WAVELETS AND MULTIRATE DIGITAL SIGNAL PROCESSING

Lecture 38: Applications of Wavelets

Prof.V. M. Gadre, EE, IIT Bombay

1 Introduction

In this lecture, we shall see two applications of wavelets and time frequency methods in greater depth. The two applications are Data Mining and Face Recognition. Data Mining is a generalization of representation of data. This is presented by Kunal Shah. Face recognition as the name suggests, it is used in security systems and other image processing and vision systems. This is presented by Shah Ronak.

2 Application of Wavelets in Data Mining

Data mining is used for efficient way of representing data. The problem statement is, given a time series data, the time series data may be large, our aim is to improve the efficiency of Multilevel surprise and trend queries. When we have a long time series data, generally we don't encounter point queries.
For example if we record temperature of a particular city for an year we never ask what is the temperature at this particular day or time. We always ask for the trend how the temperature is varying in a particular month. Such queries are called trend queries. Surprise queries are those which deal with queries like sudden change in the temperature of a particular city in a particular month. Wavelets are effiecient in handling such Queries. Multilevel indicates the various levels of abstraction of data whether the data is for a month, or an year, or a decade. So, our first problem is how to represent such a huge amount of data. One of the ways is representing the data in a matrix *i.e.*,

$$\tilde{X} = M \times N$$

Let $\tilde{X}$ be the whole data to be represented and N for example be the stock prices of a company for a year. Then N will be 365 and let M be the total no.of companies for which we are storing the data. So each row of this matrix represents the stock prices of that particular company. This is how we represent data. Now the first thing we need to do is efficiently store this data. Secondly, we require to retrieve data efficiently and thirdly we require to modify data easily. If these three things could be done easily using wavelets then our job is done.
Firstly, let us see other methods in data mining namely **Singular Value Decomposition** . We represent $\tilde{X}$ as follows,

$$\tilde{X} = U\Lambda V$$

where U is a column orthogonal matrix with size $M \times r$ and Λ is a diagonal matrix of size $r \times r$ and V is a row orthogonal matrix of size $r \times N$ where r is the rank of the matrix $\tilde{X}$. Now instead of storing data in a large $M \times N$ matrix we are storing it in 3 small matrices out of which one is a diagonal matrix. Suppose now if we want to retrieve a data of particular company *i.e.*, a row of $\tilde{X}$ the complexity required is of the size V which is $r \times N$. If N corresponds to a decade then it ends up with a huge complexity. One of the other disadvantages of this method is if we want to modify the data we need to recompute all the three matrices again which is

not the case with wavelets. This method is efficient when N is small and there is no need to update data frequently.
Now using wavelets first we take a row X of the matrix $\tilde{X}$ and we decompose it as shown in figure 1. Initially it is decomposed into approximate and detail subspaces AX1 and DX1 and then AX1 is further decomposed to its corresponding approximate and detail subspaces AX2 and DX2 and so on to the third level reducing the length of the data by 2 each time.

