,2));
      imshow(diff_im);
   end
 Stop (vid)
```

CHAPTER 3

IMAGE COMPRESSION

We have used Discrete Wavelet Transform for compressing images, which has been discussed below-

3.1 Fourier Transform

First of all, why do we need a transform, or what is a transform anyway?
Mathematical transformations are applied to signals to obtain further information from that signal that is not readily available in the raw signal. Here, we assume a time-domain signal as a raw signal, and a signal that has been "transformed" by any of the available mathematical transformations as a processed signal.
Most of the signals in practice are TIME-DOMAIN signals in their raw format. That is, whatever that signal is measuring, is a function of time. In other words, when we plot the signal, one of the axes is time (independent variable), and the other (dependent variable) is usually the amplitude. This representation is not always in many cases, the most distinguished information is hidden in the frequency content of the signal. The frequency SPECTRUM of a signal is basically the frequency components (spectral components) of that signal. The frequency spectrum of a signal shows what frequencies exist in the signal. The frequency is measured in cycles/second, or with a more common name, in "Hertz".

So how do we measure frequency, or how do we find the frequency content of a signal? The answer is FOURIER TRANSFORM (FT). If the FT of a signal in time domain is taken, the frequency-amplitude representation of that signal is obtained. In other words, we now have a plot with one axis being the frequency and the other being the amplitude. This plot tells us how much of each frequency exists in our signal.

Now, Recall that the FT gives the frequency information of the signal, which means that it tells us how much of each frequency exists in the signal, but it does not tell us when in time these frequency components exist. This information is not required when the signal is so-called stationary.

Signals whose frequency content does not change in time are called stationary signals. In other words, the frequency content of stationary signals does not change in time. In this case, one does not need to know at what times frequency components exist, since all frequency components exist at all times!!! .

For example the following signal
x(t)=cos(5*t)+cos(10*t)+cos(15*t)+cos(20*t) is a stationary signal, because it has frequencies of 5, 10, 15, and 20 Hz at any given time instant. This signal is plotted below:

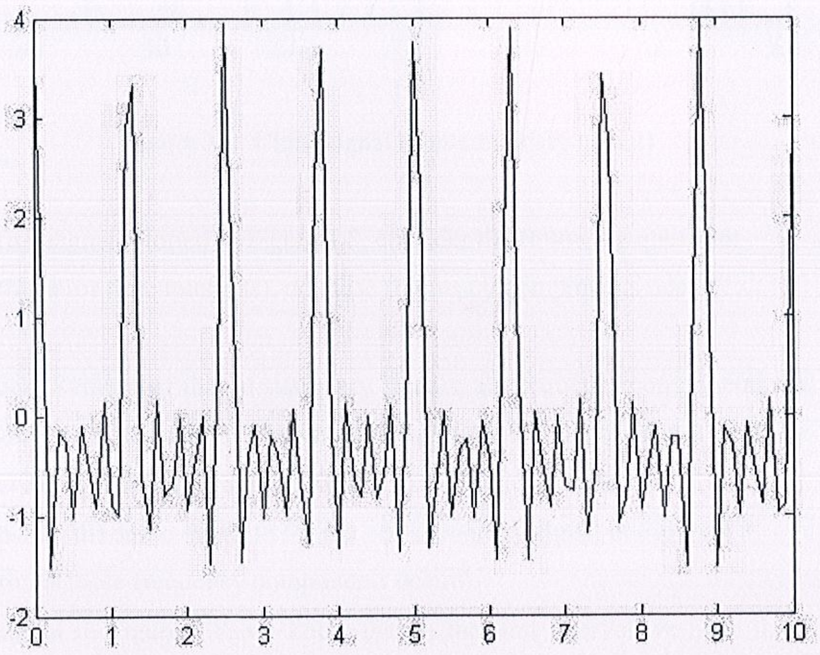


Figure 3.1 A stationary signal

Let's look at another example. Figure 3.2 plots a signal with three different frequency components at three different time intervals, hence a non-stationary signal.

