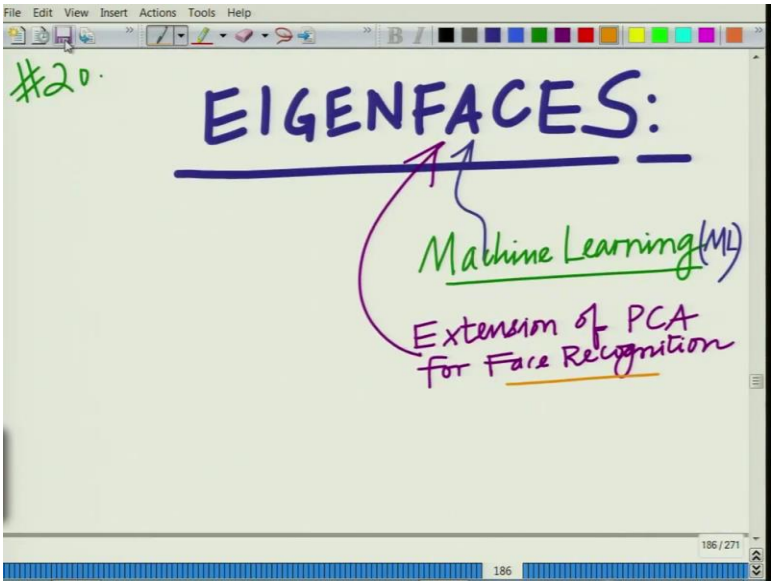
Applied Linear Algebra for Signal Processing, Data Analytics and Machine Learning
Professor. Aditya K Jagannatham
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Lecture No. 20
Computer Vision Application: Face Recognition, Eigen faces

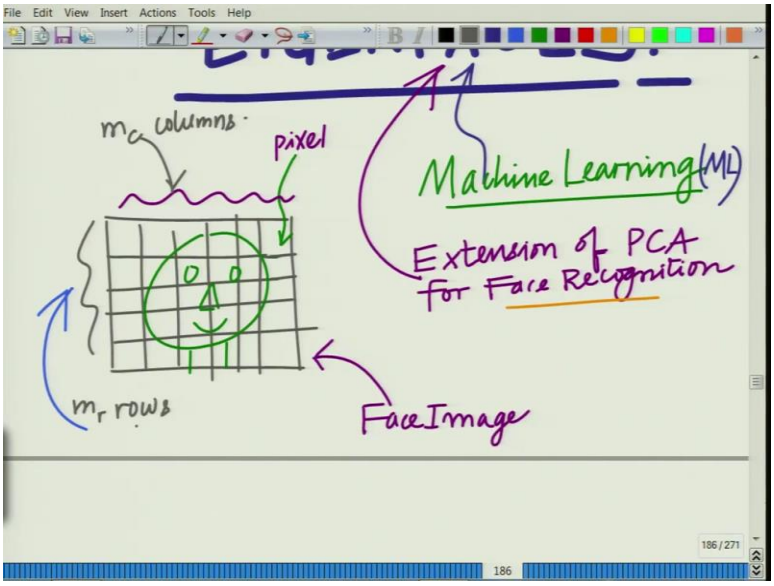
Hello, welcome to the another module in this massive open online course. So, in this module let us look at another interesting application of linear algebra in the context of machine learning this is basically termed as Eigen faces.

(Refer Slide Time: 0:28)



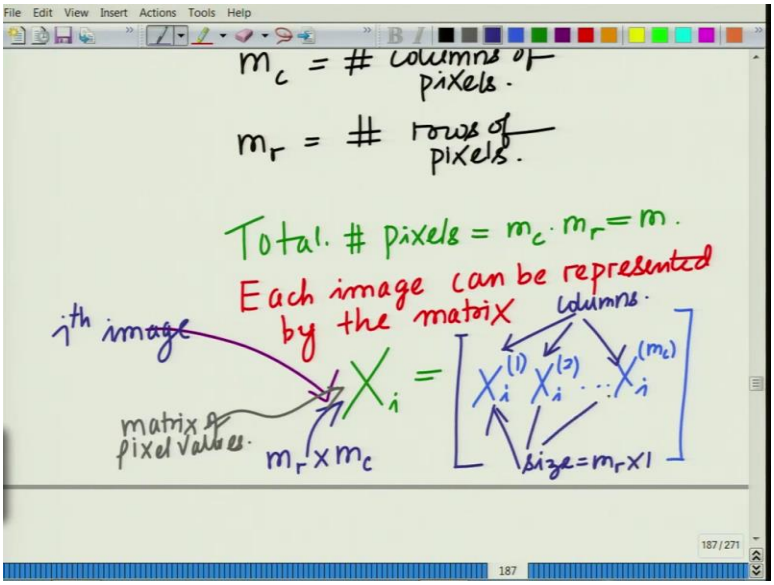
So, let us look at another application this is essentially you can think of this as an extension of PCA to face recognition. So, this is a very popular and significant algorithm. This is an application in the context of as I already told you machine learning or essentially what is the abbreviated as ML. And this is basically you can think of this as extension of PCA for face recognition applications. So, this is an application specifically in the context of face recognition this is termed as Eigen faces. And what do we mean by this Eigen faces.