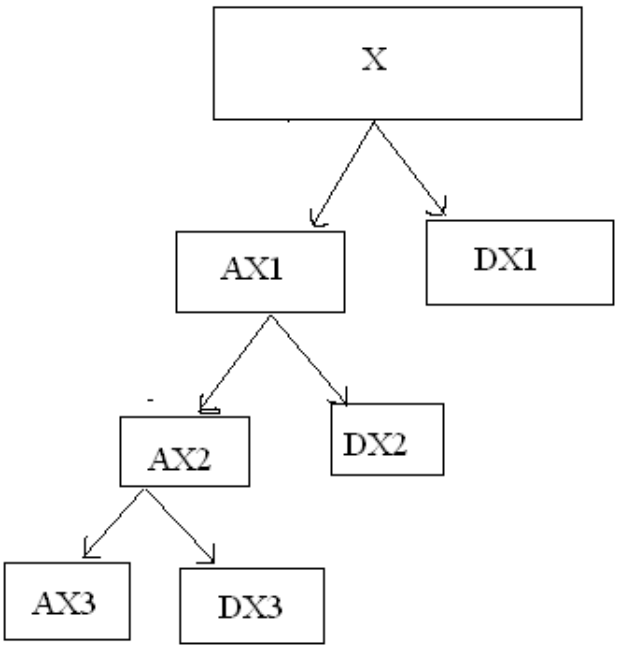


Figure 1: TSA tree

This representation is called as TSA (Trend surprise abstraction) tree. The approximate subspaces store the trend data and the detailed subspace store the surprise data. So, Wavelets naturally decompose the data into trend and surprise data. This is the main advantage of using wavelets. The split and merge operations of the data *i.e.*, decomposing into subspaces and reconstructing it from the subspaces is done according to methods we discussed in previous chapters. Some important properties of this tree are
1)Perfect Reconstruction of the original data using the approximate and detail level subspaces.
2)Power complimentary property *i.e.*, the power of the signal is preserved as we decompose into lower levels.
3)Size of each node reduces by 2 as we go down by one level.
The nodes which are not decomposed into lower levels are called leaf nodes. For example in figure 1 DX1, DX2, AX3 and DX3 are leaf nodes. As we see the split and merge operations are of negligible complexity. The Data extraction operation is the costly operation of all and its cost is directly proportional to size of the data. So the third property is very useful in this case. The leaf nodes are sufficient for surprise and trend queries. For example, if we require the trend query we just need the AX3 level using which we can go to AX2 by passing AX3 through lowpass filter and upsampler and now using AX2 in the similar way we can go to AX1 by which we can get trend queries at different levels. Similarly, if we require surprise queries we use DX3 and pass it through a Highpass filter and reach at the surprise queries at each level.
An optimal TSF tree is that tree which stores only the leaf nodes which incurs minimum cost

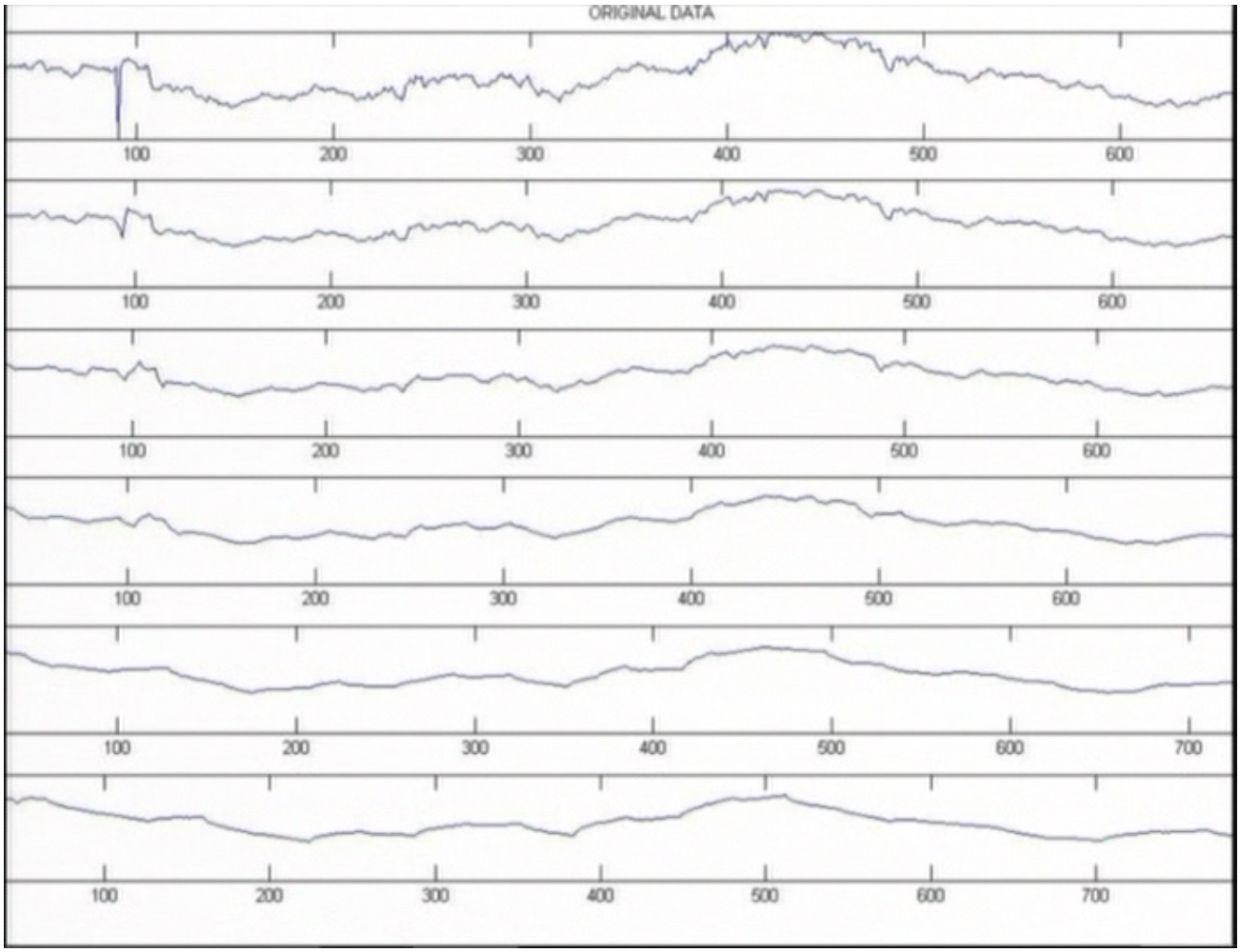


Figure 2: Results for trend query

and minimum storage. So, there is no need to store other nodes. Now, can we further reduce some of the leaf nodes and still arrive at accurate results? The answer is yes. Wavelets are very good at compression. One of the method is node dropping. In this method we exploit the property of orthogonality of wavelets. Suppose we remove a leaf node DX3 and reconstruct the original signal without DX3. Let us call this signal as $\hat{X}$. Now orthogonality property states the below result

$$|| X - \hat{X} ||^2 = \sum_{DX3} || node ||^2$$

which implies that the error between the original and reconstructed signal completely depends on the removed node. Now we can calculate

$$\frac{norm^2(node)}{size(node)}$$

for each leaf node and we store only that nodes which we feel have a significant value in the above equation and rest of the nodes we drop. The disadvantage of this method is that we may loose some important information in the dropped node. So we have another method called coefficient dropping.