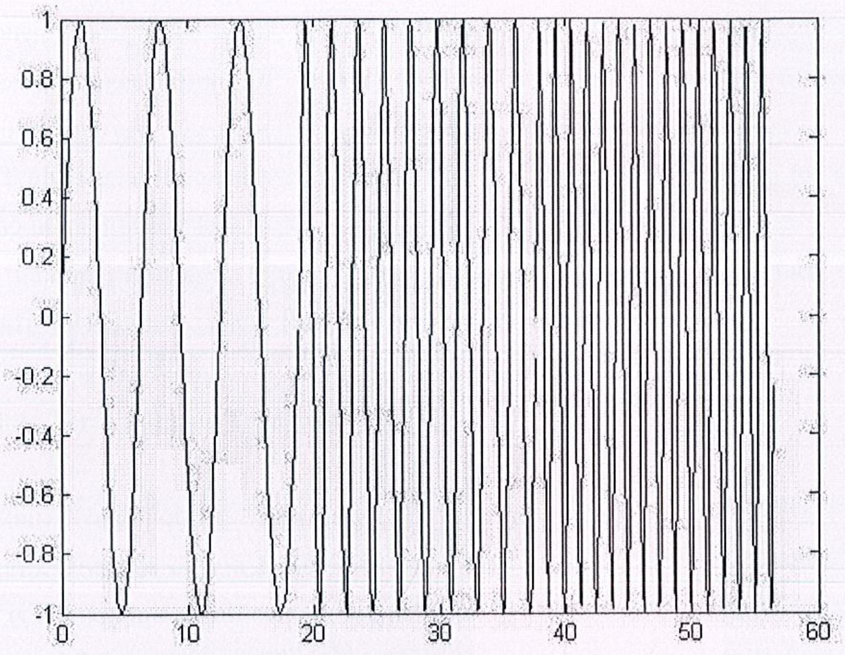


Figure 3.2 Chirp signal (non-stationary signal)

\For the first signal, plotted in Figure 1.5, consider the following question:

At what times (or time intervals), do these frequency components occur?

Answer:

At all times! Remember that in stationary signals, all frequency components that exist in the signal, exist throughout the entire duration of the signal. There is 10 Hz at all times, there is 50 Hz at all times, and there is 100 Hz at all times.

Now, consider the same question for the non-stationary signal in Figure 1.8.

At what times these frequency components occur?

For the signal in Figure 1.8, we know that in the first interval we have the highest frequency component, and in the last interval we have the lowest frequency component. Therefore, for this signal the frequency components do not appear at all times!

Recall that the FT gives the spectral content of the signal, but it gives no information regarding where in time those spectral components appear. Therefore, FT is not a suitable technique for non-stationary signal, with one exception:

FT can be used for non-stationary signals, if we are only interested in what spectral components exist in the signal, but not interested where these occur. However, if this information is needed, i.e., if we want to know, what spectral component occur at what time (interval) , then Fourier transform is not the right transform to use.
The FT gives what frequency components (spectral components) exist in the signal. Nothing more, nothing less.
When the time localization of the spectral components are needed, a transform giving the TIME-FREQUENCY REPRESENTATION of the signal is needed.

3.2 Short Term Fourier Transform (STFT)

The Short Time Fourier Transform (STFT) is a revised version of the Fourier transform. There is only a minor difference between STFT and FT. In STFT, the signal is divided into small enough segments, where these segments of the signal can be assumed to be stationary. For this purpose, a window function "w" is chosen. The width of this window must be equal to the segment of the signal where its stationarity is valid.
This window function is first located to the very beginning of the signal. That is, the window function is located at t=0. Let's suppose that the width of the window is "T" s. At this time instant (t=0), the window function will overlap with the first T/2 seconds. The window function and the signal are then multiplied. By doing this, only the first T/2 seconds of the signal is being chosen, with the appropriate weighting of the window. Then this product is assumed to be just another signal, whose FT is to be taken. In other words, FT of this product is taken, just as taking the FT of any signal.
The result of this transformation is the FT of the first T/2 seconds of the signal. If this portion of the signal is stationary, as it is assumed, then there will be no problem and the obtained result will be a true frequency representation of the first T/2 seconds of the signal.
The next step, would be shifting this window (for some t1 seconds) to a new location, multiplying with the signal, and taking the FT of the product. This procedure is followed, until the end of the signal is reached by shifting the window with "t1" seconds intervals. The following definition of the STFT summarizes all the above explanations in one line:

$$STFT_X^{(\omega)}(t,f) = \int_t [x(t) \bullet \omega^*(t - t')] \bullet e^{-j2\pi ft} dt \ldots\ldots(3)$$

x(t) is the signal itself, w(t) is the window function, and * is the complex conjugate.

STFT gives us a true time-frequency representation of the signal. We not only know what frequency components are present in the signal, but we also know where they are located in time. Here but STFT has some limitations due to which we have to use wavelet transform.

Now, in STFT, our window is of finite length, thus it covers only a portion of the signal, which causes the frequency resolution to get poorer. What we mean by getting poorer is that, we no longer know the exact frequency components that exist in the signal, but we only know a band of frequencies that exist:

If we use a window of infinite length, we get the FT, which gives perfect frequency resolution, but no time information. Furthermore, in order to obtain the stationarity, we have to have a short enough window, in which the signal is stationary. The narrower we make the window, the better the time resolution, and better the assumption of stationarity, but poorer the frequency resolution:

Narrow window ===>good time resolution, poor frequency resolution.
