Computer Vision and Image Processing - Fundamentals and Applications

Professor Dr. M.K. Bhuyan
Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati, India

Lecture 12

Image Transforms

Welcome to the NPTEL MOOCs course on Computer Vision and Image Processing – Fundamentals and Application. In my last class I discussed the concept of image transformation, the transformation converts the spatial domain information into frequency domain information. Mainly, it is the mapping from spatial domain into frequency domain. The transformation does not sync the information content present in a signal and transformation is quite useful for compact representation of data.

And also, I discussed the concept of orthogonal transformation. So, in the orthogonal transformation, I define the transformation matrix that is T inverse is equal to T transpose then, in this case, it is called the orthogonal matrix and the transformation will be orthogonal transformation. And if I consider the T the transformation matrix is a complex matrix then T inverse is equal to T complex conjugate transpose that is called unitary matrix and the corresponding transformation is called the unitary transformation.

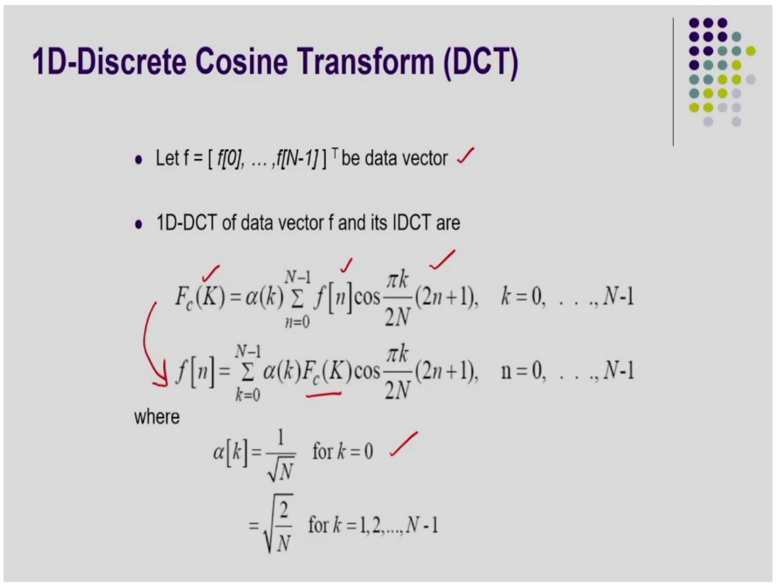
And I also highlighted that DFT the discrete Fourier transform that is not unitary. So, I can make DFT unitary by defining the DFT, so that I am getting the unitary DFT transformation. After this, I discussed the properties of unitary transformation. So, three main properties I want to highlight again1 is energy preserving property. So, in the energy preserving property if I do the transformation from the spatial domain into frequency domain, so I will be getting transform coefficients.

So, most of the energy is available in few transform coefficients that is called the energy compaction property and one is the energy preserving property that is mainly the energy into data domain is equal to energy in the frequency domain. So, that is called the energy preserving property, energy in the data domain is equal to energy in the transform domain. Also, I discussed another important property that is that Decorrelating property.

So, my original data is highly correlated and after the transformation, the transform data will be less correlated. So, for this, I define the covariance matrix. So, based on the property of the covariance matrix, I can define the decorrelating property. Last class, I define the DFT, the DFT unitary matrix. Now, today I will discuss one transformation that is called the DCT the discrete cosine transform and after this, I will discuss another transformation that transformation is KL transformation.

The DCT transformation is an orthogonal transformation, orthogonal means the T inverse is equal to T transpose that is the orthogonal matrix and the transformation is the orthogonal transformation. So, let us see what is the DCT, the discrete cosine transform.

(Refer Slide Time: 03:47)



1D-Discrete Cosine Transform (DCT)

- Let f = [f[0], … ,f[N-1]]^T be data vector ✓
- 1D-DCT of data vector f and its IDCT are

$$F_c(K) = \alpha(k)\sum_{n=0}^{N-1} f[n]\cos\frac{\pi k}{2N}(2n+1), \quad k = 0, \ldots, N\text{-}1$$

$$f[n] = \sum_{k=0}^{N-1} \alpha(k)F_c(K)\cos\frac{\pi k}{2N}(2n+1), \quad n = 0, \ldots, N\text{-}1$$

where

$$\alpha[k] = \frac{1}{\sqrt{N}} \quad \text{for } k = 0$$

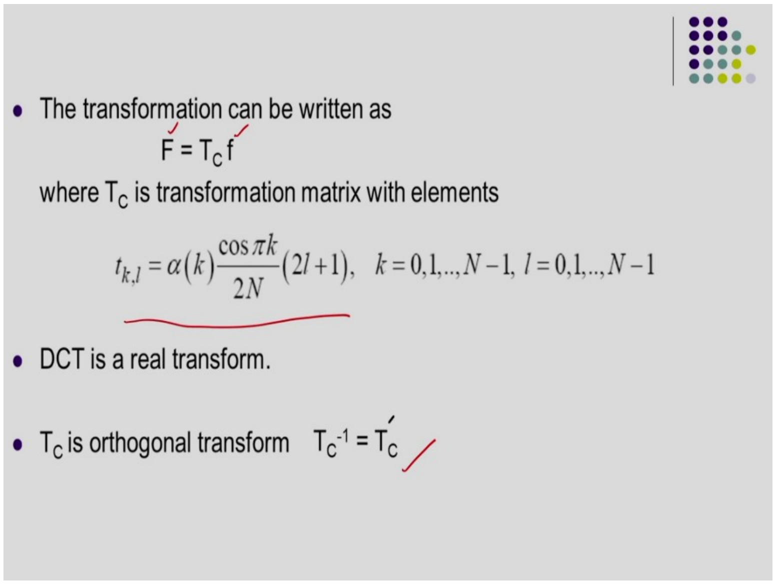
$$= \sqrt{\frac{2}{N}} \quad \text{for } k = 1,2,\ldots,N\text{-}1$$

So, first I am considering 1D-Discrete Cosine Transform here. So, I am considering the data vector, the data vector is F I am considering, so it is f 0, f1 like this f (N – 1), and I am considering the transpose because column vector I am considering. After this, the DCT can be defined like this.

So, DCT F $_c$ (k), the DCT of the input sequence, the input sequences f (n) is defined by this equation, here you can see the F c k is equal to alpha k n is equal to 0 to N minus 1 f n cos pi k divided by twice n twice n plus 1. So, in DCT this cos pi k divided by twice n twice n plus 1 that is the transformation kernel and I can also get the reconstructed data that is the inverse transformation, the inverse DCT I can determine, this is the inverse DCT.

So, F $_c$ (k) from F $_c$ (k) I am determining f (n), f (n) is the original data vector that I can determine by inverse discrete cosine transform. And in this case, I can define alpha k, alpha k is equal to 1 by root N corresponding to k is equal to 0 and it is equal to root 2 by N for k is equal to 1, 2, to N - 1. So, this is the definition of 1D-Discrete Cosine Transformation

(Refer Slide Time: 05:17)



- The transformation can be written as

  $F = T_C f$

  where T_C is transformation matrix with elements

  $$t_{k,l} = \alpha(k)\frac{\cos \pi k}{2N}(2l+1), \quad k = 0,1,..,N-1, \ l = 0,1,..,N-1$$

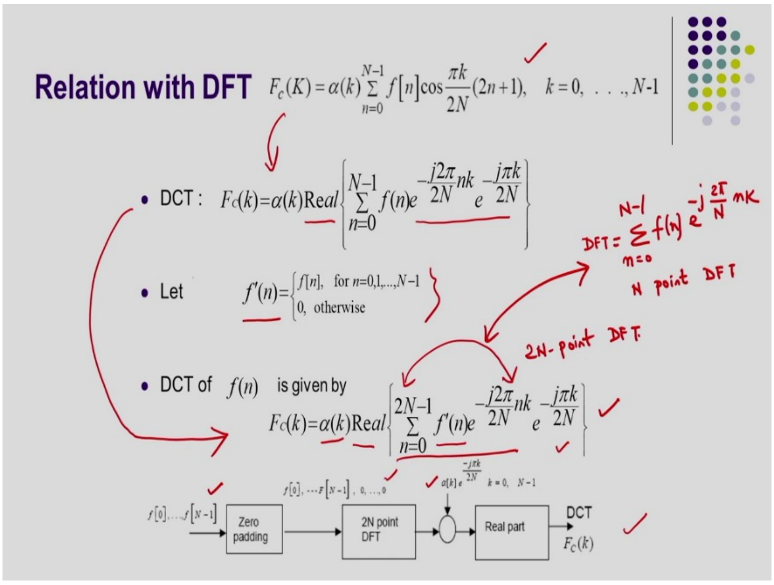
- DCT is a real transform.
- T_C is orthogonal transform $T_C^{-1} = T_C'$

And this transformation I can write like this. So, F is the DCT that is the transformed data and T C is the transformation matrix corresponding to Discrete Cosine Transform and f is the original data. And in this case, the transformation matrix T C and corresponding elements of the transformation matrix I can write like this. So, t k l the elements of the transformation matrix, so t k l is equal to alpha k cos pi k divided by twice N twice l plus 1.

So, this is the elements of the transformation matrix and in this case, the transformation matrix is real, because I am considering cos pi k divided by twice N. So, that means, in this case, I am having the real values. So, that is why the DCT is a real transform but in case of the Fourier transform I have the both cosine and the sine component, the real component, and the imaginary component, so that is why it is complex.

But in case of the DCT I have only the cos component, the cos pi k divided by twice N twice l plus 1. So, that is why the DCT is a real transformation and the DCT is an orthogonal transformation. So, here you see, the T c inverse is equal to T c transpose. So, that is why it is the orthogonal transformation.

(Refer Slide Time: 06:44)



Relation with DFT

$$F_c(K) = \alpha(k)\sum_{n=0}^{N-1} f[n]\cos\frac{\pi k}{2N}(2n+1), \quad k = 0, \ldots, N\text{-}1$$

- DCT : $F_c(k)=\alpha(k)\text{Real}\left\{\sum_{n=0}^{N-1} f(n)e^{-\frac{j2\pi}{2N}nk}e^{-\frac{j\pi k}{2N}}\right\}$
- Let $f'(n)=\begin{cases} f[n], & \text{for } n=0,1,\ldots,N-1 \\ 0, & \text{otherwise}\end{cases}$
- DCT of $f(n)$ is given by $F_c(k)=\alpha(k)\text{Real}\left\{\sum_{n=0}^{2N-1} f'(n)e^{-\frac{j2\pi}{2N}nk}e^{-\frac{j\pi k}{2N}}\right\}$

$$\text{DFT} = \sum_{n=0}^{N-1} f(n)e^{-j\frac{2\pi}{N}nk}$$

N point DFT

2N- point DFT

$$\begin{tikzcd} f[0],\ldots,f[N-1] \arrow[r] & \boxed{\text{Zero padding}} \arrow[r, "{f[0],\ldots,f[N-1],0,\ldots,0}"] & \boxed{\text{2N point DFT}} \arrow[r] & \bigcirc \arrow[r] & \boxed{\text{Real part}} \arrow[r] & \text{DCT } F_c(k) \\ & & & \alpha[k]e^{\frac{-j\pi k}{2N}}\ k=0,\ N-1 \arrow[u] & & \end{tikzcd}$$

Now, I want to show the relationship between the DCT and the DFT. So, in my first equation, you can see here, this is the definition of the DCT F c k is equal to alpha k, f n is the input data sequence n is equal to from zero to N minus 1, f n cos pi k divided by twice N twice n plus 1, so this is my F c k, F c k is the DCT.

Now, this equation I can write in this form you can see, because, already I have explained that in case of the DCT we have only the cos term there is no sine term that is, there is no imaginary component. So, that is why I am considering only the real part of the exponential function, this exponential function I am considering.

So, I am considering a Real that means, the cos term I am considering then neglecting the imaginary term that is a sine term and after this, I am considering another sequence. The new sequence I am considering is f dash n. So, what is f dash n? f dash n is equal to f n corresponding to n equal to 0 to N minus 1, and otherwise it is 0. So, I am defining a new sequence, the new sequence is f dash n.

After this, this DCT expression, I can represent in this form, you can see F c k Real and, in this case, I am considering n is equal to 0 to twice N minus 1 because in this case, you can see this 0, that means I am doing the zero padding, zero padding up to twice N minus 1. So, f n, f dash n is equal to f n for n is equal to 0 to N minus 1 and 0 up to twice N minus 1. So, what will be the length of my sequence?

The length of a sequence is from 0 to twice N minus 1 that means, it is a twice N point sequence. So, in this case for corresponding to the new sequence, the new sequence is f dash n, I can write this expression, the expression for the DCT. So, if you see this expression and you can see this term if you see these terms from this to this because you know this expression.