In coefficient dropping method we store the leaf nodes in a sequence such that only significant coefficients are stored. Since we are storing the coefficients we are supposed to store their

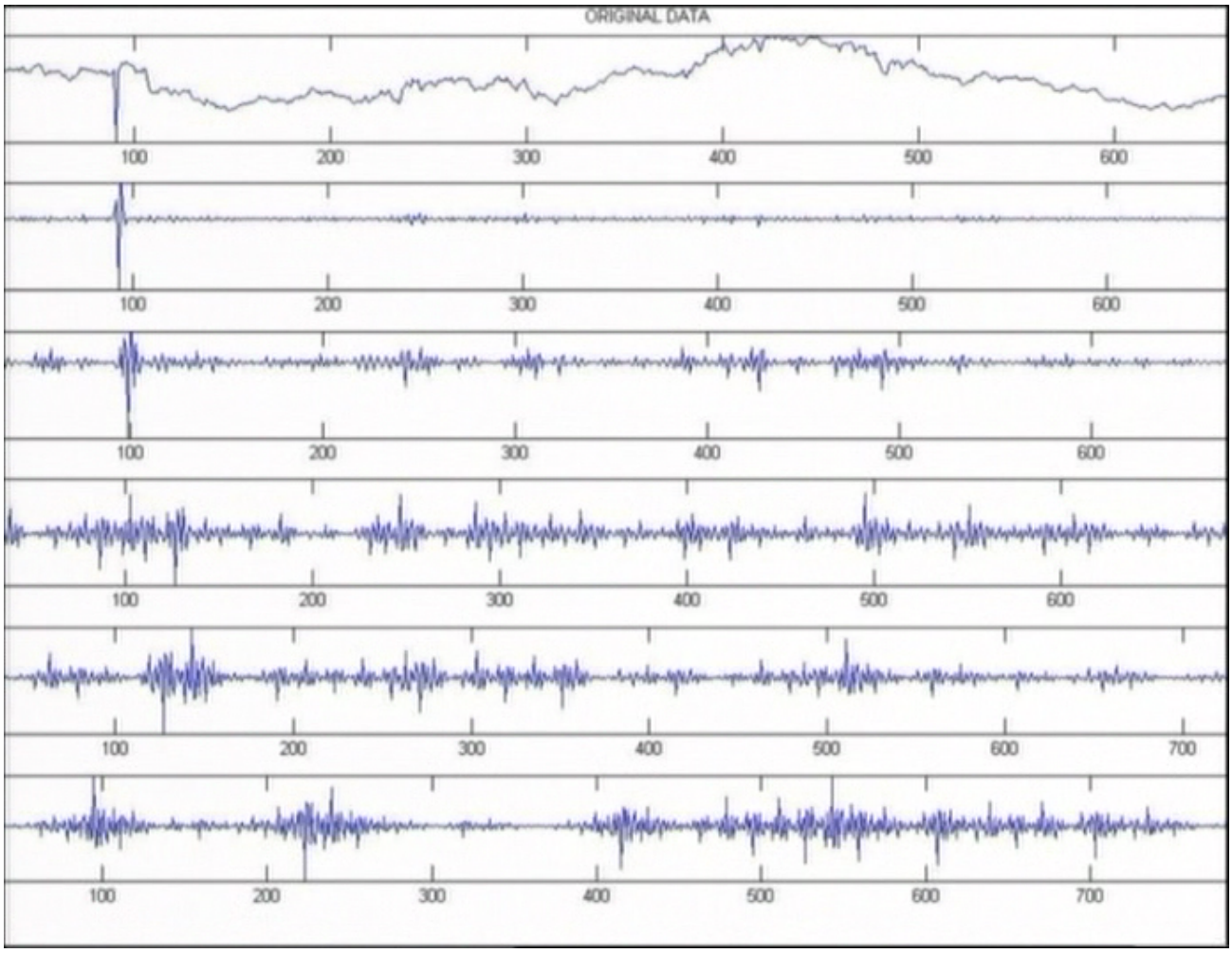


Figure 3: Results for surprise query

index also as to identify to which node it belongs. Here we must be careful with the memory constraints because for each coefficient we are storing two values, one is the coefficent and other is its index. So if we do not drop any coefficient in a node it is better to store entire node with out their indices. We store indices only when the coefficients and indices together added are less than the size of the entire node. This is about Coefficient dropping method.