Wide window ===>good frequency resolution, poor time resolution.

Anyone who would like to use STFT is faced with this problem of resolution. What kind of a window to use? Narrow windows give good time resolution, but poor frequency resolution. Wide windows give good frequency resolution, but poor time resolution; furthermore, wide windows may violate the condition of stationarity. The problem, of course, is a result of choosing a window function, once and for all, and use that window in the entire analysis. The answer, of course, is application dependent: If the frequency components are well separated from each other in the original signal, than we may sacrifice some frequency resolution and go for good time resolution, since the spectral components are already well separated from each other.

3.2.1 Multiresolution Analysis

Although the time and frequency resolution problems are results of a physical phenomenon (the Heisenberg uncertainty principle) and exist regardless of the transform used, it is possible to analyze any signal by using an alternative approach called the multiresolution analysis (MRA). MRA, as implied by its name, analyzes the signal at different frequencies with different resolutions. Every spectral component is not resolved equally as was the case in the STFT.

MRA is designed to give good time resolution and poor frequency resolution at high frequencies and good frequency resolution and poor time resolution at low frequencies. This approach makes sense especially when the signal at hand has high frequency components for short durations and low frequency components for long durations.

3.3 The Continuous Wavelet Transform

The continuous wavelet transform was developed as an alternative approach to the short time Fourier transform to overcome the resolution problem. The wavelet analysis is done in a similar way to the STFT analysis, in the sense that the signal is multiplied with a function, similar to the window function in the STFT, and the transform is computed separately for different segments of the time-domain signal. However, there are two main differences between the STFT and the CWT:

1. The Fourier transforms of the windowed signals are not taken, and therefore single peak will be seen corresponding to a sinusoid, i.e., negative frequencies are not computed.

2. The width of the window is changed as the transform is computed for every single spectral component, which is probably the most significant characteristic of the wavelet transform.

The continuous wavelet transform is defined as follows

$$CWT_x^{\psi}(\tau, s) = \Psi_x^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\psi^*\left(\frac{t-\tau}{s}\right) dt$$

As seen in the above equation , the transformed signal is a function of two variables, tau and s , the translation and scale parameters, respectively. psi(t) is the transforming function, and it is called the mother wavelet . The term mother wavelet gets its name due to two important properties of the wavelet analysis as explained below:

The term wavelet means a small wave . The smallness refers to the condition that this function is of finite length. The term mother implies that the functions with different region of support that are used in the transformation process are derived from one main function, or the mother wavelet. In other words, the mother wavelet is a prototype for generating the other window functions.

The term translation is used in the same sense as it was used in the STFT; it is related to the location of the window, as the window is shifted through the signal. This term, obviously, corresponds to time information in the transform domain. However, we do not have a frequency parameter, as we had before for the STFT. Instead, we have scale parameter which is defined as "1/frequency". The term frequency is reserved for the STFT.

3.3.1 Scale

The parameter scale in the wavelet analysis is similar to the scale used in maps. As in the case of maps, high scales correspond to a non-detailed global view of the signal, and low scales correspond to a detailed view. Similarly, in terms of frequency, low frequencies (high scales) correspond to a global information of a signal (that usually spans the entire signal), whereas high frequencies (low scales) correspond to a detailed information of a hidden pattern in the signal (that usually lasts a relatively short time

Scaling, as a mathematical operation, either dilates or compresses a signal. Larger scales correspond to dilated (or stretched out) signals and small scales correspond to compressed signals. In terms of mathematical functions, if f(t) is a given function f(st) corresponds to a contracted (compressed) version of f(t) if s > 1 and to an expanded (dilated) version of f(t) if s < 1 .

However, in the definition of the wavelet transform, the scaling term is used in the denominator, and therefore, the opposite of the above statements holds, i.e., scales s > 1 dilates the signals whereas scales s < 1 , compresses the signal.

3.3.2 Computation Of CWT

Interpretation of the above equation will be explained in this section. Let x(t) is the signal to be analyzed. The mother wavelet is chosen to serve as a prototype for all windows in the process. All the windows that are used are the dilated (or compressed) and shifted versions of the mother wavelet.

Once the mother wavelet is chosen the computation starts with s=1 and the continuous wavelet transform is computed for all values of s, smaller and larger than ``1''. However, depending on the signal, a complete transform is usually not necessary. For all practical purposes, the signals are bandlimited, and therefore, computation of the transform for a limited interval of scales is usually adequate

For convenience, the procedure will be started from scale s=1 and will continue for the increasing values of s , i.e., the analysis will start from high frequencies and proceed towards low frequencies. This first value of s will correspond to the most compressed wavelet. As the value of s is increased, the wavelet will dilate. The wavelet is placed at the beginning of the signal at the point which corresponds to time=0. The wavelet function at scale ``1'' is multiplied by the signal and then integrated over all times. The result of the integration is then multiplied by the constant number 1/sqrt{s}. The final result is the value of the transformation, i.e., the value of the continuous wavelet transform at time zero and scale s=1. In other words, it is the value that corresponds to the point tau =0, s=1 in the time-scale plane.