Suppose, if I consider the DFT, DFT is n is equal to 0 to N minus 1, f n and e to the power minus j twice pi divided by N n k, this is the definition of N point DFT. So, if you compare this one and this one then it is nothing but it is the twice N point DFT, that expression is nothing but it is a twice N point DFT. So, from this expression, you can see the relationship between DCT and the DFT.

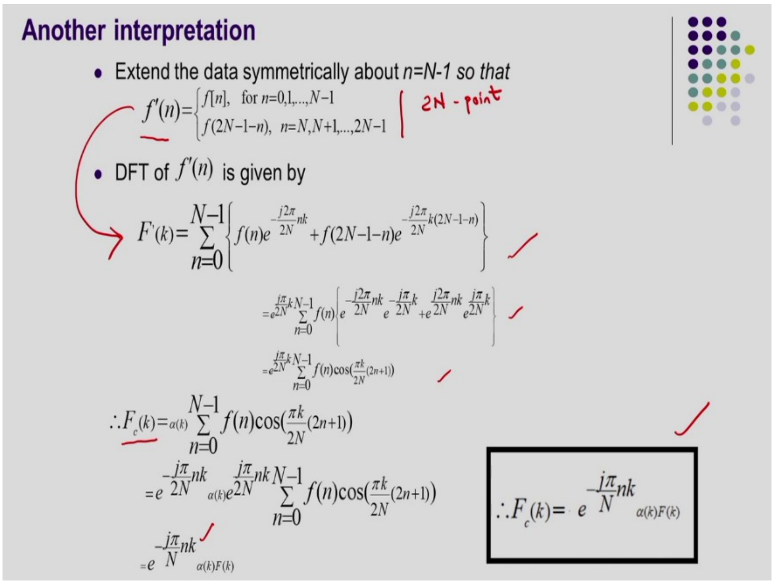
So, how to get the DCT from the DFT? If you see this block diagram, you can see the input data sequence is f naught, f1, up to f N minus 1 this is my input data sequence. After this I want to get a new data sequence, the new data sequence is f dash n. So, for this, I have to do zero padding, so I am doing zero padding. So, what will be the length of the new sequence?

The length of the new sequence will be twice N, so it is from zero to twice N minus 1 and after this, I am computing the twice N point DFT. So, that means I am completing this part twice N point DFT. After this you can see my I am doing the multiplication by alpha k and also by this e to the power minus j pi k divided by twice N. So, that is why I am doing multiplication by alpha k e to the power minus j pi k divided by twice N.

And after this, I am considering one operation that operation is the Real. So, I am considering the real part of this and I am getting the DCT, the DCT of that sequence. So, you can see that, you can obtain the DCT from the DFT by this process. So, first I have to do the zero padding, so that the length of the sequence will be twice N and after this, I have to calculate the twice N point DFT.

And after this I have to do multiplication, the multiplication is alpha k e to the power minus j pi k divided by twice N and after this, I am doing one operation that is the real part I am considering and I am getting that DCT. So, that means the DCT can be computed by using the DFT.

(Refer Slide Time: 11:20)



Another interpretation

- Extend the data symmetrically about *n=N-1 so that*

$$f'(n)=\begin{cases} f[n], & \text{for } n=0,1,\ldots,N-1 \\ f(2N-1-n), & n=N,N+1,\ldots,2N-1 \end{cases} \quad \Big| \ 2N\text{-point}$$

- DFT of $f'(n)$ is given by

$$F'(k)=\sum_{n=0}^{N-1}\left\{f(n)e^{-\frac{j2\pi}{2N}nk}+f(2N-1-n)e^{-\frac{j2\pi}{2N}k(2N-1-n)}\right\} \checkmark$$

$$=e^{\frac{j\pi}{2N}k}\sum_{n=0}^{N-1}f(n)\left\{e^{-\frac{j2\pi}{2N}nk}e^{-\frac{j\pi}{2N}k}+e^{\frac{j2\pi}{2N}nk}e^{\frac{j\pi}{2N}k}\right\} \checkmark$$

$$=e^{\frac{j\pi}{2N}k}\sum_{n=0}^{N-1}f(n)\cos\left(\frac{\pi k}{2N}(2n+1)\right) \checkmark$$

$$\therefore F_c(k)=\alpha(k)\sum_{n=0}^{N-1}f(n)\cos\left(\frac{\pi k}{2N}(2n+1)\right) \checkmark$$

$$=e^{-\frac{j\pi}{2N}nk}\alpha(k)e^{\frac{j\pi}{2N}nk}\sum_{n=0}^{N-1}f(n)\cos\left(\frac{\pi k}{2N}(2n+1)\right)$$

$$=e^{-\frac{j\pi}{N}nk}\alpha(k)F(k) \checkmark$$

$$\boxed{\therefore F_c(k)=e^{-\frac{j\pi}{N}nk}\alpha(k)F(k)}$$

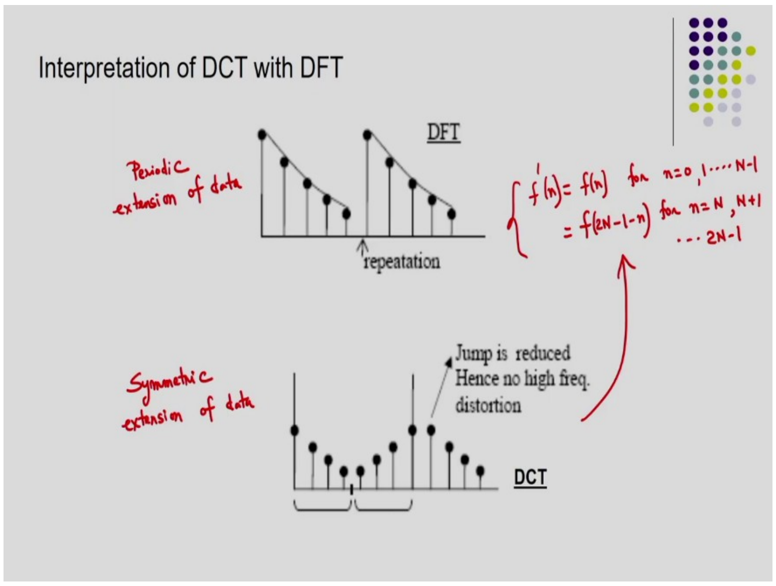
So, in this slide, I am showing another interpretation that is the relationship between the DCT and the DFT. In this case, you can see again I am considering one new sequence, the new sequence is generated from the original sequence, the original sequence is f n. So, this is my new sequence f dash n. So, what is f dash n? f dash n is equal to f n corresponding to n equal to 0,1, up to N minus 1 and it is f twice N minus 1 minus n corresponding to n, n plus 1 up to twice N minus 1.

So, what will be the length of the new sequence? The length of a new sequence is twice N point. After this, I can determine the DFT of this sequence. Thus, you can see the mathematics. After determining the DFT of the new sequence, finally, I am getting the F c k, F c k is nothing but the DCT of the sequence. So, I am getting the DCT of the sequence that is equal to e to the power minus j pi N into n k alpha k into f k.

So, this is I am writing here. So, this is a very important representation. So, how to get the DCT from the DFT? So, what is F c k? F c k is nothing but the DCT of the sequence is equal to e to the power minus j pi N n k alpha k F k. So, you can see. So, by using this approach also you can determine the DCT from the DFT because the DFT already, you know how to compute the DFT.

So, there are many algorithms like FFT also you can use to determine the DFT and by using the DFT you can determine the DCT of a particular sequence. So, for this I have to define the new sequence, the new sequence is f dash n, so this is a new sequence and that is the twice N point sequence. So, by using this method, I can determine the DCT from the DFT.

(Refer Slide Time: 13:37)



Interpretation of DCT with DFT

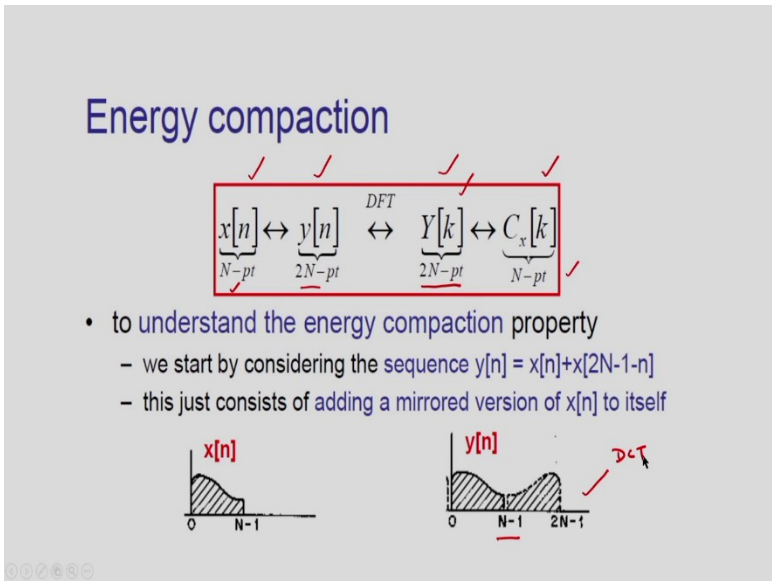
DFT
Periodic extension of data
repeatation
Symmetric extension of data
Jump is reduced
Hence no high freq. distortion
DCT

$$\left\{ \begin{array}{ll} f'(n) = f(n) & \text{for } n = 0, 1, \ldots N-1 \\ = f(2N-1-n) & \text{for } n = N, N+1 \ldots 2N-1 \end{array} \right.$$

Now, if you see this extension, the extension what I have done the extension is something like this f dash n, I am having the new sequence f dash n is equal to f n corresponding to n is equal to 0,1, up to N minus 1 and f is equal to twice N minus 1 minus n for n is equal to N, N and plus 1 up to twice N minus 1.

So, I am getting a new sequence like this, the DFT is nothing but the periodic extension of data. This is the case for the DFT, in case of the DCT if you see the second case, this is nothing but the symmetric extension of data. What I have done here that is a symmetric extension of data. So, in case of the DFT, it is nothing but the periodic extension of data but in case of the DCT I have the symmetric extension of data.

(Refer Slide Time: 15:15)



Energy compaction

$$\boxed{\underbrace{x[n]}_{N-pt} \leftrightarrow \underbrace{y[n]}_{2N-pt} \overset{DFT}{\leftrightarrow} \underbrace{Y[k]}_{2N-pt} \leftrightarrow \underbrace{C_x[k]}_{N-pt}}$$

- to understand the energy compaction property
  - we start by considering the sequence y[n] = x[n]+x[2N-1-n]
  - this just consists of adding a mirrored version of x[n] to itself

Now, in this case, you can see the previous discussion you can see, the input sequence was the N point sequence and after this, I am generating a new sequence that is the twice N point sequence and after this, I am determining that DFT. So, there is a twice N point DFT and from a twice N point DFT I am getting the N point DCT.

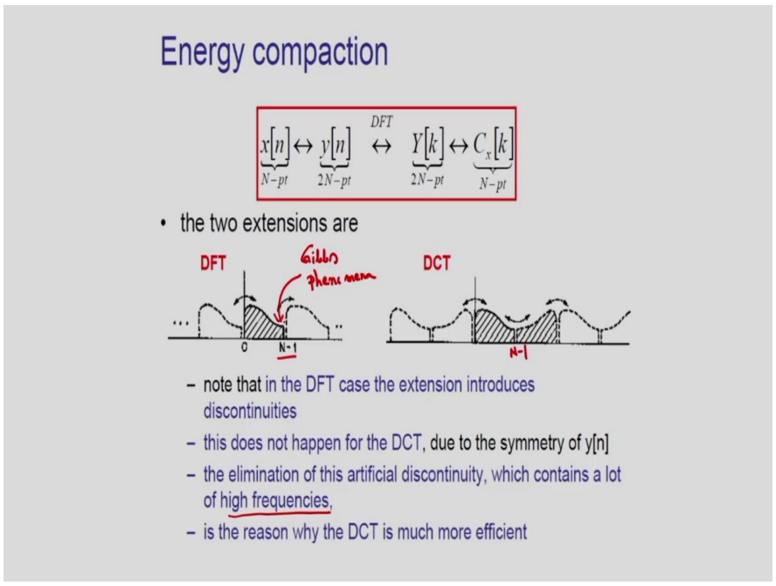
So, what I discussed in my last slides that from the input sequence, the input sequence is the N point sequence after this I am having the new sequence, the new sequence is twice N point sequence and after this, I am determining the twice N point DFT. So, this DFT is the twice N point DFT and from the twice N point DFT I am calculating the N point DCT.