Let us look at some results. Figure 2 shows the trend graphs. The first graph is original data. It is taken from yahoo stock market and it is of SBI's of 2 years. The second graph is of 2 day decomposition level, Third graph is 4 day level and so on. As we go down we observe that averaging is increased.

Figure 3 shows the results for surprise queries. As we observe, in the original data there is a surprise data which is being averaged out in the subsequent levels. But as we go down subsequent levels we can observe more surprise data which we could not observe in original data.

Figure 4 shows the results for node dropping method. As we observe the recovered data is almost similar to original data except for the small surprise data in the beginning of the original data. Here about 300 coefficients are removed out of 700 and yet we are able to achieve almost our original data.

The figure 5 shows the results for coefficent dropping method. Here we can see that even that small surprise data is also retained. Here Daubechies family of filters are used. As we increase the length of the filter the averaging is done more efficiently.

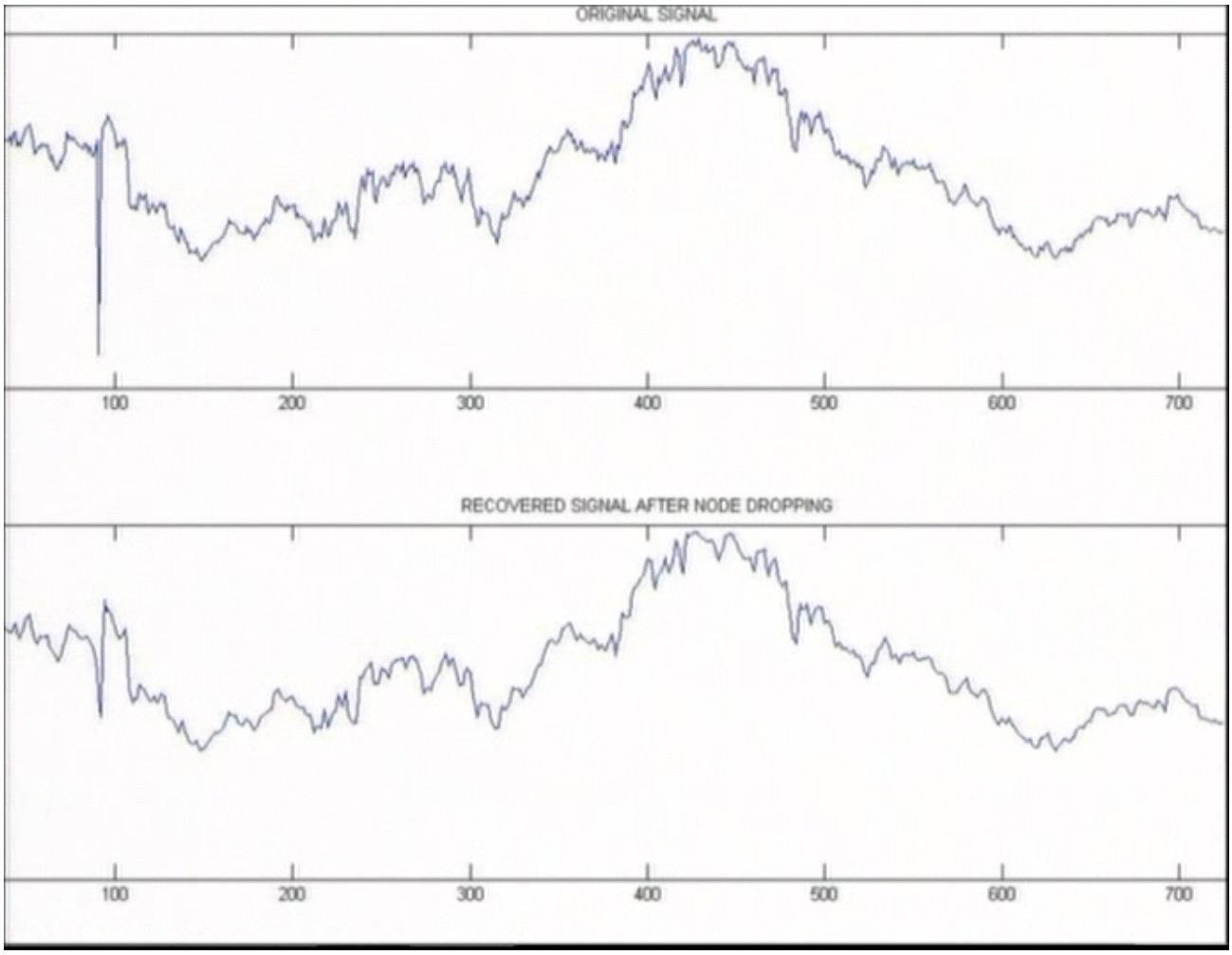


Figure 4: Results for node dropping method

The reference taken by the student for this application is
C.Shahabi, X.Tian, W.Zhao. TSA tree: A Wavelet-based Approach to Improve Multi-Level Surprise and Trend Queries on Time-Series Data. In Statistical and Scientific Database Management, pages 55-68, 2000.