(Refer Slide Time: 1:49)



What is the procedure for this Eigen faces algorithm? Let us consider a typical face image for instance one can say a typical I am drawing a carriage here, a typical face image comprises of pixels you can think of this as basically comprising of so this is a image of my cartoon image of a face. And this comprises of essentially each of these is a pixel. So, this is basically a face image, a frontal face image and let us say there are mc rows of pixels. Let us say there are mc rows of pixels. And there are, there are mc columns and there are mr rows of pixels in this facial image.

(Refer Slide Time: 3:23)

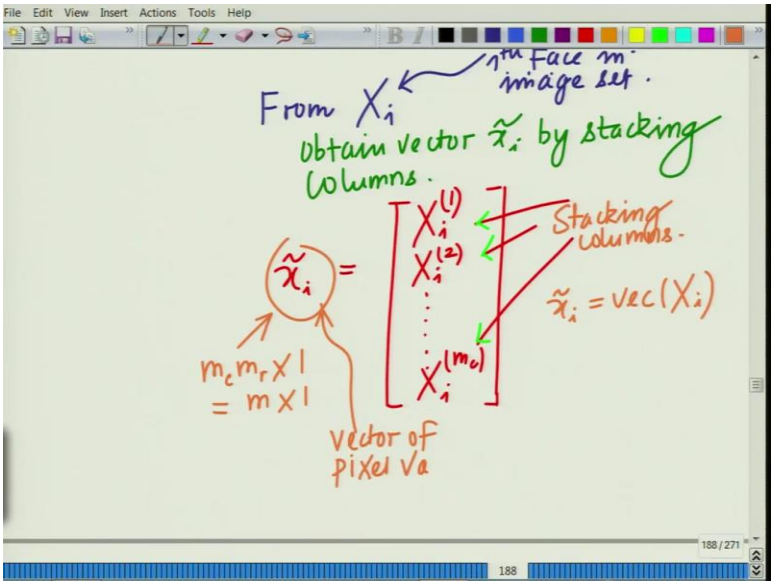


So, you have in the face image mc equal to number of columns of pixels mr equal to number of rows of pixels. And therefore, total number of pixel this is given as mc times mr. mc into mr let us call this as m, m is the total number of pixels. So, now each image xi each image now you see each image can be represented by the matrix which contains a pixel value. So, each image can be represented by the matrix. So, the ith image can represented by a matrix Xi, this is essentially the ith image. This contains of size as I told you mr cross mc this will be of size mr. Where mr is a number of rows mc is the number of columns mr cross mc so each image xi is essentially a matrix.

So, an image is essentially a 2 dimensional signal which contains certain number of rows, certain number of column which contain basically the pixel values. So, let us see these columns are xi1, xi2, xi mc these are the columns. So, this is the ith image xi xc this is a matrix of pixel value. This is mr cross mc and it must be clear but let me write it this is the matrix an image is a matrix of pixel values. And if you look at this, these are the columns. Columns of pixel values there are mc column size of each column mr cross 1. So, each column in this is basically of size mr cross 1.

So, if you look at in image what we are saying is let us take a simple example let say we have a 256 cross 256 image is basically has 256 rows and 256 columns which means essentially matrix of say 256 cross 256 it has 256 rows and 256 columns of basically pixel values. So,, you can arrange this as a matrix comprising of 256 columns each column has 256 elements. The reason we are doing is because now we want to form a vector.

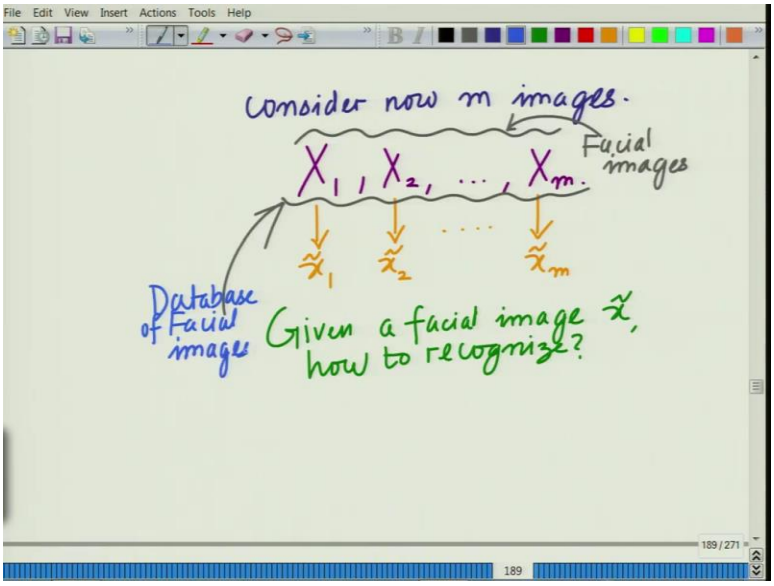
(Refer Slide Time: 7:11)