The wavelet at scale s=1 is then shifted towards the right by tau amount to the location t=tau, and the above equation is computed to get the transform value at t=tau , s=1 in the time-frequency plane.

This procedure is repeated until the wavelet reaches the end of the signal. One row of points on the time-scale plane for the scale s=1 is now completed. Then, s is increased by a small value. Note that, this is a continuous transform, and therefore, both tau and s must be incremented continuously. However, if this transform needs to be computed by a computer, then both parameters are increased by a sufficiently small step size. This corresponds to sampling the time-scale plane. The above procedure is repeated for every value of s. Every computation for a given value of s fills the corresponding

The Wavelet Series is just a sampled version of CWT and its computation may single row of the time-scale plane. When the process is completed for all desired values of s, the CWT of the signal has been calculated. The figures below illustrate the entire process step by step.

If the signal has a spectral component that corresponds to the current value of s (which is 1 in this case), the product of the wavelet with the signal at the location where this spectral component exists gives a relatively large value. If the spectral component that corresponds to the current value of s is not present in the signal, the product value will be relatively small, or zero. For high scales, on the other hand, the continuous wavelet transform will give large values for almost the entire duration of the signal, since low frequencies exist at all times.

3.4 Discrete Wavelet Transform

In numerical analysis and functional analysis, a discrete wavelet transform (DWT) is any wavelet transform for which the wavelets are discretely sampled. As with other wavelet transforms, a key advantage it has over Fourier transforms is temporal resolution: it captures both frequency *and* location information.

The Wavelet Transform is developed to overcome the short comings of STFT, which can also be used to analyze the non-stationary signals.

While STFT gives constant resolution at all frequencies, DWT uses multi-resolution techniques which gives different frequencies at different resolutions.

3.4.1 The Subband Coding and The Multiresolution Analysis

The main idea is the same as it is in the CWT. A time-scale representation of a digital signal is obtained using digital filtering techniques. Recall that the CWT is a correlation between a wavelet at different scales and the signal with the scale (or the frequency) being used as a measure of similarity. The continuous wavelet transform was computed by changing the scale of the analysis window, shifting the window in time, multiplying by the signal, and integrating over all times. In the discrete case, filters of different cutoff frequencies are used to analyze the signal at different scales. The signal is passed through a series of high pass filters to analyze the high frequencies, and it is passed through a series of low pass filters to analyze the low frequencies.

The resolution of the signal, which is a measure of the amount of detail information in the signal, is changed by the filtering operations, and the scale is changed by upsampling and downsampling (subsampling) operations. Subsampling a signal corresponds to reducing the sampling rate, or removing some of the samples of the signal. For example, subsampling by two refers to dropping every other sample of the signal. Subsampling by a factor *n* reduces the number of samples in the signal *n* times.

Upsampling a signal corresponds to increasing the sampling rate of a signal by adding new samples to the signal. For example, upsampling by two refers to adding a new sample, usually a zero or an interpolated value, between every two samples of the signal. Upsampling a signal by a factor of *n* increases the number of samples in the signal by a factor of *n*.

Here the signal is sent through a series of High and Low pass filters depending upon the number of levels. This is called 1-D DWT because the DWT is applied here in one plane only.