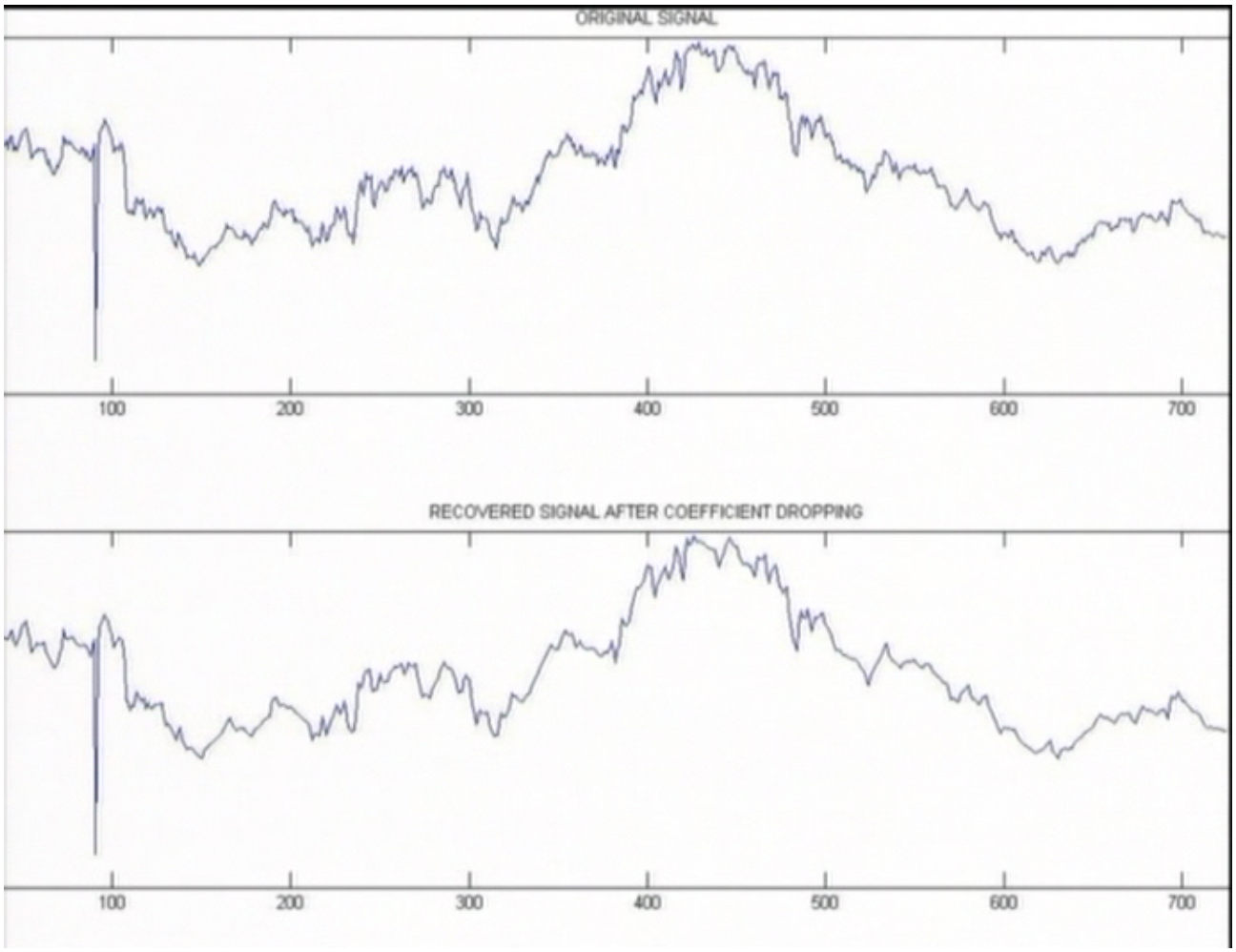


Figure 5: Results for node dropping method

3 Face Recognition through wavepacket analysis

The two keywords in this presentation are Face recognition and another is wave packet analysis. Firstly, why do we need wave packet analysis for face recognition? The answer is, we need decorrelation in spatial domain as well as frequency domain for the task of classifictaion. But, for this wavlets meet the basic requirement. Then why do we go for wavepacket analysis? In wavepacket analysis we decompose detail subspaces not only the approximate subspaces. When we do task of classification we should not miss any information from the underlying signal. The underlying signal here is a face image. The task of Face recognition can be accomplished both by wavelets and wavepacket transform. Wavepacket transform is used for richer representation of image.

Why is Face recognition required? One of the answer is for biometric authentication. Face can be easily morphed by covering with sun glasses or by growing a moustache. But some of the images like retina and fingerprints are difficult to be morphed. However, Face recognition is used in surveillance. In task of surveillance we may do activity tracking and recognition and also provide abnormal detection. Also, can be used in videos for automatic character(Actor) Recognition.