So, you can see, in case of the DFT the N point data is represented by the twice N point DFT but this N point data is represented by N point DCT. So, you can compare the energy compaction property. In the first case in the DFT case, the N point data is represented by the twice N point and DFT that is if I considered a new sequence suppose that is the twice N point data is represented by twice N point DFT.

And in case of the DCT these twice N point data is represented by only N point DCT. So, you can compare the energy compaction property that is the energy compaction of DCT is better than DFT because the twice N point data is represented by twice N point DFT, but in case of DCT, the twice N point data is represented by N point DCT.

So, that is why I can say that DCT is better than DFT for a case, the case is the energy comparison, the energy comparison property. And in case of the DCT, I have already explained this one, this is nothing but the symmetric extension of data around a point the point is N minus 1, this is the case of the DCT I am doing the symmetric extension of data around a point N minus 1.

(Refer Slide Time: 17:39)


DFT
Gibbs phenomena
0
N-1
DCT
N-1

Energy compaction

$$\boxed{\underbrace{x[n]}_{N-pt} \leftrightarrow \underbrace{y[n]}_{2N-pt} \overset{DFT}{\leftrightarrow} \underbrace{Y[k]}_{2N-pt} \leftrightarrow \underbrace{C_x[k]}_{N-pt}}$$

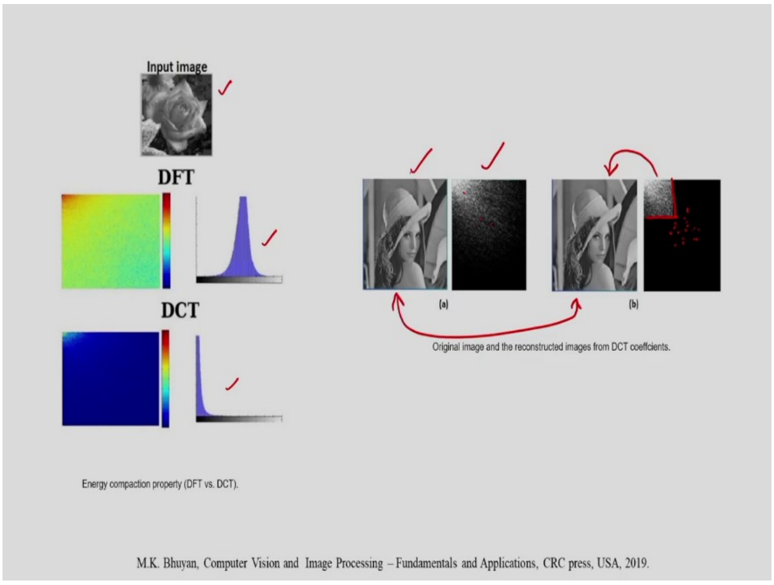
- the two extensions are
  - note that in the DFT case the extension introduces discontinuities
  - this does not happen for the DCT, due to the symmetry of y[n]
  - the elimination of this artificial discontinuity, which contains a lot of high frequencies,
  - is the reason why the DCT is much more efficient

In this case, again I have shown the concept of the DFT and the DCT, DFT is nothing but the periodic extension of data, and DCT you can see it is a symmetric extension of data around a point N minus 1. So, you can see here, in case of the DFT because of the discontinuity, you can see that discontinuity here, discontinuity, this is a discontinuity. So, there is a jump in the discontinuity.

And in this case, we have the high-frequency distortion and this high-frequency distortion is called the Gibbs phenomena, this is a high-frequency distortion, because there is a jump in the extension, the extension is around the point N minus 1. In case of the DCT, you can see the transition is pretty smooth around point N minus 1. So, transition is very smooth around point N minus 1. So, that means the high-frequency distortion will be less in case of the DCT.

So, I have explained in the slide here that one is the high-frequency distortion. And in case of the DCT high-frequency distortion is less because we have smooth transition at the boundaries, around the point N minus 1. But in case of the DFT because of this abrupt change of this value, you can see, that we have the high-frequency distortion and this is called a Gibbs phenomenon.

(Refer Slide Time: 19:24)



Energy compaction property (DFT vs. DCT).

Original image and the reconstructed images from DCT coefficients.

M.K. Bhuyan, Computer Vision and Image Processing – Fundamentals and Applications, CRC press, USA, 2019.

And in this case, you can see the energy compaction property of the DFT and the DCT. In this example, I have shown the input image is this and I am considering the DFT first, you can see the energy compaction here I am showing in the graph. And in case of the DCT you can see this energy compaction, energy compaction means, most of the energy is available in few coefficients.

In DCT, it is better than the DFT and in this second example, in this case, I have shown the image reconstruction after performing the discrete cosine transform. So, in this case, I am considering one image, input image and I am determining the DCT of this image. So, I have the transform coefficients, you can see the DCT coefficients here, all the DCT coefficients. These are the DCT coefficients.

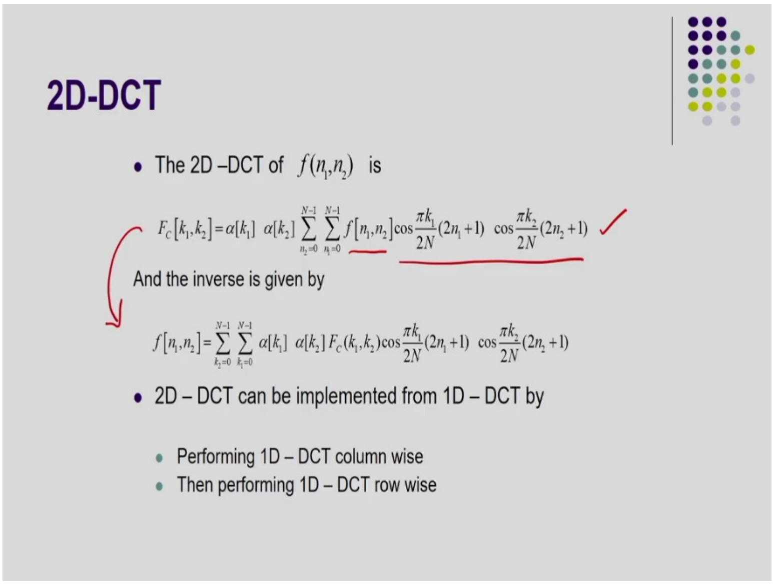
And I have already explained that most of the energy is available only in few coefficients. So, that is why I am considering only the high-value coefficients, only this part I am considering neglecting all the remaining coefficients, only this portion I am considering and after considering these coefficients, I am reconstructing the image. So, this is the reconstructed image and you can see if I compare visually this image and this image, visually you cannot see the significant difference between the input image and the reconstructed image.

So, you can see here, that DCT and DFT I have explained. So, from the DFT you can determine the DCT I have shown to a process and first, you have to define a new sequence from the new sequence, I have to calculate the DFT and from the DFT, I can determine the DCT. DFT you can calculate by the popular algorithms like FFT the fast Fourier transform and from this, you can easily determine the DCT.

And if I want to compare the DCT and the DFT you can see DCT has more advantages as compare to DFT. The first advantage is the DCT is a real transformation DFT is not a real transformation, the second point is the energy compaction property. So, if I compare the DFT and the DCT, so DCT is better than DFT.

And after this, I have shown that one example, how to reconstruct the image from the DCT coefficients. So, only we have to consider few coefficients based on the energy value and we can neglect the remaining coefficients because most of the energy is available only in few coefficients. So, you can see the difference between the DCT and the DFT.

(Refer Slide Time: 22:21)



2D-DCT

- The 2D –DCT of $f(n_1,n_2)$ is

$$F_C[k_1,k_2]=\alpha[k_1]\ \alpha[k_2]\sum_{n_2=0}^{N-1}\sum_{n_1=0}^{N-1} \underline{f[n_1,n_2]}\cos\frac{\pi k_1}{2N}(2n_1+1)\ \cos\frac{\pi k_2}{2N}(2n_2+1) \checkmark$$

And the inverse is given by

$$f[n_1,n_2]=\sum_{k_2=0}^{N-1}\sum_{k_1=0}^{N-1}\alpha[k_1]\ \alpha[k_2]\,F_C(k_1,k_2)\cos\frac{\pi k_1}{2N}(2n_1+1)\ \cos\frac{\pi k_2}{2N}(2n_2+1)$$

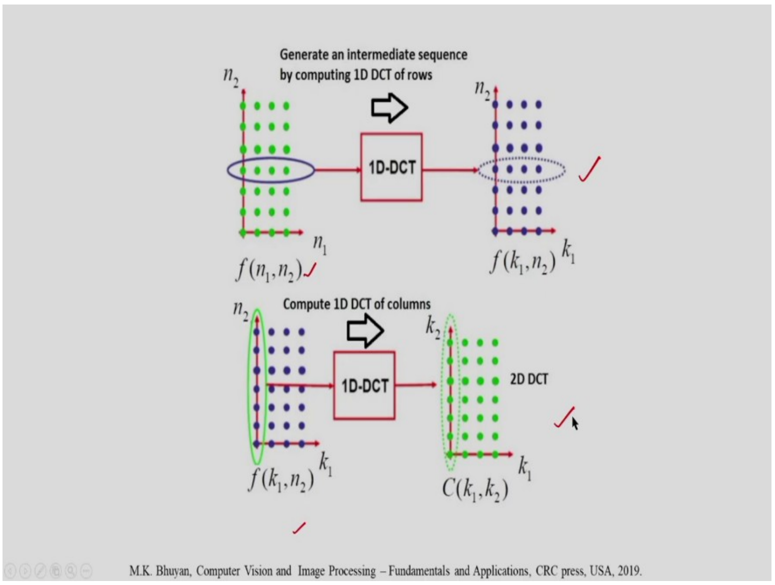
- 2D – DCT can be implemented from 1D – DCT by
  - Performing 1D – DCT column wise
  - Then performing 1D – DCT row wise

Now, I am showing the 2D-DCT, the two-dimensional Discrete Cosine Transform. So, this is the extension of only 1D-DCT you can see here. So, F c k1, k2, so k1 is the index corresponding to the1 direction that is the x-direction suppose, and K2 is the index corresponding to the y-direction and I have alpha k1, alpha k2 and my input to the sequences this f n1, n2 and my cos term, the two cos term cos pi k1 divided by twice N twice n plus 1 and cos pi k2 divided by twice N twice N2 plus 1.

And from this expression, I can determine the inverse DCT. So, I can determine the inverse DCT by using this equation. Now, already I have explained two properties, one is the separable property of the kernel and also the symmetric property. So, because of the separable property, I can implement the transformation, the 2D transformation first along the row direction in one direction, and after this I can perform the 1D transformation in another direction.

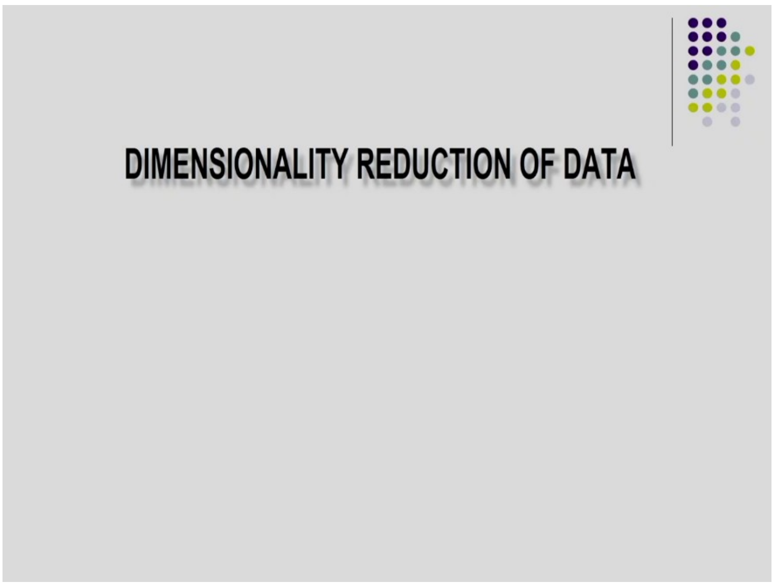
So, that means, for a 2D-DCT that can be implemented by one-dimensional DCT. So, first I can perform the 1D-DCT along with the columns and after this, I can perform the 1D-DCT along the rows. So, this is because of the separable property of the kernel.

(Refer Slide Time: 23:54)


Generate an intermediate sequence by computing 1D DCT of rows
n_2
n_1
1D-DCT
$f(n_1,n_2)$
$f(k_1,n_2)$
k_1
Compute 1D DCT of columns
k_2
2D DCT
$C(k_1,k_2)$
M.K. Bhuyan, Computer Vision and Image Processing – Fundamentals and Applications, CRC press, USA, 2019.

Here, I have shown the procedure. So, my input data sequence is f n1, n2 and first I have to apply the DCT, one-dimensional DCT along the rows. So, in this case, I am getting this one f k1 n2. So, DCT is applied along the rows and after this, I am applying the DCT along with the columns. So, after this, I am getting the 2D-DCT that is that two-dimensional Discrete Cosine Transform.