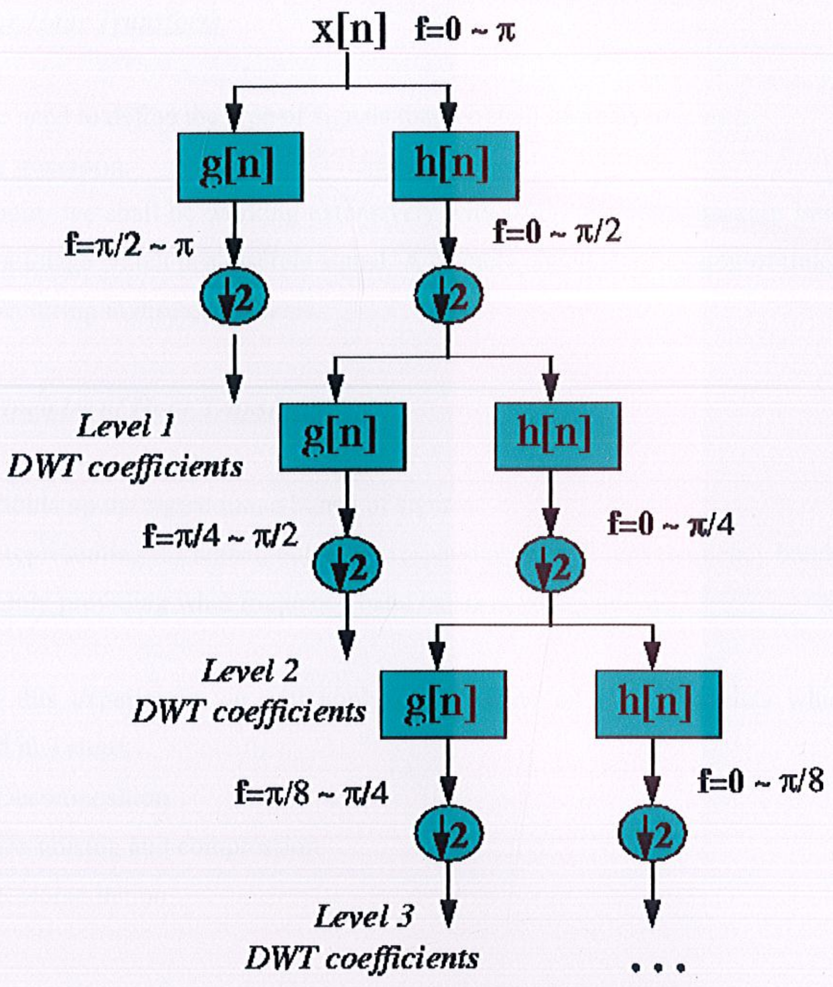


Figure 3.3 Subband coding

3.5 Haar Wavelet

A Haar wavelet is the simplest type of wavelet. In discrete form, Haar wavelets are related to a mathematical operation called the Haar transform. The Haar transform serves as a prototype for all other wavelet transforms.

3.5.1 The Haar Transform

First, we need to define the type of signals that we shall be analyzing with the Haar transform.

Throughout we shall be working extensively with *discrete signals* because here our input is a image which is a discrete signal. A discrete signal is a function of time with values occurring at discrete instants.

3.5.2 Principles of Haar Transform

- Splits up the signal into a bunch of signals.
- Representing the signal, but all corresponding to different frequency bands.
- Only providing what frequency band exists at what interval.

Here in this experiment we will apply compression on Haar Wavelets which is followed in 3 steps:

1. Decomposition
2. De-noising and compression
3. Reconstruction

Like all wavelet transforms, the Haar transform decomposes a discrete signal into two sub signals of half its length. One subsignal is a running average or *trend;* the other sub signal is a running difference or *fluctuation.*

Average or trend gives us the low frequency content (detail of the image) of the signal whereas running difference or fluctuations gives us the high frequency component (outline of the image).

3.5.3 Procedure

1. Decomposition

   Haar transform break the image matrix into 2 different sets namely, averaging and difference coefficients gives the detail and outline information respectively. Here we will be first decomposing the input image in Red, Green, Blue colors. We will be using function *'wavedec2'* in ***MATLAB***, here 2 in wavedec2 stands for 2-D DWT, which is discussed in detail in the next heading. It breaks the input signal coefficients into averaging and difference coefficients trend.

F[n] = [f1 f2 f3 f4 f5 f6 f7 f8]

- Find the average of each pair of samples

  A = (f1 + f2) /2
- Find the difference between the average and sample.

  D = f1 – D
- Fill the first half with averages
- Fill the second half with differences
- Repeat the process on the first half

Lets consider an example:

F(n) = [3 5 4 8 13 7 5 3]

- Step 1

[3 5 4 8 13 7 5 3]

[4 6 10 4 -1 -2 3 1]

Ex (3+5)/2 = 4

Ex 3 – 4 = -1

Fill the first 4 places in the matrix with detail coefficients and the rest 4 with difference.

- Step 2

Here the components are again processed. The difference coefficients obtained from step 1 are copied same and the step 1 will be applied only to average coefficients.

[4 6 10 4 -1 -2 3 1]

[5 7 -1 3 -1 -2 3 1]

Ex (4+6)/2 = 5

Ex 4 -5 = -1

- Step 3

Now the difference coefficients obtained from step 2 are copied and the process is applied to the new obtained average coefficients.

[5 7 -1 3 -1 -2 3 1]

[6 -1 -1 3 -1 -2 3 1]

Ex (5+7)/2 = 6

Ex 5 – 6 =-1

Here the first value is the averaging coefficient also known as the row average. The rest information is the set of differencing coefficients. The row average is the low frequency content which gives the maximum detail about the image whereas difference shows the various details of the image.