There are two approaches used for face recognition, one is geometric approach and the second is feature based recognition. In geometric based approach one goes on detecting basic features

like nose, eyes, chin and generate face using those features. The problem here is these features are difficult to extract. So, Feature based extraction is used for this application. The basic block diagram of this project is shown in figure 6.

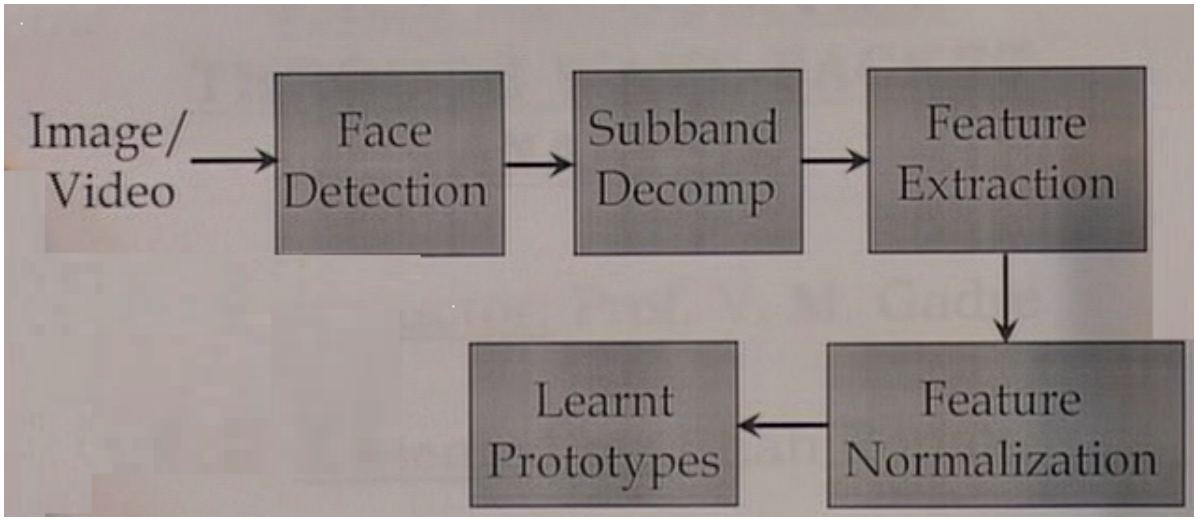


Figure 6: Block diagram for face recognition

In this presentation we are not going to discuss about how face detection is done. After face is extracted we do subband decomposition using wavepacket transform. We already store some features in Learnt prototype and then we do matching with a given image. There are two types of applications based on face recognition. One is content based image retrieval and the other is simple classification of image into different classes. Decomposition of image into subbands is done as shown in figure 7.