(Refer Slide Time: 24:24)



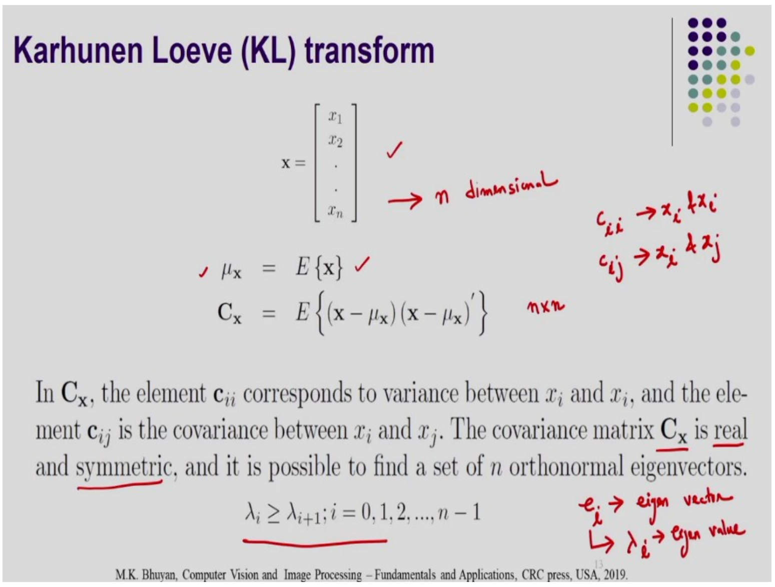
DIMENSIONALITY REDUCTION OF DATA

So, this is about the DCT the next I want to discuss the concept of dimensionality reduction of data. So, up till now, I discussed that these two transformations one is the DFT another one DCT. In case of the DFT and in case of the DCT the transformation kernel is fixed. Yeah, already you know what is the transformation kernel for the DFT and also for the DCT.

Now, I am going to discuss another transformation that is called the KL transformation. In the KL transformation, the transformation kernel is not fixed. The transformation kernel depends on the statistics of the input data. That is the difference between the KL transformation and the DCT and the DFT, or maybe I can consider that discrete sine transform, I have not discussed the discrete sine transform.

So, for all these transformations DCT, DST, and also the DFT. The transformation kernel is fixed. In the KL transformation, the transformation kernel depends on statistics of the input data. Now, let us discuss about the KL transformation.

(Refer Slide Time: 25:38)



Karhunen Loeve (KL) transform

$$\mathrm{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} \checkmark \quad \rightarrow n \text{ dimensional}$$

$$\checkmark\ \mu_{\mathrm{x}} = E\{\mathrm{x}\} \checkmark$$

$$\mathrm{C_x} = E\left\{(\mathrm{x}-\mu_{\mathrm{x}})(\mathrm{x}-\mu_{\mathrm{x}})'\right\} \quad n \times n$$

$c_{ii} \rightarrow x_i \,\&\, x_i$

$c_{ij} \rightarrow x_i \,\&\, x_j$

In $\mathbf{C_x}$, the element $\mathbf{c}_{ii}$ corresponds to variance between x_i and x_i, and the element $\mathbf{c}_{ij}$ is the covariance between x_i and x_j. The covariance matrix $\underline{\mathbf{C_x}}$ is <u>real</u> and <u>symmetric</u>, and it is possible to find a set of n orthonormal eigenvectors.

$$\underline{\lambda_i \geq \lambda_{i+1}; i = 0, 1, 2, ..., n-1}$$

$e_i \rightarrow$ eigen vector

$\hookrightarrow \lambda_i \rightarrow$ eigen value

M.K. Bhuyan, Computer Vision and Image Processing – Fundamentals and Applications, CRC press, USA, 2019.

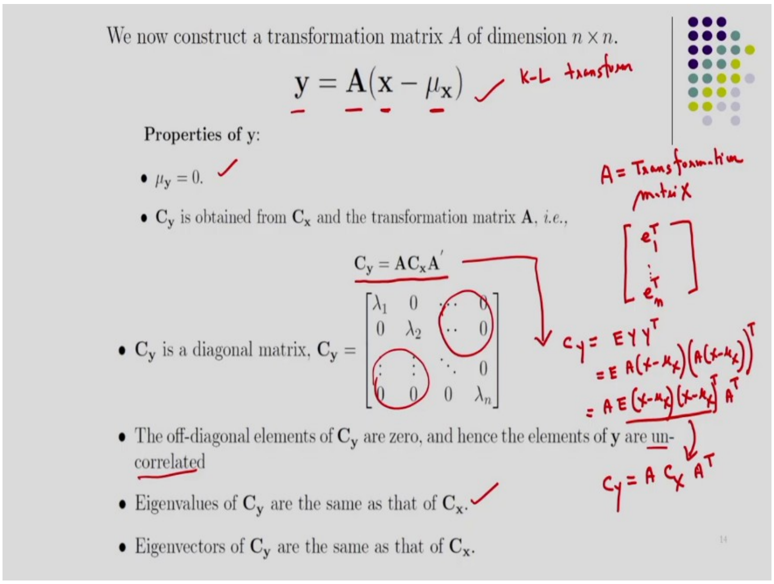
So, that is the Karhunen Loeve Transformation, that is the KL transformation. So, what is the principle here? So, first I am considering a population vector, suppose a vector is considered that is the x is a vector, the population vector is considered x1, x2, x n, and for this vector, I can determine the mean, the mean I can determine that is the mu x I can determine.

So, this input vector is the n-dimensional corresponding to this n-dimensional vector I can determine the mean that is mu x I can determine. And also, I can determine the covariance matrix, the covariance matrix is C x. What is the dimension of the covariance matrix? It is N cross N, so I have the mean and the covariance. Now, if I consider the elements of the covariance matrix suppose C i i corresponds to the variance between x i and x i.

Another one is C i j that is the covariance between x i and x j and one important point is the covariance matrix that is C x is a real matrix and also the symmetric matrix, so it is a symmetric matrix, then in this case it is possible to find a set of an orthonormal eigenvector. So, it is easy to find the n number of orthonormal eigenvectors. So, in this case, I am considering e i as the eigenvectors, and corresponding to this eigenvector, my eigenvalue is lambda i this is my eigenvalue.

So, from the covariance matrix I can determine the eigenvector and after this the corresponding eigenvalue I can determine. These eigenvalues I can arrange in the descending order of the magnitude, you can see here the eigenvalues are arranged in the descending order of the magnitude. So, lambda i is greater than equal to lambda i plus 1 corresponding to i is equal to 0,2, n minus 1. Now, the next step is,

(Refer Slide Time: 28:18)



We now construct a transformation matrix A of dimension $n \times n$.

$$\underline{\mathbf{y}} = \underline{\mathbf{A}}(\underline{\mathbf{x}} - \underline{\mu_{\mathbf{x}}}) \quad \checkmark \text{ K-L transform}$$

Properties of $\mathbf{y}$:

- $\mu_{\mathbf{y}} = 0$. ✓
- $\mathbf{C_y}$ is obtained from $\mathbf{C_x}$ and the transformation matrix $\mathbf{A}$, *i.e.*,

$$\underline{\mathbf{C_y} = \mathbf{A}\mathbf{C_x}\mathbf{A}'}$$

- $\mathbf{C_y}$ is a diagonal matrix, $\mathbf{C_y} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \lambda_n \end{bmatrix}$
- The off-diagonal elements of $\mathbf{C_y}$ are zero, and hence the elements of $\mathbf{y}$ are un-correlated
- Eigenvalues of $\mathbf{C_y}$ are the same as that of $\mathbf{C_x}$. ✓
- Eigenvectors of $\mathbf{C_y}$ are the same as that of $\mathbf{C_x}$.

A = Transformation matrix

$$\begin{bmatrix} e_1^T \\ \vdots \\ e_n^T \end{bmatrix}$$

$$C_y = E\, y y^T$$
$$= E\, A(x-\mu_x)\left(A(x-\mu_x)\right)^T$$
$$= A\, E\,\underline{(x-\mu_x)(x-\mu_x)^T}\, A^T$$
$$C_y = A\, C_x\, A^T$$

I want to determine the transformation matrix. The transformation matrix is A, so this is my transformation matrix. The first row of the transformation matrix is the eigenvector corresponding to the largest eigenvalue. So, in this case, I am considering e 1 suppose, so this is the first row of the transformation matrix is the eigenvector corresponding to the largest eigenvalue and like this, I am considering what will be the last?

The last row of the transformation matrix is the e n T n that is the last row of that transformation matrix is the eigenvector corresponding to smallest eigenvalue, because in this case already I have explained the eigenvalues are arranged in the descending order of magnitude. Now, in this case, I am defining a transformation, the transformation is y is equal to A x minus mu x that I am determining.

And this transformation is called the KL transformation. So, this is my transformation matrix x is the input vector, mean is the mean already I have determined, and I am getting the transform data, the transformed data is y. So, this transformation is called the KL transformation. So, now I want to see the properties of y, y means the transform data.

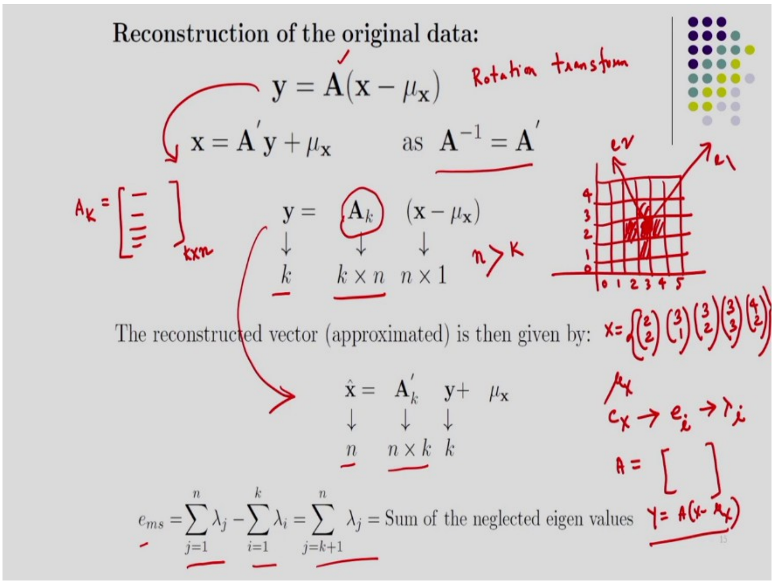
So, the first property is the mean of y is equal to 0, the output will be 0 mean and also, I can determine the covariance matrix of y. The covariance matrix of y is obtained from C x and the transformation matrix A. So, I can determine the covariance matrix this I can determine the covariance matrix of y that is nothing but the C y, it is nothing but the expected value of y y transpose the mean of y is equal to 0.

So, that is what I can write like this. So, E is equal to I can write A x minus mu x and A x minus mu x transpose So, it is equal to A expected value of E x minus mu x x minus mu x transpose and A T. So, from this you can see the covariance matrix of y is equal to A C x, so this part is nothing but C x and A T, so I have determined this. So, here you can see the covariance matrix of y is equal to A C x A transpose, so this is the covariance matrix. And you can see the covariance matrix of y is a diagonal matrix.

So, that means, you can see the only the diagonal elements are available and the off-diagonal elements all these elements are 0, so C y is a diagonal matrix. So, what is the meaning of this? So, already I have explained the meaning that is, after the transformation that transformed data will be uncorrelated the original data is highly correlated.

But after the transformation, because I have the diagonal covariance matrix, after the transformation the transformed data will be uncorrelated. So, that is the meaning of this, the transformed data will be uncorrelated. And another thing is the eigenvalues of C y are the same as that of C x and the eigenvectors of C y are the same as that of C x that you can verify.

(Refer Slide Time: 32:12)



Reconstruction of the original data:

$$y = A(x - \mu_x)$$ Rotation transform

$$x = A' y + \mu_x \qquad \text{as} \quad A^{-1} = A'$$

$$A_K = \begin{bmatrix} - \\ - \\ - \\ - \end{bmatrix}_{K \times n}$$

$$\underset{k}{\underset{\downarrow}{y}} = \underset{k \times n}{\underset{\downarrow}{A_k}} \; \underset{n \times 1}{\underset{\downarrow}{(x - \mu_x)}} \qquad n > K$$

The reconstructed vector (approximated) is then given by:

$$\underset{n}{\underset{\downarrow}{\hat{x}}} = \underset{n \times k}{\underset{\downarrow}{A_k'}} \; \underset{k}{\underset{\downarrow}{y}} + \mu_x$$

$$e_{ms} = \sum_{j=1}^{n} \lambda_j - \sum_{i=1}^{k} \lambda_i = \sum_{j=k+1}^{n} \lambda_j = \text{Sum of the neglected eigen values}$$

$$X = \left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix} \begin{pmatrix} 3 \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} \begin{pmatrix} 3 \\ 3 \end{pmatrix} \begin{pmatrix} 4 \\ 2 \end{pmatrix} \right\}$$

μ_x

$C_x \rightarrow e_i \rightarrow \lambda_i$

$A = [\ ]$

$Y = A(x - \mu_x)$

Now, let us consider this case so, the reconstruction of the original data but before going to the reconstruction of the original data, what I can show you? Suppose, I am considering one image suppose, so in this image suppose, these are the points suppose 0,1,2,3,4,5, like this 0,1,2,3,4. Suppose, in this image I have these pixels.