How consider a image as an example:

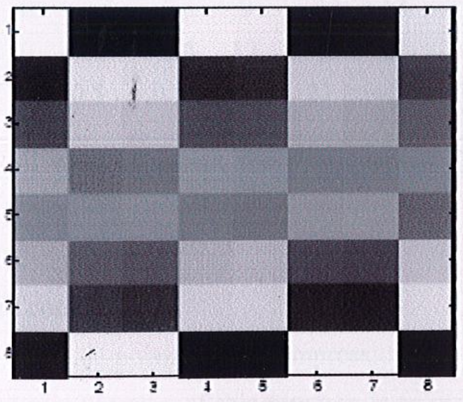


Fig 3.4 Gray scale image

Mathematical representation of image

$$A=\begin{bmatrix}64 & 2 & 3 & 61 & 60 & 6 & 7 & 57\\ 9 & 55 & 54 & 12 & 13 & 51 & 50 & 16\\ 17 & 47 & 49 & 20 & 21 & 43 & 42 & 24\\ 40 & 26 & 27 & 37 & 36 & 30 & 31 & 33\\ 32 & 34 & 35 & 29 & 28 & 38 & 39 & 25\\ 41 & 23 & 22 & 44 & 45 & 19 & 18 & 48\\ 49 & 15 & 14 & 52 & 53 & 11 & 10 & 56\\ 8 & 58 & 59 & 5 & 4 & 62 & 63 & 1\end{bmatrix}$$

Black represents 0 whereas white is representated by 255.

Here the Haar Transform is first applied to rows and columns

After applying Haar Transform:

$$\begin{bmatrix}32.5 & 0 & 0.5 & 0.5 & 31 & -29 & 27 & -25\\ 32.5 & 0 & -0.5 & -0.5 & -23 & 21 & -19 & 17\\ 32.5 & 0 & -0.5 & -0.5 & -15 & 13 & -11 & 9\\ 32.5 & 0 & 0.5 & 0.5 & 7 & -5 & 3 & -1\\ 32.5 & 0 & 0.5 & 0.5 & -1 & 3 & -5 & 7\\ 32.5 & 0 & -0.5 & -0.5 & 9 & -11 & 13 & -15\\ 32.5 & 0 & -0.5 & -0.5 & 17 & -19 & 21 & -23\\ 32.5 & 0 & 0.5 & 0.5 & -25 & 27 & -29 & 31\end{bmatrix}$$

The first element of all rows represent the row average. The rest are the detail coefficients.

2. De-noising and compression

For de-noising and compression we call the '*wdencmp*' in ***MATLAB*** editor. The basic function of this function is to reduce the redundancy by dumping out some values which is decided according to the set *threshold*. Now we have to choose the *threshold* for the matrix according to the compression rates needed.

Threshold = 5

$$\left[\begin{array}{cccc|cccc}
32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
32.5 & 0 & 0 & 0 & 27 & -25 & 23 & -21 \\
32.5 & 0 & 0 & 0 & 11 & 9 & -7 & 0 \\
32.5 & 0 & 0 & 0 & 0 & 7 & -9 & 11 \\
32.5 & 0 & 0 & 0 & 21 & -23 & 25 & 27
\end{array}\right]$$

All the values below 5 will be discarded and the rest remains the same.

RESULT

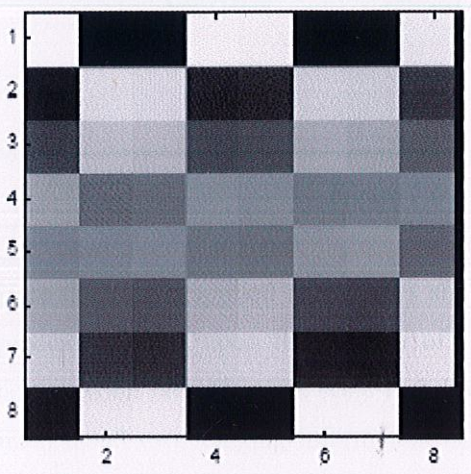


Fig 3.5 (a) Original Image

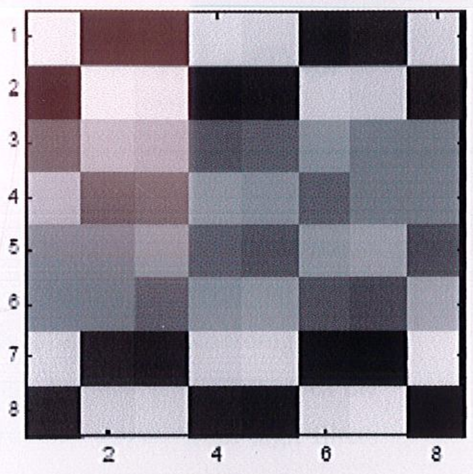


Fig 3.5(b) Decompressed Image

3. Reconstruction

Here the decomposed RGB constituents are added to generate the output compressed image by using the function *waverec2* function in ***MATLAB*** which reconstructs the decomposed 2-D DWT coefficients of R G B constituents. In ***MATLAB*** we have used '*cat*' function which concatinates the image. By accepting three parameters i.e. decomposed R G and B.