So, from this xi from xi which is the ith face which corresponds to basically ith face in you think of this as you are training data set. ith face or ith face in your set. ith face in image set. Let us think of it that ith face in the image set. So, you can now from xi obtain vector x tilde i by stacking the columns that is we have x tilde i. You stack the columns of xi one below the other. So, you have xi1, xi2, stack the columns one below the other or one above the other. You have mc columns each is of size m cross 1. So, you are basically stacking the columns.

So, this also be fairly easy to understand this is a standard operation. This is also known as the vec operations so what you are saying is xi tilde and mathematics is also known as the vec operation. So, you take a matrix and stack its columns to obtain a vector. And naturally this x tilde i is going to be it contains basically you have mc columns each of size m. So, this is going to be mc into mr cross 1 vector but mc into mr equal to 1. So, this is going to be mr cross 1 and this is going to be a vector of essentially all remember that is going to be your vector of pixel values.

(Refer Slide Time: 10:11)

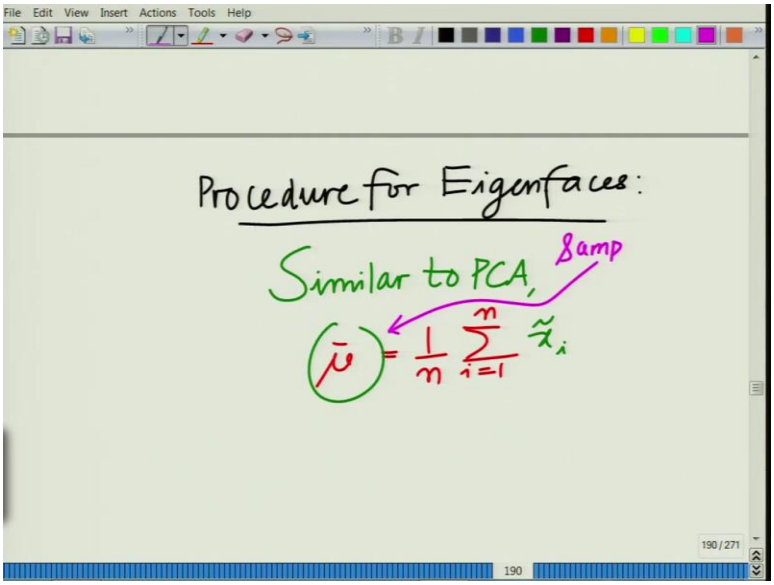


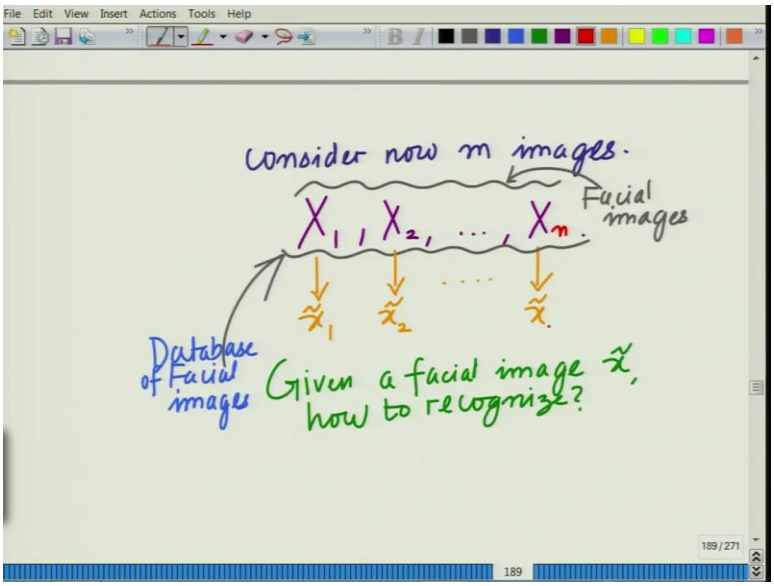
And consider now m images. So, we are going to be so we are obtaining this vector from the image which is essentially a matrix of pixel values. We are obtaining this vector of pixel values and now we are going to illustrate the procedure for face recognition. So, now let us say you have m images in your face set, your face image set or facial image set. So, you have X1, X2, up to Xm and corresponding to this so these are the face images.