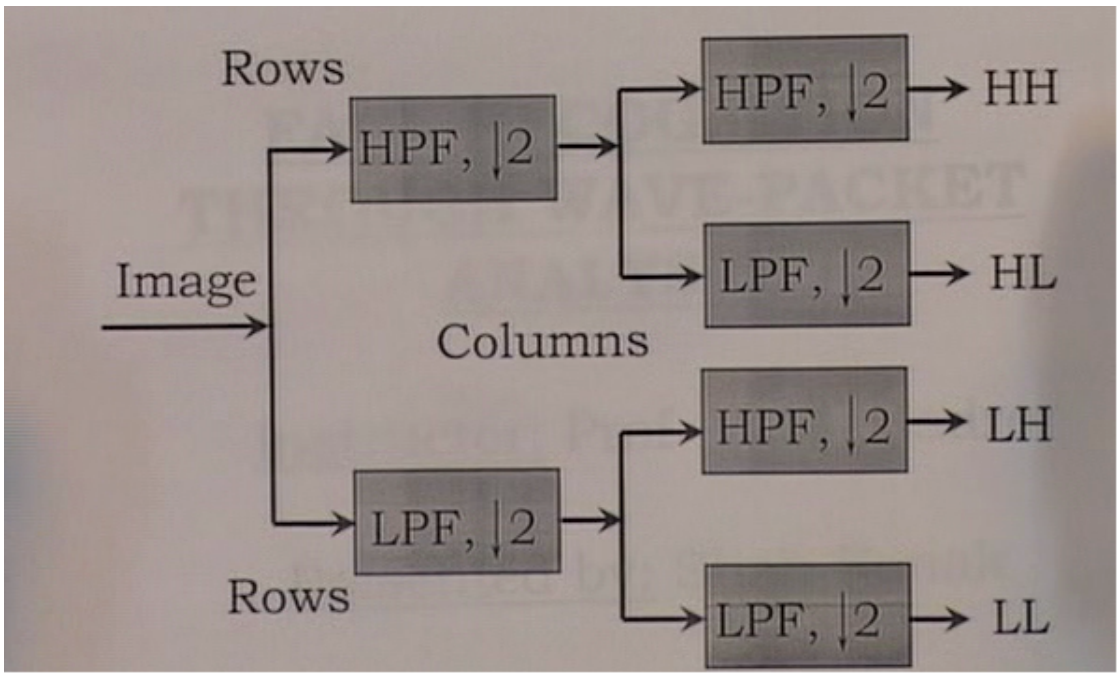
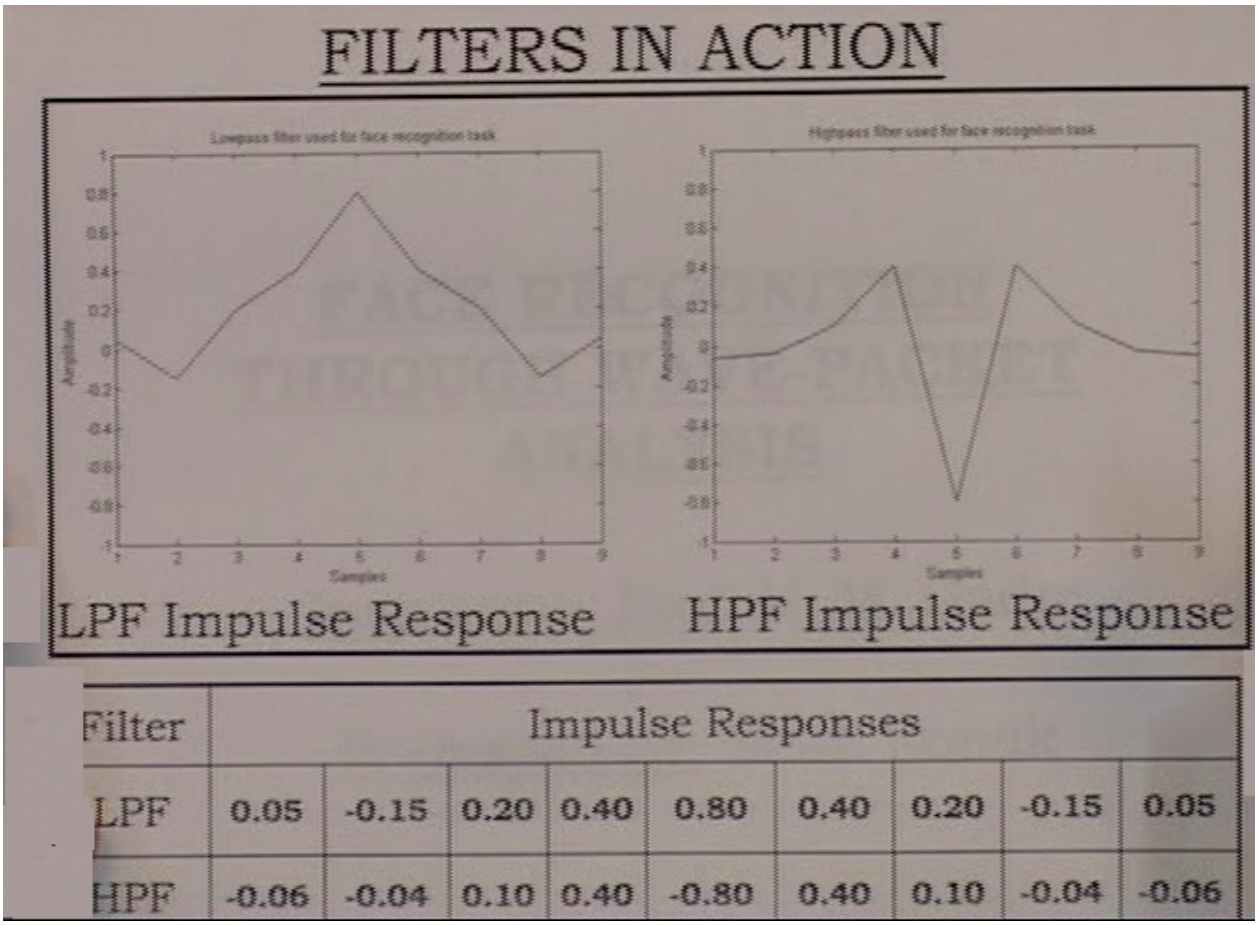


Figure 7: Decomposition of Image into various Sub bands

In Wavelet analysis we just decompose the LL subband. But in Wavepacket analysis we also

decompose the other subbands as well. So, when decompose using Wavepacket transform we get 16 subbands, Where as in wavelets we just get 4 subbands. Here we are not bothered about perfect reconstruction so our analysis filters need to be good. We need filters which have good feature extraction capabilities. Filters used for this application are shown in figure 8.