3.5.4 2-D DWT

2 dimensional discrete wavelet transform follows the same process as 1-D DWT but the only difference is that now it is applied first along the rows then to columns. Here the signal is passed through a high and a low pass filter. The output of low pass filter is sent again through a low pass filter.

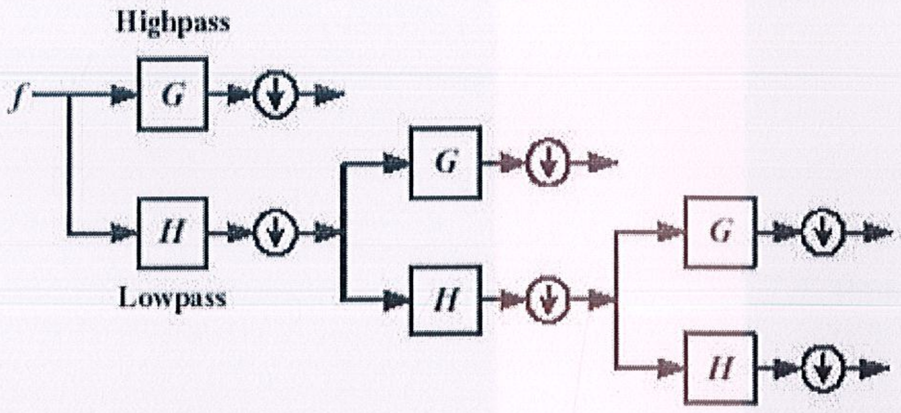


Figure 3.6 DWT procedure

Here the 1-D is applied to the rows and then to the columns. The recursion of this process depends upon the number of scales which is estimated according to the compression rate. Here the scale is 2. The output from low-low pass filter gives the main detail or averaging information and the rest i.e. the outputs from high filters gives the other outline informations.

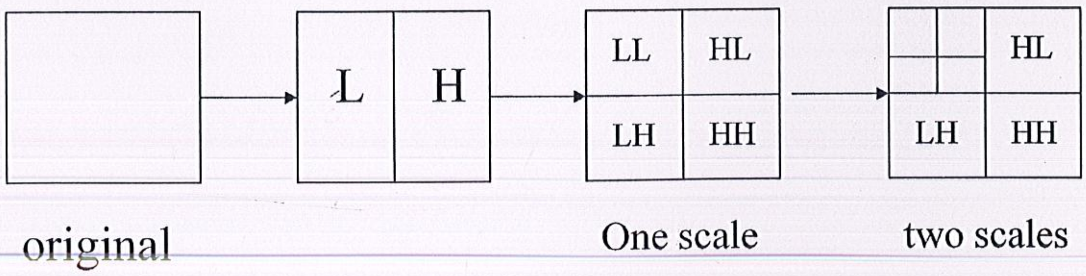


Figure 3.7 Applying DWT to a image

In fig3.7 we used 2 scales. The output from LL gives us the detail or is the value of the row average obtained from Haar Transform and rest are difference values obtained from passing through high pass filter.

Algorithm

1. To gather the information about our image acquisition device,

- First of all, to know the adaptor name, we type on the MATLAB command prompt-

```
Imaqhwinfo
ans =
        InstalledAdaptors: {'coreco'  'winvideo'}
        MATLABVersion: '7.2 (R2006a)'
          ToolboxName: 'Image Acquisition Toolbox'
       ToolboxVersion: '1.10 (R2006a)'
```

- Then, we type the following to know about the device ID-

```
Imaqhwinfo('winvideo')
info =
           AdaptorDllName: [1x81 char]
           AdaptorDllVersion: '1.10 (R2006a)'
           AdaptorName: 'winvideo'
           DeviceIDs: {[1]}
```

- Then, to know the supported video formats, we type-

```
dev_info = imaqhwinfo('winvideo',1)
dev_info =
                DefaultFormat: 'RGB24_640x480'
                DeviceFileSupported: 0
                DeviceName: 'Integrated Camera'
               DeviceID: 1
               ObjectConstructor: 'videoinput('winvideo', 1)'
               SupportedFormats: {1x13 cell}
```

So, the information we gathered after our first step is-

Adaptor name- winvideo

Device ID- 1

Supported Video Format- 'RGB24_640x480'

2. After gathering all the information, open up a new M-file. Now, we create a video input object and set its properties. Then, create an avi file, in which the output video shall be stored.