So, 2D binary image I am considering and, in this case, white means object is present and black means no object is present that means 0. So, in this case, I am considering white and this is the white portion. Corresponding to this I can determine the population vector. So, corresponding to which point the pixel is present. So, 2 and 2 suppose, because this point is 2 and 2.

Suppose, another point is 3 and 1, in this point it is present 3 and 1, the another one is 3 and 2, another point is 3 and 3 in this pixel the pixel is present that is white pixel is present. Another one is 4 and 2, so I can consider. So, this is my population vector. From the population vector, I can determine the mean, the mean of this vector I can determine, and also, I can determine the covariance matrix.

After determining the covariance matrix and the mean I can determine the eigenvalues and eigenvectors of the covariance matrix that I can determine the eigenvectors and also the eigenvalues I can determine and after this, I can determine the transformation matrix, the transformation matrix is A, the transformation matrix is obtain by eigenvectors. After this I can apply the transformation, the transformation is A x minus mu x I can apply, so this is my transformation.

So, in this example, I am considering one image and, in this image, I am considering the white pixels these pixels, and corresponding to these pixels I am having the vector x. So, from this x I can determine mean, covariance, and the transformation matrix and after this I can apply the transformation. So, after the transformation, what I am getting? I will be getting a new coordinate system.

So, my new coordinate system will be something like this, this is the direction of the eigenvectors e1 and e2. So, application of this transformation I am getting a new coordinate system whose origin is at the centroid of the object pixel. So, I can determine the centroid of the object pixel it is the centroid of the object pixel and the axis of the new coordinate system will be parallel to the direction of the eigenvectors.

So, here I have shown eigenvectors e1 and e2. So, one axis is e1 another axis is e2. So, I can say the KL transformation is nothing but the rotation transformation. This is the rotation transformation that means it aligns the data along the direction of the eigenvectors and because of these alignments, different elements of y will be uncorrelated. So, I am repeating this that means, I am doing the KL transformation.

The KL transformation is nothing but the rotation transformation, it aligns the data along the direction of the eigenvectors, and because of this alignment different elements of y will be uncorrelated, so that is the meaning of the KL transformation. After this, I am considering the deconstruction of the original data. So, in this case, you can see, so from this transformation, I can determine the reconstructed data that is the x I can determine, A transpose y plus mu x.

And in this case, I am considering the orthogonal transformation A inverse is equal to A transpose, so I am having this one. So, I can reconstruct the original data and the perfect reconstruction is possible. Now, let us consider the transformation matrix is something like suppose A k, that is I can call as the truncated transformation matrix.

In this transformation matrix, I am not considering all the eigenvectors. In the first case, I have considered all the eigenvectors for the construction of the transformation matrix. In the second case, I am considering A k. So, in this case, I am only considering k number of eigenvectors of C x, C x is the covariance matrix of the input data. So, I am considering k largest eigenvectors.

So, then in this case I have how many rows will be there now because I am only considering k number of eigenvectors, so dimension of A k will be k cross n. So, here I have shown k cross n. And in this case, I am applying that KL transformation. So, A k is nothing but the truncated transformation matrix x minus mu x and I am getting y. So, what will be the dimension of y now, the dimension of y is the k.

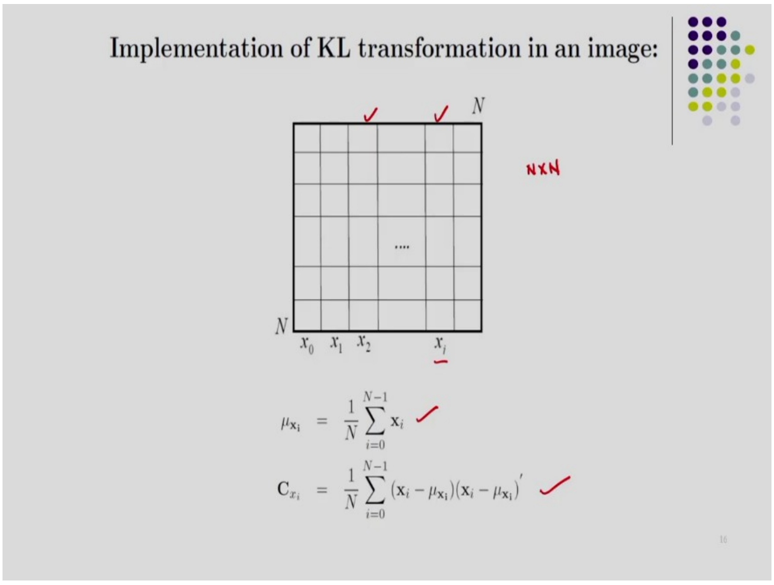
So, that means the dimension is reduced, the original dimension was n and after the transformation, I have the dimension k and n is greater than k. So, this is the principle of dimensionality reduction. So, I can reduce the dimension of the data. So, in the second case, if I want to reconstruct the original data that is not possible because I am not considering all the eigenvectors.

So, that is why if I want to reconstruct from this one, I am getting the approximate reconstruction. So, that is why this is x gap. So, you can see I have this A k transpose it is n cross k and it is k the dimension of the y is k, and what will be the dimension of the reconstructed data, the reconstructed vector? The dimension will be n. So, in the second case, the perfect reconstruction is not possible because I am not considering all the eigenvectors for the construction of the transformation matrix.

Now, in this case, I can determine the mean square error because perfect reconstruction is not possible. So, what will be the mean square error? So, you can see here, the first time you can see that is the lambda j I am determining, this j is equal to 1 to n that means, I am considering all the eigenvalues minus i is equal to 1 to k lambda i that means, I am considering only k number of eigenvalues.

So, if I subtract these two you can see here, so I am getting this one. So, j is equal to k plus 1 up to n lambda j. So, that is nothing but some of the neglected eigenvalues. So, the mean square error depends on the sum of the neglected eigenvalues.

(Refer Slide Time: 39:38)



Implementation of KL transformation in an image:

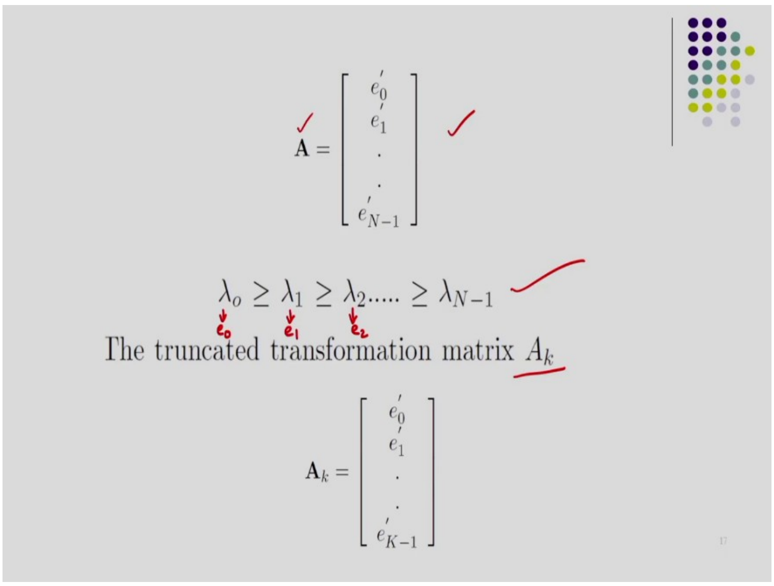
$$\mu_{x_i} = \frac{1}{N}\sum_{i=0}^{N-1} x_i$$

$$C_{x_i} = \frac{1}{N}\sum_{i=0}^{N-1} (x_i - \mu_{x_i})(x_i - \mu_{x_i})'$$

Now, implementation under KL transformation in an image. So, if I consider this is an image. So, in this image I am considering N cross N image, in this N cross N image, you can see the columns, the columns are x naught, x1 x2 like this. So, in this case, every column I can consider as a vector.

So, this column I can consider as a vector, this column I can consider a vector, and after considering this is a vector corresponding to particular this column x i I can determine the mean and also, I can determine the covariance corresponding to a particular column, the column is x i that I am considering.

(Refer Slide Time: 40:20)



$$A = \begin{bmatrix} e_0' \\ e_1' \\ \cdot \\ \cdot \\ e_{N-1}' \end{bmatrix}$$

$$\underset{e_0}{\lambda_o} \geq \underset{e_1}{\lambda_1} \geq \underset{e_2}{\lambda_2} \ldots \geq \lambda_{N-1}$$

The truncated transformation matrix $\underline{A_k}$

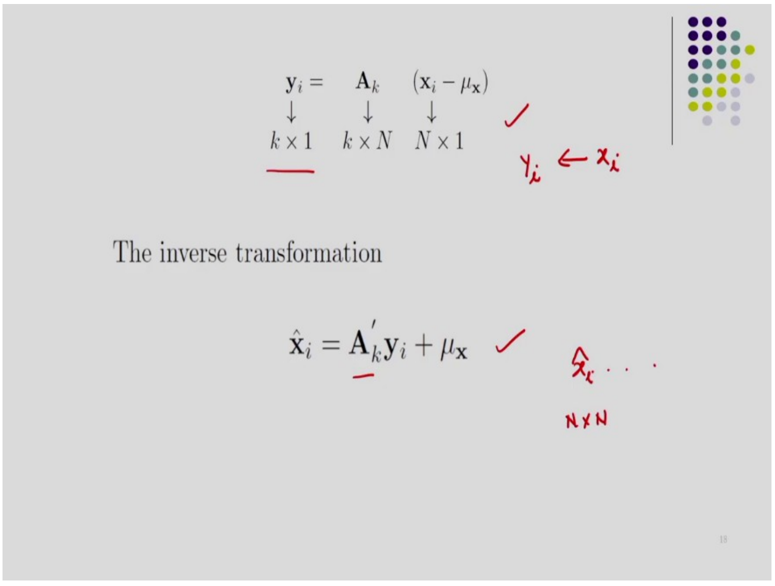
$$A_k = \begin{bmatrix} e_0' \\ e_1' \\ \cdot \\ \cdot \\ e_{K-1}' \end{bmatrix}$$

And after this I am determining the transformation matrix, the transformation matrix is A. So, what is the e naught dash? e naught dash is the eigenvectors corresponding to the first largest eigenvalue. What is e1 dash? So, e1 dash is the second eigenvectors corresponding to the second largest eigenvalue. So, like this, I have to arrange all the eigenvalues that is in the descending order of magnitude.

So, I am arranging all the eigenvalues in the descending order of the magnitude and corresponding to the lambda naught I have the eigenvalue eigenvector e naught, corresponding to lambda 1 I have the eigenvector e1, corresponding to lambda 2 I have the eigenvector e2 like this. And from this, I am constructing the transformation matrix, the transformation matrix is A.

Also, I can consider the truncated transformation matrix that is the truncated transformation matrix is A k. In this case, I am not considering all the eigenvectors, I am only considering the first k number of eigenvectors corresponding to the eigenvalues. So, I am having the truncated transformation matrix, after this what I am doing.

(Refer Slide Time: 41:36)



$$\begin{array}{ccc} \underline{\mathbf{y}_i} = & \mathbf{A}_k & (\mathbf{x}_i - \mu_\mathbf{x}) \\ \downarrow & \downarrow & \downarrow \\ k \times 1 & k \times N & N \times 1 \end{array} \checkmark$$

$\mathbf{y}_i \leftarrow \mathbf{x}_i$

The inverse transformation

$$\hat{\mathbf{x}}_i = \underline{\mathbf{A}'_k}\mathbf{y}_i + \mu_\mathbf{x} \checkmark$$

$\hat{x}_i$ . . .