So, these are your facial images and corresponding to these you have x1 or x tilde 1, x tilde 2 so on and x tilde m which are the corresponding vectors. And then, now what we do is, how the question we want to ask is given a new image, a new face given or given an image, given a facial image x, given a new facial image x tilde. How to recognize this? That is essentially how to map it to your existing session that is which face does this correspond to? Does it correspond to x1, x2? Which one of the existing face is does this correspond to.

So, we have an existing a vast database. So, let us think you have this database of faces and now you have a new face. Let us say this belongs to some person you want to identify, now to which facial image in your database does this new face correspond to and see that clearly this has a lot of applications and lot of whenever you want to identifies some person. The, you want to identify an unknown person from an image of his or her face you call upon this algorithm. So, and this is vast applications you can imagine. So, this is your, you can think of this as essentially or database, your database of facial images.

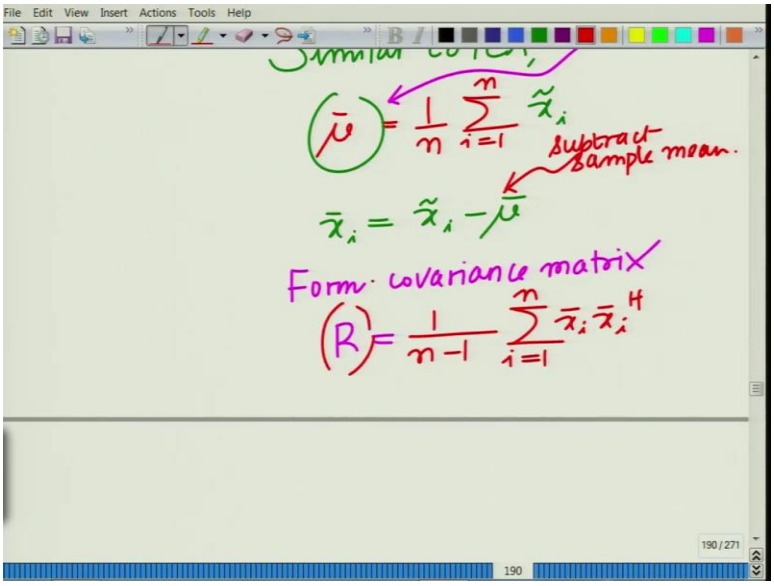
(Refer Slide Time: 13:28)





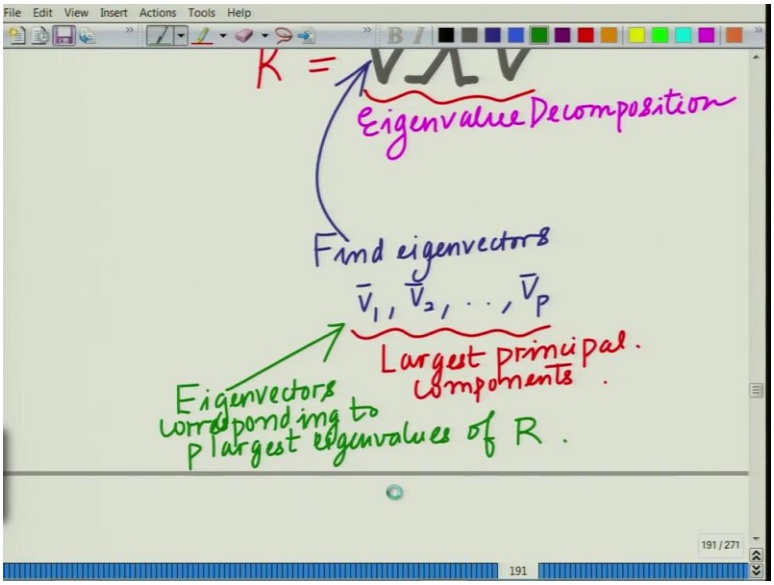
Now the Eigen faces algorithm proceeds as follows remember we are talking specifically about the Eigen faces algorithm. So, we will talk about the procedure for the Eigen faces. So, we will talk about the procedure for Eigen faces as follows. So, what we do is similar to PCA I hope all of you remember the principle component analysis algorithm that we have describe before. You form the mean vector of these facial images. So, you have this 1 over n this should be n. this is the size of your database. So, you have 1 over n summation i equal to 1 to n xi tilde this is the mean you can also remember they are also called as the sample mean.

(Refer Slide Time: 15:04)



And from all the images subtract the sample mean. So, you obtain from each image xi you obtain x bar i equal to x tilde i minus mu bar. So, we subtract the sample mean so we subtract the sample mean and then we from the covariance matrix. Now from the covariance matrix we form the covariance matric rather the estimate of the covariance matrix which is given as R equal to 1 over n minus 1 summation i equal to n xi bar xi bar Hermitian. So, this is basically your covariance this is basically your covariance matrix. So, this is essentially your covariance matrix.