3. Then, we initialize a while loop with 'frames acquired' as the parameter and set it to the desired value. Inside the loop, we fetch two frames at a time, and assign them to two different variables.

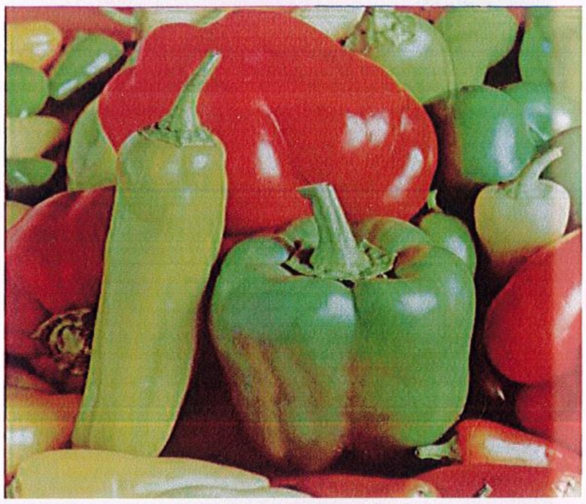
4. After extracting two frames at a time, these frames are compressed using haar wavelets with the help of following steps-
- Firstly, we break up the frame into R,G,B components.
-Then, apply wavelet decomposition to each component using the function 'wavedec2'.
-After this, wavelet coefficient thresholding is applied at 5 levels, using haar wavelets.
-Then, wavelets are reconstructed in the format of R,G,B components.
-Finally, these R,G,B components are added up using 'cat' function to form the compressed image.

5. Now, that we have compressed frames, we find out the absolute difference between them using 'imabsdiff' function.

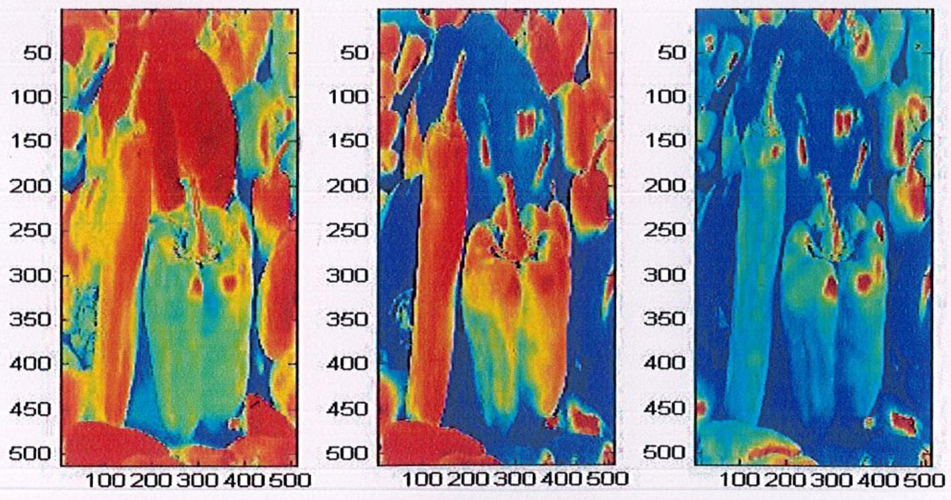
6. Finally, we transverse through every element of the difference matrix calculated in the previous step and pass them through an 'if' condition, to check whether they are non- zero. If any element of the matrix is found to be non-zero, it shows us that the frames were unequal, and hence, we add that frame to the output video created earlier.

Results

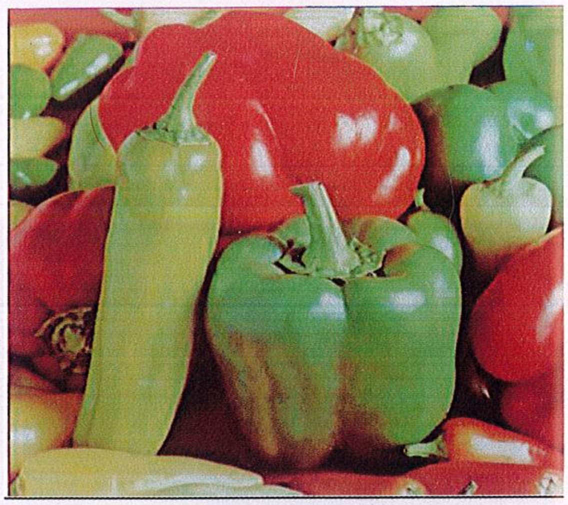
- Results of Compression of a test image


Original test image – 495kB

Compressed R,G and B components



Compressed Image- 175kB



Amount of Compression = ((495-175) / 495) * 100

= 64.64 %