N×N

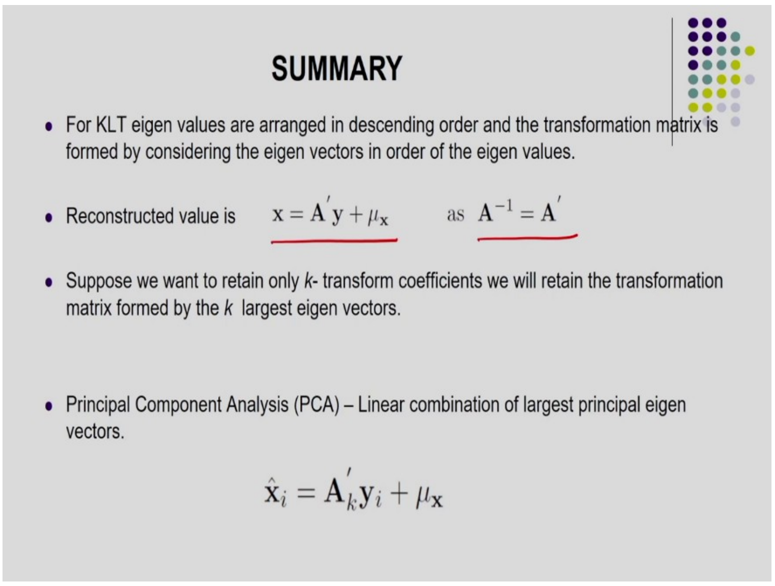
I am showing the transformation here, so y i is equal to k cross 1, A k k cross N and in this case N cross 1, so I am doing this one. So, this is the transformation I am having. So, after the transformation, my transform will be k cross 1, like this. So, in this case for the entire image how to get this. So, if I collect all the y's mainly, so I have the y i corresponding to x i.

So, for all the columns if I can determine y i, in this case, for every x i of the image we get y i. If the transformation of all the column vectors of the 2D image is done, then we will get the

N transform vector that is y i we will be getting with a dimension k. And in this case, you can see it is an inverse transformation I am doing that is the reconstructed value of xi.

Approximate reconstruction because in this case, I am only considering the truncated transformation matrix. So, if I do the collection of all the x i's, then in this case I can get the reconstructed image. So, collection of all the excise will give the N cross N transform image.

(Refer Slide Time: 43:02)



SUMMARY

- For KLT eigen values are arranged in descending order and the transformation matrix is formed by considering the eigen vectors in order of the eigen values.
- Reconstructed value is $x = A'y + \mu_x$ as $A^{-1} = A'$
- Suppose we want to retain only *k*- transform coefficients we will retain the transformation matrix formed by the *k* largest eigen vectors.
- Principal Component Analysis (PCA) – Linear combination of largest principal eigen vectors.

$$\hat{x}_i = A_k' y_i + \mu_x$$

So, in summary, I can say in the case of transformation, the eigenvalues are arranged in the descending order of magnitude and the transformation matrix is formed by considering the eigenvectors in the order of the eigenvalues. So, already I have explained this concept and also the reconstructed below I can determine by using this expression and in this case, I am considering the orthogonal transformation A inverse is equal to A transpose.

And in this case, suppose we want to retain only k transform coefficients we will retain the transformation matrix formed by the k largest eigenvectors, and based on this I can define one transformation that is the principal component analysis that is nothing but the linear combination of largest principal eigenvectors. So, this is the definition of the principal component analysis. So, this is a summary of the KL transformation.

So, what is the summary of the KL transformation? The first from the population vector I can determine the mean and the covariance and from the covariance matrix I can determine the eigenvectors and the eigenvalues. The eigenvalues are arranged in descending order of magnitude. And after this I can determine the transformation matrix, the transformation matrix is obtained from the eigenvectors and after this, I can do the KL transformation.

And again, I can reconstruct the original data and also, I can determine the truncated transformation matrix. In the truncated transformation matrix, I can only consider the k number of largest eigenvectors, and based on this I can determine the truncated transformation matrix.

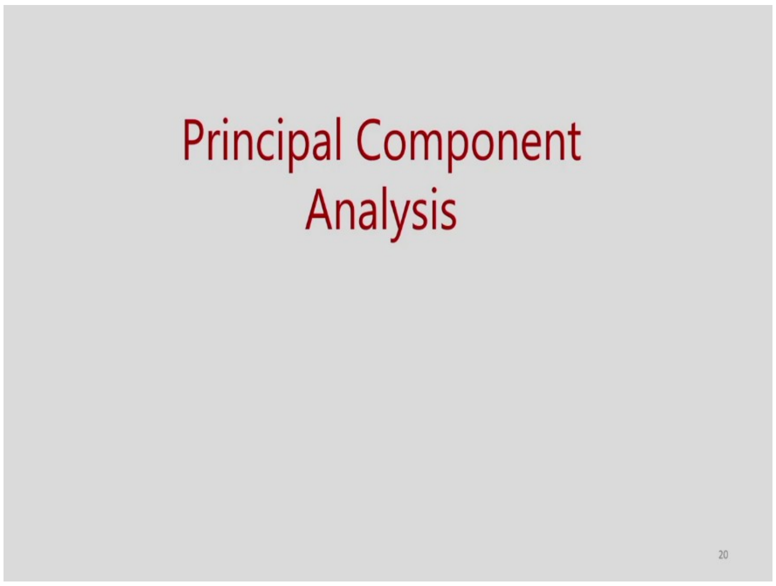
Now, you can see this if I want to do compression of data, the compression depends on the value k. So, how many eigenvectors I am considering for making the transformation matrix. And for reconstruction what information I need? I need the information of A k and also, I need the information of y i that is the transformed data that information I need. From this I can determine x, x is the input data that I can determine.

So, this is about the KL transformation the one problem of the KL transformation here you see, the transformation depends on the statistics of the input data. So, already I have mentioned that is in other transformations, the transformation kernel is fixed, but in this case, you can see the transformation kernel is not fixed it is derived from the input data. So, suppose in the applications like the real-time applications, suppose the non-stationary data.

Then in this case what will happen for each and every instance I have to determine the mean of the data vector and after this, I have to determine the covariance matrix from the covariance matrix, I have to determine the eigenvectors and eigenvalues and after this, I can determine a transformation matrix. Since the data is non-stationary for each and every instance, I have to do this. So, that is why the real-time implementation is very difficult it cannot be applied.

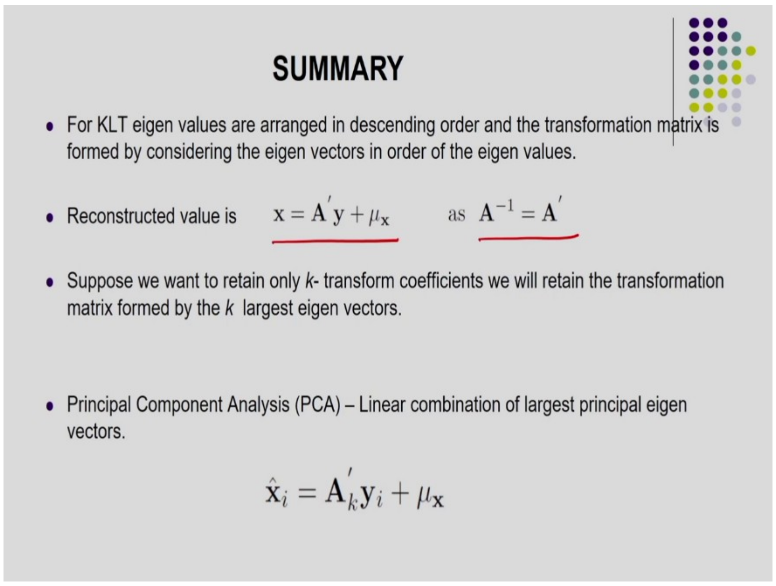
So, that is why the KL transformation is not applied for image compression or video compression because for non-stationary data, we have to compute all the parameters mainly the mean covariance, and also, I have to determine the transformation matrix. So, that is why real-time implementation is difficult. But, if I consider the decorrelating property, so it can perfectly decorrelate the input data because I am having the diagonal covariance matrix. So, that is why the perfect decorrelation is possible by considering the KL transformation.

(Refer Slide Time: 46:53)



Principal Component Analysis

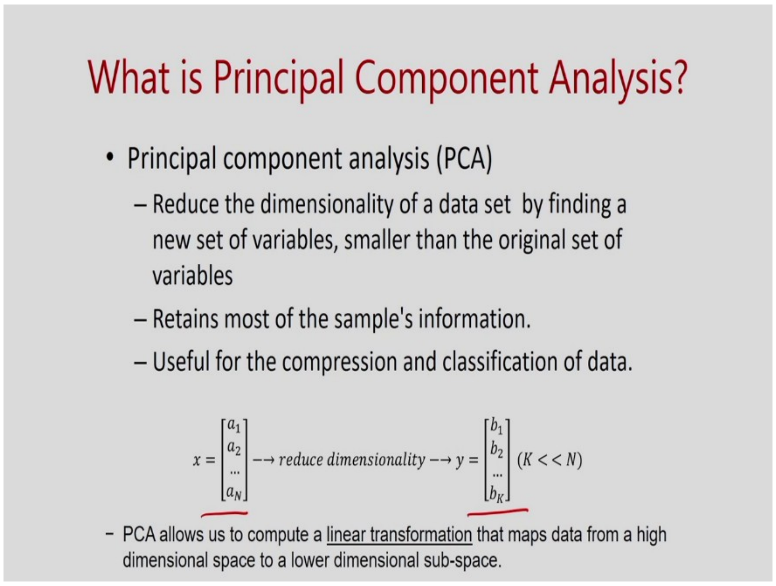
20



SUMMARY

- For KLT eigen values are arranged in descending order and the transformation matrix is formed by considering the eigen vectors in order of the eigen values.
- Reconstructed value is $\underline{\mathrm{x} = \mathrm{A}^{'}\mathrm{y} + \mu_{\mathrm{x}}}$ as $\underline{\mathrm{A}^{-1} = \mathrm{A}^{'}}$
- Suppose we want to retain only *k*- transform coefficients we will retain the transformation matrix formed by the *k* largest eigen vectors.
- Principal Component Analysis (PCA) – Linear combination of largest principal eigen vectors.

$$\hat{\mathrm{x}}_i = \mathrm{A}_k^{'}\mathrm{y}_i + \mu_{\mathrm{x}}$$



What is Principal Component Analysis?

- Principal component analysis (PCA)
  - Reduce the dimensionality of a data set by finding a new set of variables, smaller than the original set of variables
  - Retains most of the sample's information.
  - Useful for the compression and classification of data.

$$x = \underline{\begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_N \end{bmatrix}} \dashrightarrow reduce\ dimensionality \dashrightarrow y = \underline{\begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix}} \quad (K << N)$$

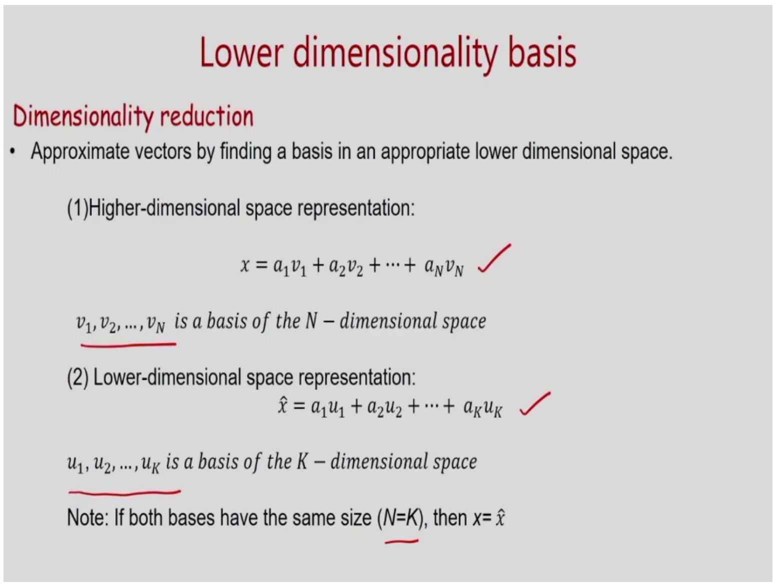
- PCA allows us to compute a <u>linear transformation</u> that maps data from a high dimensional space to a lower dimensional sub-space.

Now, I want to show the principal component analysis. What is the principal component analysis? Already, I have explained the principal component analysis means the linear combination of largest principal eigenvectors. So, in the principal component analysis again I am showing the same thing in the KL transformation.

So, I can reduce the dimensionality of the data set by finding a new set of variables smaller than the original set of variables and retains most of the sample's information and useful product compression and classification of data.

So, you can see I have the input vector x and after this, I am reducing the dimensionality, so the dimension is reduced, because in this case, the K is less than N. So, PCA the principal component analysis allows us to compute a linear transformation that maps data from a high dimensional space to a lower-dimensional subspace.

(Refer Slide Time: 47:51)



Lower dimensionality basis

Dimensionality reduction

- Approximate vectors by finding a basis in an appropriate lower dimensional space.