(Refer Slide Time: 16:26)

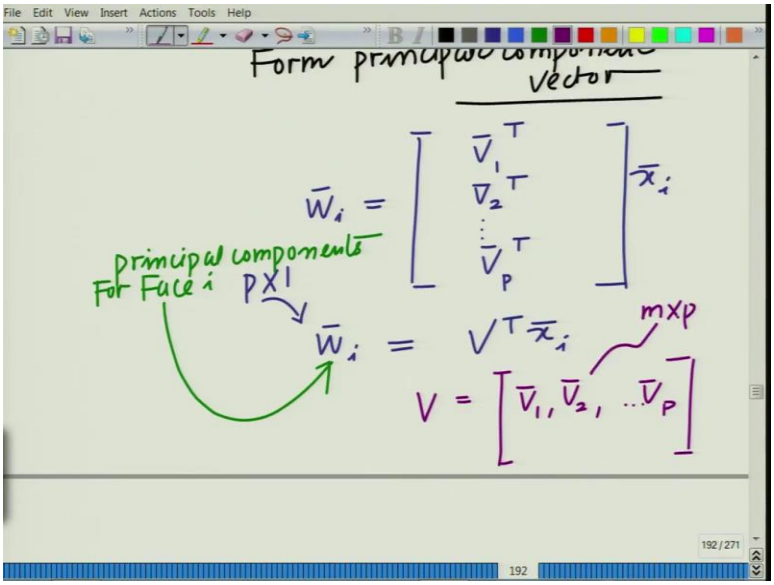


Now once you form the covariance matrix we form the Eigen value decomposition of the covariance matrix and that is given as V you remember the Eigen value decomposition that is given as V lambda V transpose remember this is an Eigen faces algorithm. So, there is some relation to the Eigen values and Eigen vectors in fact what we are going to see the Eigen faces algorithm as the name implies has a very deep relation to the Eigen values and Eigen faces.

And what is the relation this is the relation. So, we talk about the Eigen values of this covariance Eigen. So, this is basically the other covariance matrix and this is basically the Eigen value decomposition. We have the covariance matrix this is basically our Eigen value decomposition. And from this essentially we do ease we find the Eigen vectors largest principals of, we find the Eigen vectors remember similar to PCA find Eigen vectors v1 bar, v2 bar, vp bar this correspond to the largest principal components.

Or you can think of this as the direction corresponding to the are just principal components. You find basically these are the Eigen values corresponding to the P largest Eigen vector or these are the Eigen vectors corresponding to the P largest Eigen values of the covariance matrix. These are Eigen vectors corresponding to P largest Eigen values. P largest Eigen values of R which is the Eigen vectors corresponding to the P largest Eigen values of R which is basically your covariance matrix.

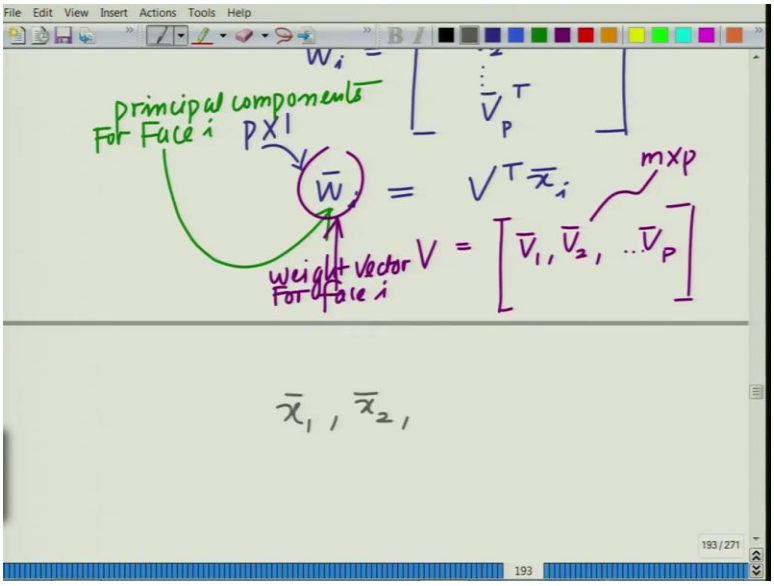
(Refer Slide Time: 19:14)



And now what we do is we form the principal component vector for each image x bar i form the principal component vector. And we know this is given basically by taking the objection of each facial image x bar i along the principal direction along the principal components. So, essentially what we do is now we have the w bar i which is basically given as you take the projection of v1 bar this is your v1 bar transpose, v2 bar transpose, vp bar transpose times your x bar.

Or essentially we are saying this is your vt transpose x bar i and this is your w bar and this will be a P cross 1 vector. And these are the principal components corresponding to image principal components for you can think of this principal components for the face i, principal components for the ith face. These are the principal components for these are the principal components corresponding to face i and remember this matrix v is v equals again v1 bar, v2 bar, vp bar which you can clearly see each vector is a size of m. So, this will be m cross P matrix.