Filter	Impulse Responses								
LPF	0.05	-0.15	0.20	0.40	0.80	0.40	0.20	-0.15	0.05
HPF	-0.06	-0.04	0.10	0.40	-0.80	0.40	0.10	-0.04	-0.06

Figure 8: The filters used for this project

The impulse responses of both LPF and HPF are also shown in the figure. We decompose the image upto two levels. We don't really need to further decompose image because the image becomes smaller and smaller as we decompose into subsequent levels. When we decompose a an approximate subspace we require a LPF and a BPF. Where as, for decomposing a Detail sub space we require a BPF and a HPF. All these filters are separable in nature and can be used for both dimension separately. Altogether we get 16 subspaces after two level decomposition. The Higher frequency features of face such as eyes can be seen when detail subspace is decomposed into its subbands. After all these 16 bands are extracted then we can go for feature extraction. In each of the 16 images there are features required for face recognition. If we go for all the pixels in all 16 images then our feature extraction vector becomes very large. So, we go for moments like first order moment and second order moment in all the 16 images giving rise to 32 features. But for detailed subspaces we have zero mean. So a 17 dimensional feature vector is sufficient. But, the approximate and detail subspaces are different. Detail subspace have more information compared to approximate subspace. The feature vectors which are extracted are shown in figure 9.
We had two boxes in approximate subspace and in each box we can extract mean and variance for each box. For Detail subspace we extract variance and altogether we get 19 feature vectors.

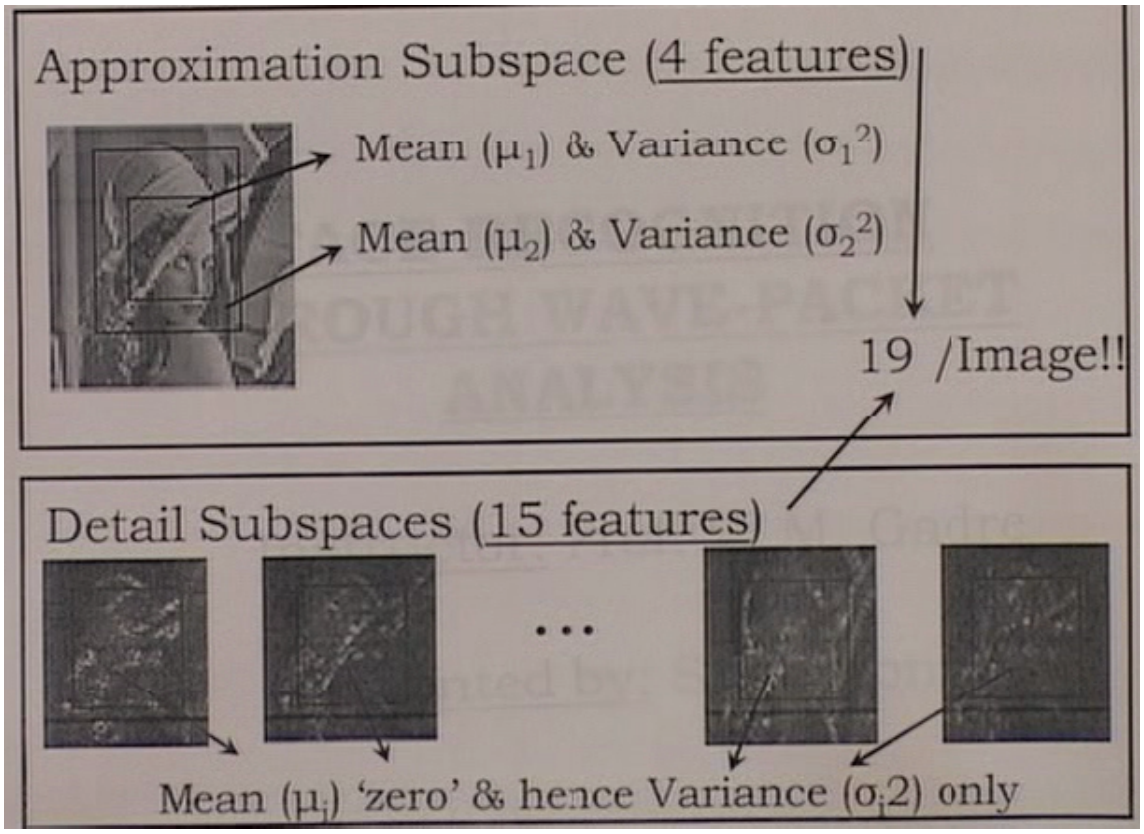


Figure 9: Feature vectors for different subspaces

We compare these features with already stored feature vectors. We use Distance metric for matching. The distance is calculated using Bhattacharya distance for all the feature vectors extracted. The experimental results obtained are shown in figure 10.
These are the two Applications discussed in this lecture.

Exp	Number of Images used in Learning/ class	Total Images used in learning	Total number of query images	Number of images matched to the native class	Accuracy (±1.25%)
1	4	40	80	66	82.5%
2	6	60	80	64	80.0%
3	8	80	80	64	80.0%

Figure 10: Various experimental results