(1)Higher-dimensional space representation:

$$x = a_1 v_1 + a_2 v_2 + \cdots + a_N v_N \checkmark$$

$\underline{v_1, v_2, \ldots, v_N}$ *is a basis of the N – dimensional space*

(2) Lower-dimensional space representation:

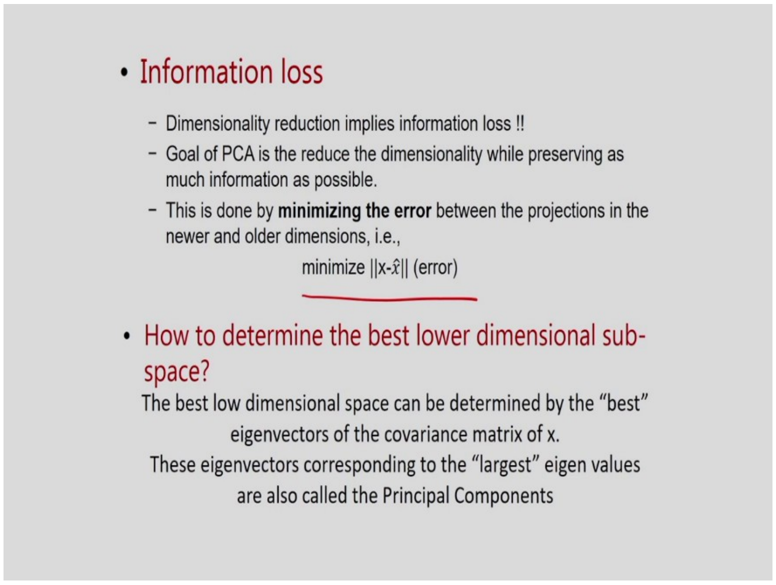
$$\hat{x} = a_1 u_1 + a_2 u_2 + \cdots + a_K u_K \checkmark$$

$\underline{u_1, u_2, \ldots, u_K}$ *is a basis of the K – dimensional space*

Note: If both bases have the same size ($\underline{N=K}$), then $x = \hat{x}$

And in this case, what is the high dimensional subspace you can see. So, this is my high dimensional subspace and this is my low dimensional subspace. This v1, v2, v n you can see here is a basis of the N-dimensional space and if I consider u1, u2, u k is the basis of the K-dimensional space and suppose, if N is equal to K that means, in this case, the dimension is not reduced. So, here I have shown this example, one in the high dimensional space another one lower-dimensional space representation.

(Refer Slide Time: 48:29)



- Information loss
  - Dimensionality reduction implies information loss !!
  - Goal of PCA is the reduce the dimensionality while preserving as much information as possible.
  - This is done by **minimizing the error** between the projections in the newer and older dimensions, i.e.,

    minimize $||x-\hat{x}||$ (error)

- How to determine the best lower dimensional sub-space?

  The best low dimensional space can be determined by the "best" eigenvectors of the covariance matrix of x.

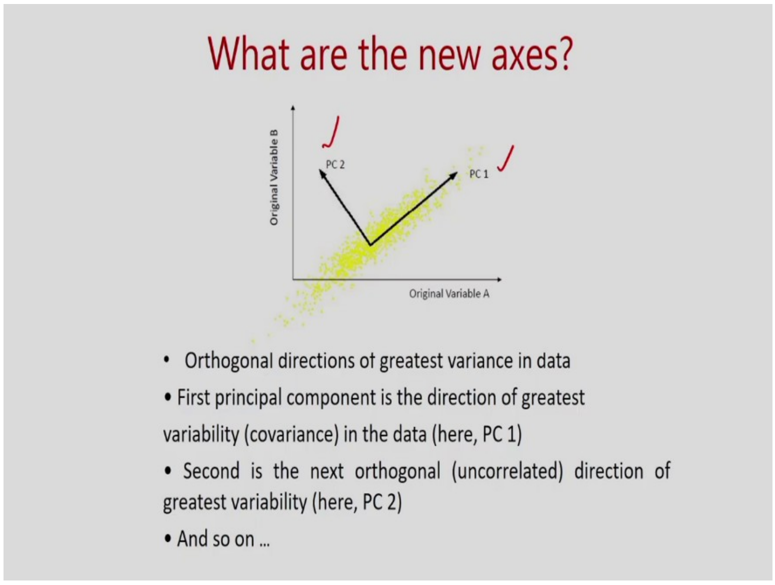
  These eigenvectors corresponding to the "largest" eigen values are also called the Principal Components

And the information loss, what is the information loss? Because of the dimensionality reduction, information will be lost. So, the goal of the PCA is to reduce dimensionality while preserving as much information as possible. This is mainly the minimizing the error between the projection in the new and older dimensions.

So, that means, in this case, I have to minimize this, one is the original data another one is the reconstructed data, that I have to minimize. Now, how to determine the best lower-dimensional subspace? The best low dimensional subspace can be determined by the best eigenvector of the covariance matrix of x.

So, already this concept I have explained and these eigenvectors corresponding to the largest eigenvalue, and these are called the Principal Components.

(Refer Slide Time: 49:20)



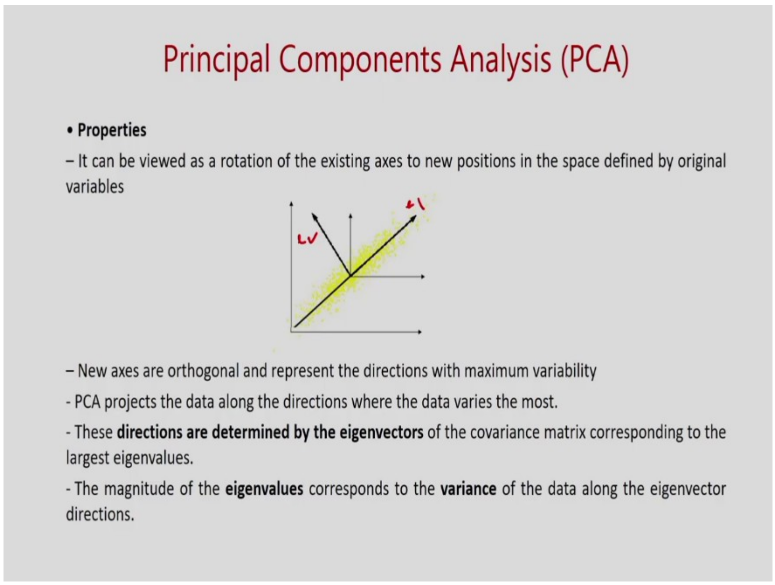
What are the new axes?

Original Variable B
PC 2
PC 1
Original Variable A

- Orthogonal directions of greatest variance in data
- First principal component is the direction of greatest variability (covariance) in the data (here, PC 1)
- Second is the next orthogonal (uncorrelated) direction of greatest variability (here, PC 2)
- And so on ...

And here I have shown the principal components that is nothing but the eigenvectors. So, orthogonal direction of the greatest variance in data. So, I have the first principal component PC 1, the second principal component PC 2, and in this case, like this, I have the principal components that is nothing but the direction of the eigenvectors, these are the eigenvectors.

(Refer Slide Time: 49:45)



Principal Components Analysis (PCA)

- **Properties**

– It can be viewed as a rotation of the existing axes to new positions in the space defined by original variables

– New axes are orthogonal and represent the directions with maximum variability

- PCA projects the data along the directions where the data varies the most.

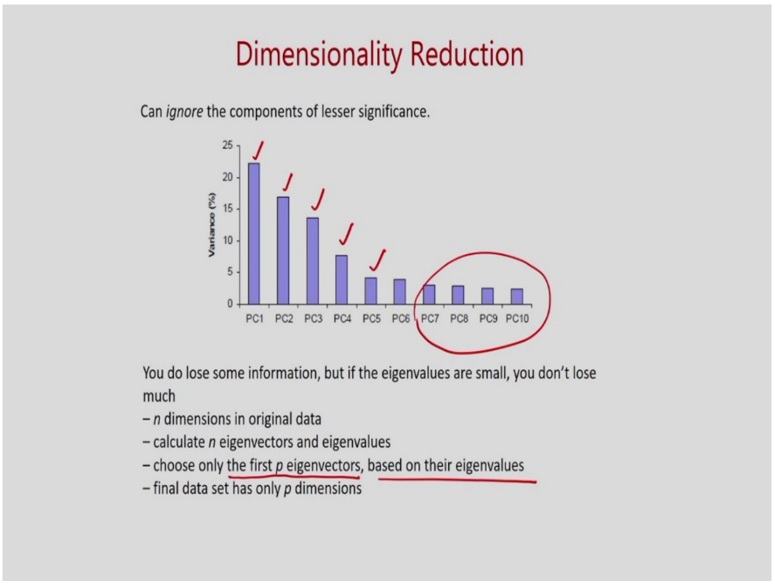
- These **directions are determined by the eigenvectors** of the covariance matrix corresponding to the largest eigenvalues.

- The magnitude of the **eigenvalues** corresponds to the **variance** of the data along the eigenvector directions.

So, already, I have explained because of the KL transformation, I will be getting a new coordinate system. In the new coordinate system, my new axis will be the eigenvectors. So, here you can see these are my eigenvectors e1, e2, these are eigenvectors. So, new axis are orthogonal and represent the directions with maximum variability, and the principal component analysis projects data along the direction where the data varies the most.

And these directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalue. So, this concept already I have explained. The magnitude of the eigenvalues corresponds to the variance of the data along the direction of the eigenvectors.

(Refer Slide Time: 50:36)



Dimensionality Reduction

Can *ignore* the components of lesser significance.

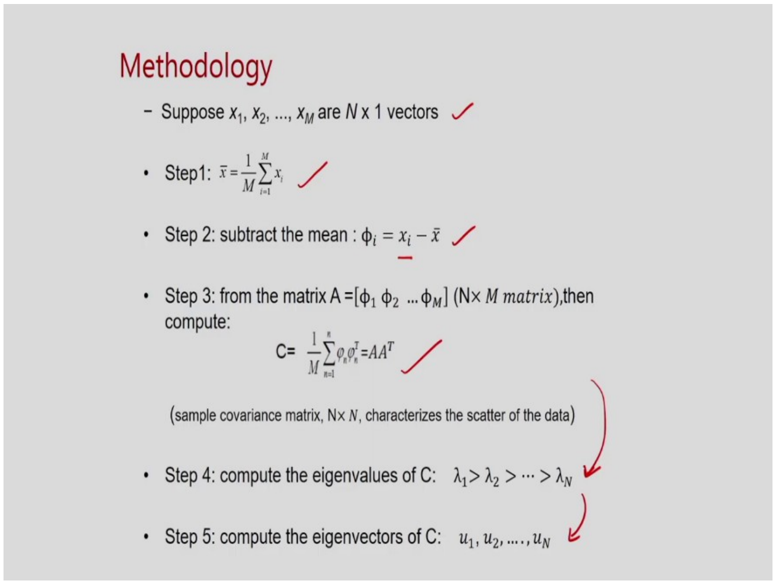
You do lose some information, but if the eigenvalues are small, you don't lose much

- *n* dimensions in original data
- calculate *n* eigenvectors and eigenvalues
- choose only the first *p* eigenvectors, based on their eigenvalues
- final data set has only *p* dimensions

And here I have shown that dimensionality reduction. Can ignore the components of the lesser significance. So, you can see I am considering the principal components PC1, PC2, PC3, PC4 all the PC like this I am considering and you can see these principal components have the maximum information. So, I can neglect the remaining principal components because already I have explained the property, that property is called energy compaction property.

So, most of the energy is available only in few coefficients. So, in this case, I am considering the largest eigenvalues and the corresponding eigenvectors. So, that means the PC1, PC2, PC3, PC4, PC5 I am considering and remaining eigenvectors I am not considering, the principal components, that is I can select the first p eigenvectors based on the eigenvalues and I can neglect the remaining eigenvectors.