(Refer Slide Time: 22:05)

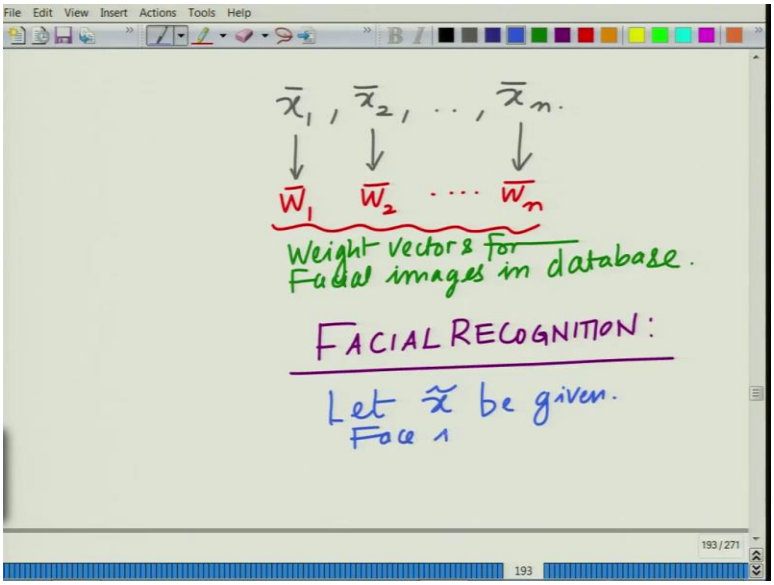


And now so this is basically you can call this as the weight vector for image, weight vector of face i. This is the weight vector for face this is the weight vector for face i. So, you can see this is also basically nothing but our presentation of face i, a compact a compressed P dimensional representation remember that is what PCI gives us what PCA does is it determines the principal components corresponding to the projection of each facial of each data vector along the direction which have the largest variance.

So, this is a vector containing the P you can think of it as the vector correspond containing the principal components of the face i or the significant components or the largest or the components of the face i that has the largest variance. This is a compressed version of that rather than storing that and using that large dimensional that m dimensional vector remember each face image if you think about it 256 cross 256 image face image of pixels.

So, if you represent that as a vector that will be 256 times 256 which essentially 2 to the power of 16. So, that is a large vector we are taking the significant directions of the significant components of that along the Eigen vectors of the covariance matrix corresponding to the largest Eigen values calling that as principal components and this is basically compressed representation of that facial image.

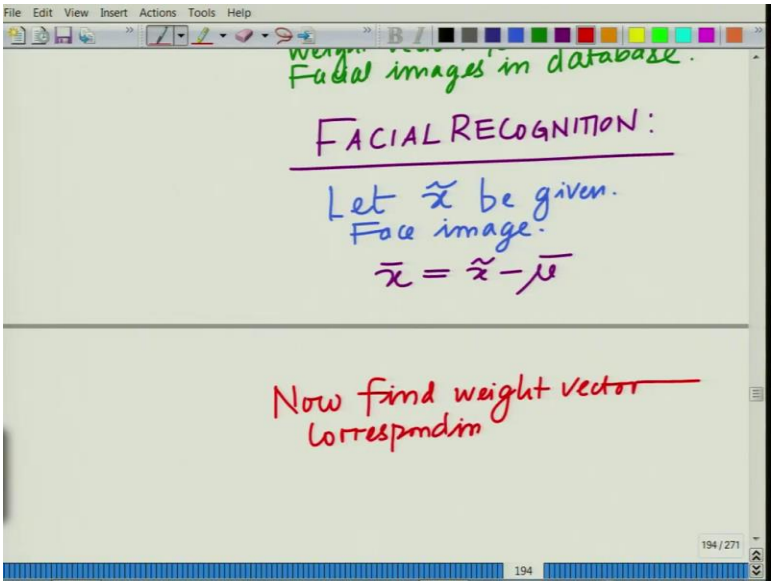
(Refer Slide Time: 24:08)



And now we similarly form so now we similarly form from your x1 bar, x2 bar for these images in your data set we form the weight vectors we form the weight vectors w1 bar, w2 bar, w2 bar these are the weight vectors corresponding to the images, facial images in that database. Weight vectors or facial images in the, these are the weight vectors for the facial images in the database. And now how to perform the facial, so these are the weight vectors for the n images.

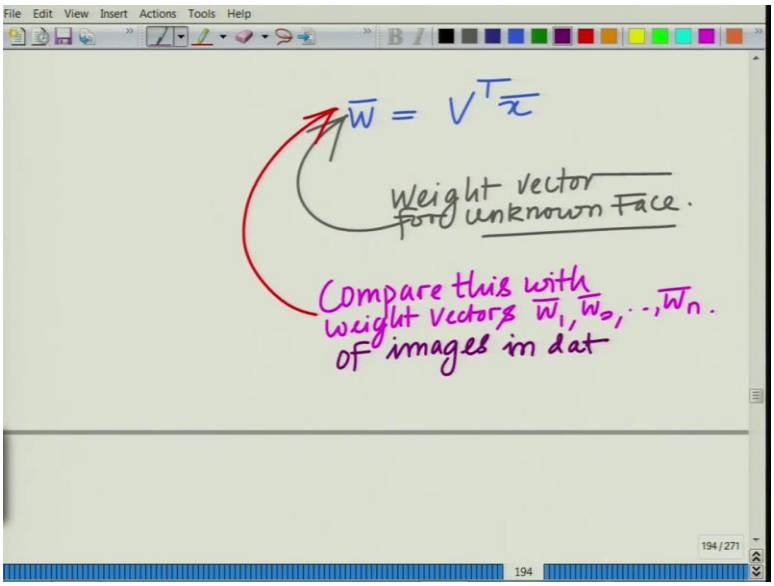
In your database you have n images and now what is the procedure for how do we perform facial recognition. As we say let x tilde i be the new face that is not yet identified. X tilde i be the given face image.

(Refer Slide Time: 25:46)



Corresponding to x tilde i we subtract the mean. Now from x tilde i we obtain x bar i by x from x tilde we obtain x bar as x tilde minus mu bar that we subtract the mean.

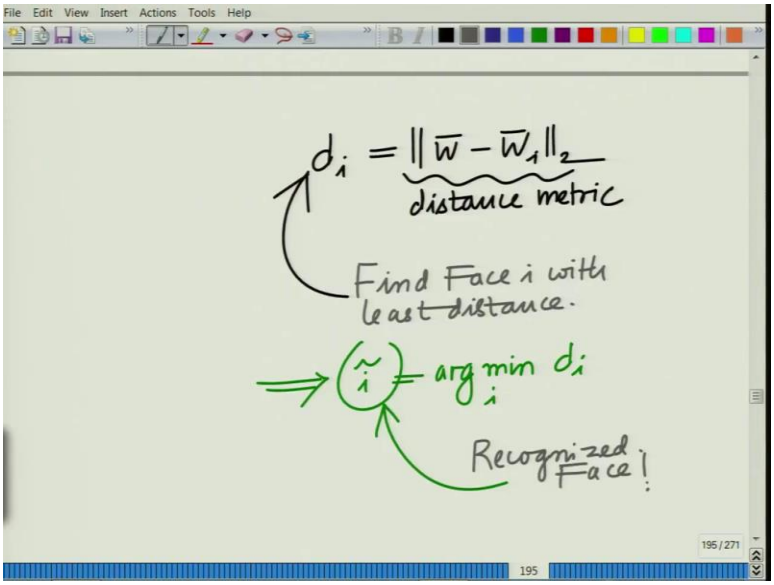
(Refer Slide Time: 26:35)

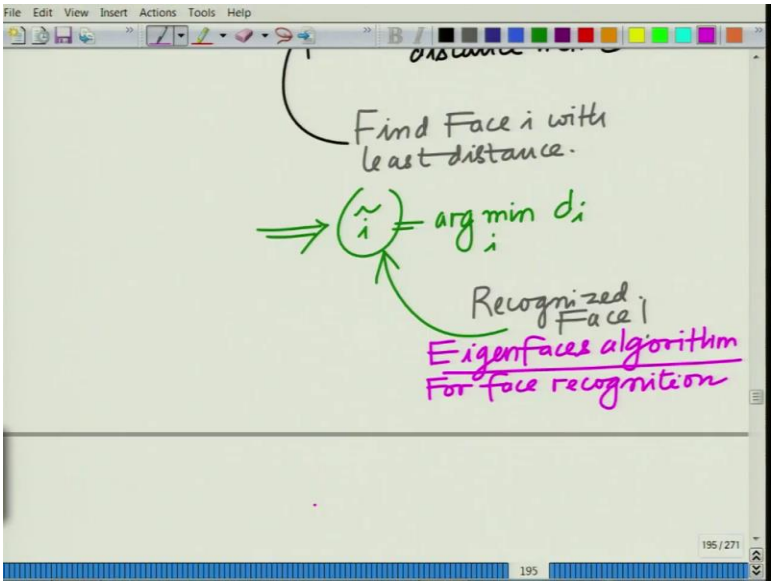


Once again now find the weight vector corresponding to x bar that will be given as w bar which is V transpose times x bar. So, you are finding the principal components corresponding to x bar and this is the weight vector for the unknown face. So, this is the weight vector for the unknown face. This is the vector corresponding to the unknown face and now compare this with the weight vector so of the images, compare this with the weight vector of the images in your database.

Compare this w bar n for images in the database. And the face image from the database with the closest weight vector w bar is the recognized image essentially that is the idea.

(Refer Slide Time: 28:25)





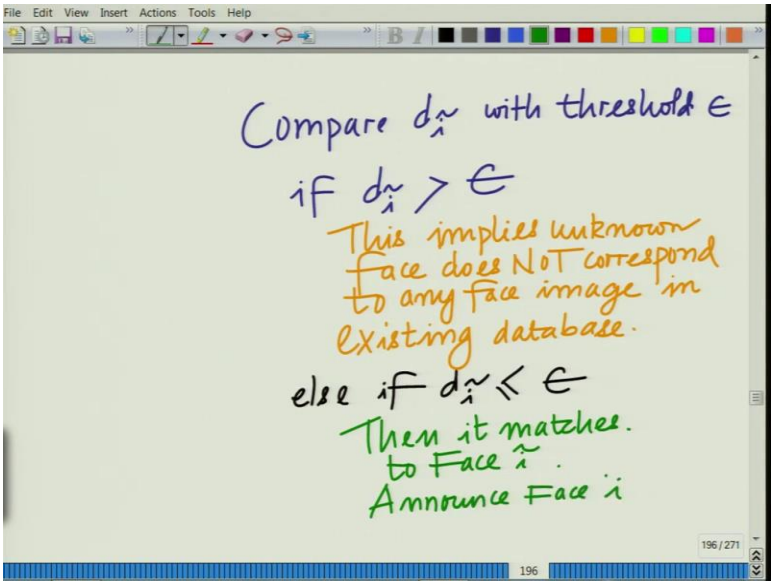
So, you compare it, so you form the distance di of the weight vector of this unknown face to each vector in the, so this is the distance. This is the l2 norm which is essentially the distance or you can think of this as the Euclidean distance this is your essentially or distance matrix or this is essentially the distance of the weight vector w bar i from weight of each image i in the database.

And then find the least, find the image i we find the face image i with the least distance implies find i tilde equals arg min which is essentially the arg minimum argument of di that is the

distance di the i such that the distance di is minimum and i tilde this is essentially your recognized face. This i tilde which basically corresponds to the minimum distance that is the weight vector which has the minimum distance weight vector i w bar i which has the minimum distance to the weight vector w bar of this new facial image that is essentially the recognized image. So, this is your Eigen faces algorithm for face recognition.

So, this is essentially basically your whole thing that we described so far is basically what we call as the Eigen faces. Eigen faces algorithm this is basically the Eigen faces algorithm for face recognition. The only small change here is going to be the fact that see sometimes this image this new image of a new face and practical, practice that might also arise frequently that is this facial image of this unidentified person might not actually be present in your database. So, it might not correspond to any facial image in your database.

(Refer Slide Time: 31:15)



So, typically we compare this distance with a threshold. So, di, so compare di or this di tilde rather that is the minimum di that you found di tilde with some threshold epsilon. If this minimum distance di tilde is greater than epsilon, then it essentially means that minimum distance that has no face in the database is a close match. Then it does not match with any of the facial images in the dataset. So, this typically means that this means this implies that the unknown face, this implies that the unknown image does not correspond to any face image in

existing database. It does not correspond to any unknown image because it is not particularly close to any of the images in the database.

On the other hand, if this di tilde is less than equal to epsilon else if di tilde is less than epsilon, then this e matches to face i tilde. Then it matches to face i tilde. So, this match, so essentially what you do at this point is you announce face i tilde from the dataset as the closest facial image that is unknown image you announce face i tilde. Essentially, you announce that i tilde matches to that face i tilde or essentially what you can do is instead of taking the in fact there can be variations of this instead of taking the facial.

If you have a large database sometimes what might happen is that it might closely match with the images of several faces. So, you might not just take the i tilde which has the least distance but you might take a certain number let say 3 or 4 or a certain number k of images which have the lowest distance corresponding to weight vector of this new unidentified face. And then you do a manual search or you probably do a more an expert search, you call in a next part of something of that sort and then try to better identify or more carefully identify which of this faces in the database does this space correspond to or does it not in fact correspond to any face at all.

So, these are all variations of this algorithm. So, essential idea here is that you are taking this large database which potentially comprises of a large number of facial images and more over each facial image itself we consider the modest resolution of 256 cross 256. This facial high resolution facial images now a days because of the improvement in the resolution of the cameras this can be 512 cross 512 or 1024 cross 1024 and so on which means the size of each vector can be huge.

So, you have a large database of faces each data point has a huge size and now how do you take a new facial image and essentially announce compare it or essentially check with which image, facial image in your dataset this image, this new facial image matches or which the image in your dataset to which this new image corresponds. So, this is the Eigen faces algorithm which can be used which is essentially a machine learning application of the principals of Eigen values and Eigen vectors that we have learned so far.

It is an interesting application of and essentially PCA principal components analysis. Essentially a very interesting application of I would say linear algebra and PCA in a practical context of,

practical example of facial recognition. So, with this let us stop here and we will continue with other such interesting applications as well as new concepts in the subsequent modules. Thank you very much.