(Refer Slide Time: 51:39)



Methodology

- Suppose $x_1, x_2, \ldots, x_M$ are N x 1 vectors
- Step1: $\bar{x} = \frac{1}{M}\sum_{i=1}^{M} x_i$
- Step 2: subtract the mean : $\phi_i = x_i - \bar{x}$
- Step 3: from the matrix A =$[\phi_1\ \phi_2\ \ldots \phi_M]$ (N× M $matrix$),then compute:

$$C= \frac{1}{M}\sum_{n=1}^{M}\phi_n\phi_n^T = AA^T$$

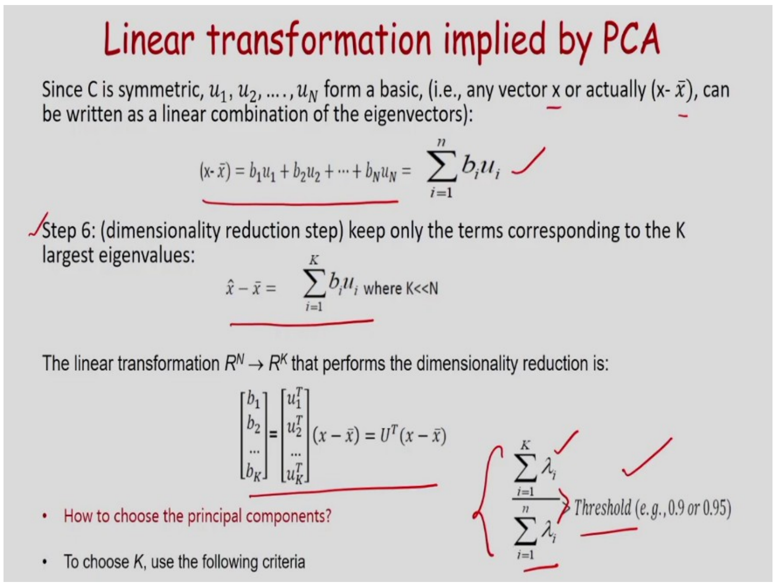
(sample covariance matrix, N× N, characterizes the scatter of the data)

- Step 4: compute the eigenvalues of C: $\lambda_1 > \lambda_2 > \cdots > \lambda_N$
- Step 5: compute the eigenvectors of C: $u_1, u_2, \ldots, u_N$

So, what is the method again I am explaining this one. So, I have the input vector. So, dimension is N cross 1, from N cross 1 I can determine the mean of this and after this from the original data, I can subtract the mean and after this, I can determine the covariance matrix I can determine, from the covariance matrix I can determine the eigenvalues I can determine and from the eigenvalues I can determine the eigenvectors.

So, this principle already this concept I have already explained. So, from the input data how to determine the mean and how to determine the covariance, and from that covariance, how to determine the eigenvalues and eigenvectors.

(Refer Slide Time: 52:20)



Linear transformation implied by PCA

Since C is symmetric, $u_1, u_2, \ldots, u_N$ form a basic, (i.e., any vector x or actually (x- $\bar{x}$), can be written as a linear combination of the eigenvectors):

$$(x-\bar{x}) = b_1u_1 + b_2u_2 + \cdots + b_Nu_N = \sum_{i=1}^{n} b_iu_i$$

Step 6: (dimensionality reduction step) keep only the terms corresponding to the K largest eigenvalues:

$$\hat{x} - \bar{x} = \sum_{i=1}^{K} b_iu_i \text{ where } K<<N$$

The linear transformation $R^N \rightarrow R^K$ that performs the dimensionality reduction is:

$$\begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_K \end{bmatrix} = \begin{bmatrix} u_1^T \\ u_2^T \\ \ldots \\ u_K^T \end{bmatrix}(x-\bar{x}) = U^T(x-\bar{x})$$

- How to choose the principal components?
- To choose K, use the following criteria

$$\frac{\sum_{i=1}^{K}\lambda_i}{\sum_{i=1}^{n}\lambda_i} > \text{Threshold (e.g., 0.9 or 0.95)}$$

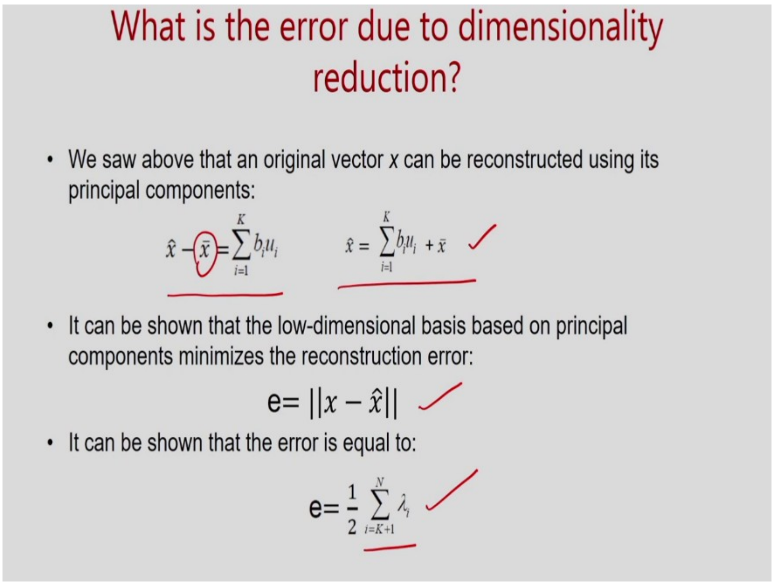
After this, you can see if the C is symmetric then these eigenvectors u1, u2, u3 like this form a basic, and in this case any vector x minus x bar that is the mean can be written as a linear combination of the eigenvectors you can see here and also if I consider the truncated transformation matrix, then, in this case, I can reduce the dimension in the step 6.

So, this is the, I am only considering the K number of eigenvectors. In the first case, I considered the N number of eigenvectors. So, in the step 6, I am considering the dimension reduction step. So, you can see the dimension is reduced by this expression and how to choose the principal component.

So, already I have explained, so principal components are selected based on the eigenvalues, the magnitude of the eigenvalues. And in this case, the one condition you can consider. So, you can see here, the lambda i this should be greater than particularly greater than. So, lambda I am considering for the K number of eigenvectors from i is equal to 1 to K.

And in the second case here you see the lambda I, I am considering from i is equal to n that is for all the eigenvalues, then in this case, if this ratio is greater than a particular threshold, so based on this condition, I can select the value of K, so threshold I have to consider. For example, I can consider point 9 or maybe point 95, and this ratio I can determine and from this, I can select the value of K, K number of eigenvectors.

(Refer Slide Time: 54:06)



What is the error due to dimensionality reduction?

- We saw above that an original vector x can be reconstructed using its principal components:

$$\underline{\hat{x} - \bar{x} = \sum_{i=1}^{K} b_i u_i} \qquad \underline{\hat{x} = \sum_{i=1}^{K} b_i u_i + \bar{x}}$$

- It can be shown that the low-dimensional basis based on principal components minimizes the reconstruction error:

$$e = ||x - \hat{x}||$$

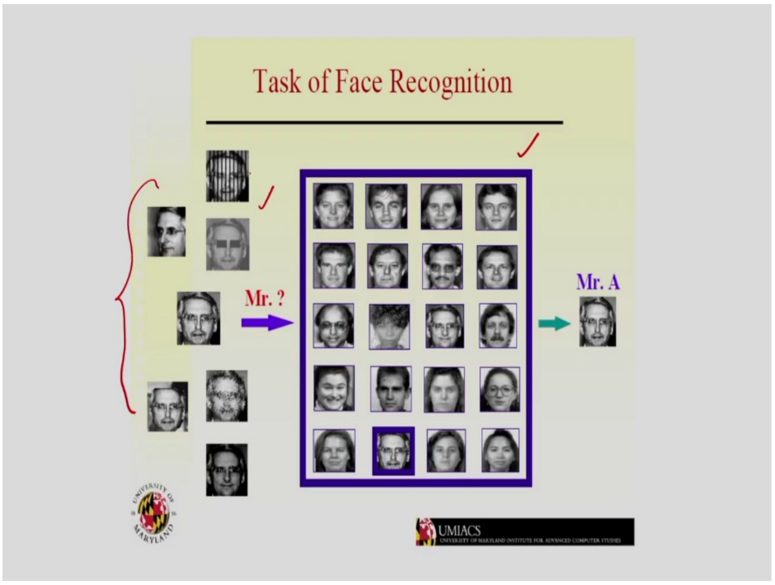
- It can be shown that the error is equal to:

$$e = \frac{1}{2} \sum_{i=K+1}^{N} \lambda_i$$

And in this case, we have seen that the original vector x can be reconstructed using the principal components. So, this is my reconstructed information, so this is my reconstructed vector. In this case, I considered a mean but in this case x approximate, and because I am considering the truncated transformation matrix, so that is why I am getting the approximate x that I can get.

And also, you can see the error also I can determine, the error means the low dimensional basis based on the principal component minimizes the reconstruction error. So, the reconstruction error I can determine and the error can be determined like this, this is the expression for the error, the error is equal to 1 by 2 and summation lambda i and in this case, I am considering the neglected eigenvalues, the sum of the neglected eigenvalues that I am considering.

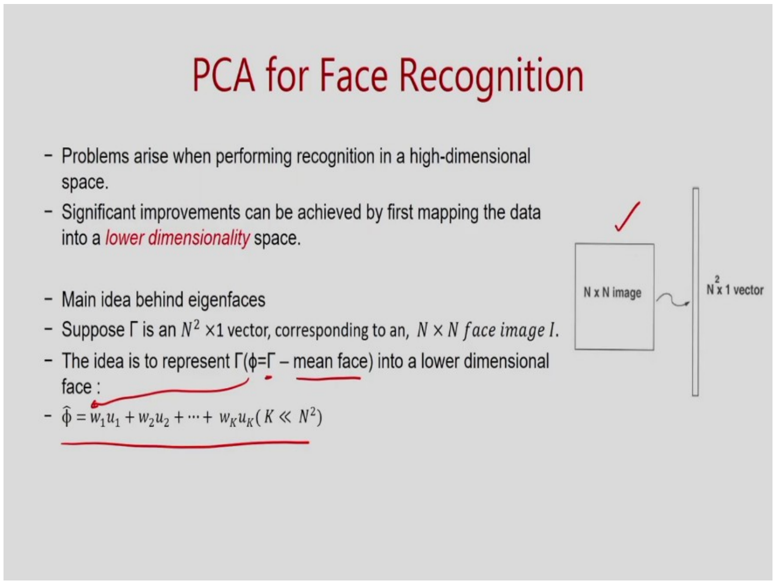
(Refer Slide Time: 55:01)



Task of Face Recognition

Now, I have shown one example, how to consider the PCA for face recognition? So, in this example, I have shown in my first class, so this is the face recognition problem. So, we have the database and we have the input images. So, in this case, I have to select whether this particular face is available in a database or not. So, this is the task of face recognition. So, my input images maybe like this, it may be occluded or maybe some illumination variation I may consider and, in this case, I have to find that particular face in the database.

(Refer Slide Time: 55:37)



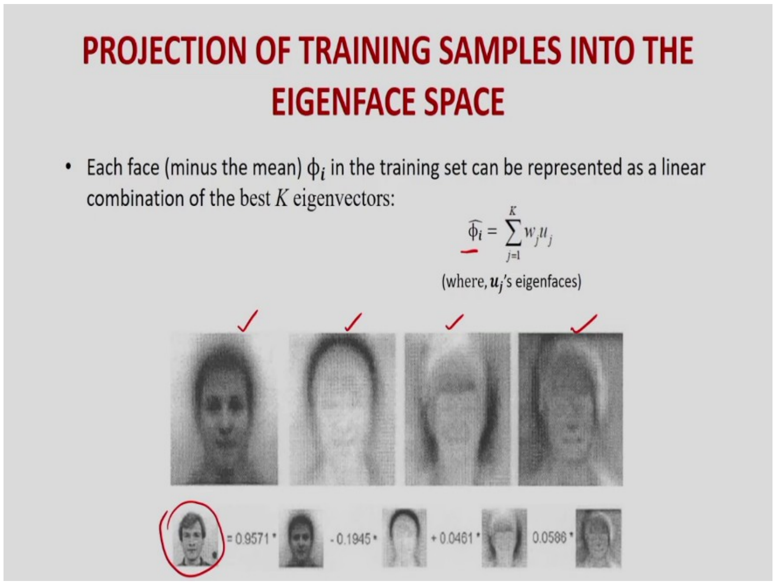
PCA for Face Recognition

- Problems arise when performing recognition in a high-dimensional space.
- Significant improvements can be achieved by first mapping the data into a *lower dimensionality* space.

- Main idea behind eigenfaces
- Suppose Γ is an $N^2 \times 1$ vector, corresponding to an, $N \times N$ *face image I*.
- The idea is to represent $\Gamma(\phi = \Gamma -$ mean face) into a lower dimensional face :
- $\hat{\phi} = w_1 u_1 + w_2 u_2 + \cdots + w_K u_K (K \ll N^2)$

For this, I can apply the PCA for the face recognition. So, this is my input image and by image that I can consider as N square into 1 vector that I can consider. And in this case, you can see the original face is this, minus mean face I can determine, the average face I can determine, and that can be approximately represented by this, this can be approximately represented like this because I am only considering the K number of basis that is expressed into low dimensionality space.

(Refer Slide Time: 56:11)



PROJECTION OF TRAINING SAMPLES INTO THE EIGENFACE SPACE

- Each face (minus the mean) ϕ_i in the training set can be represented as a linear combination of the best K eigenvectors:

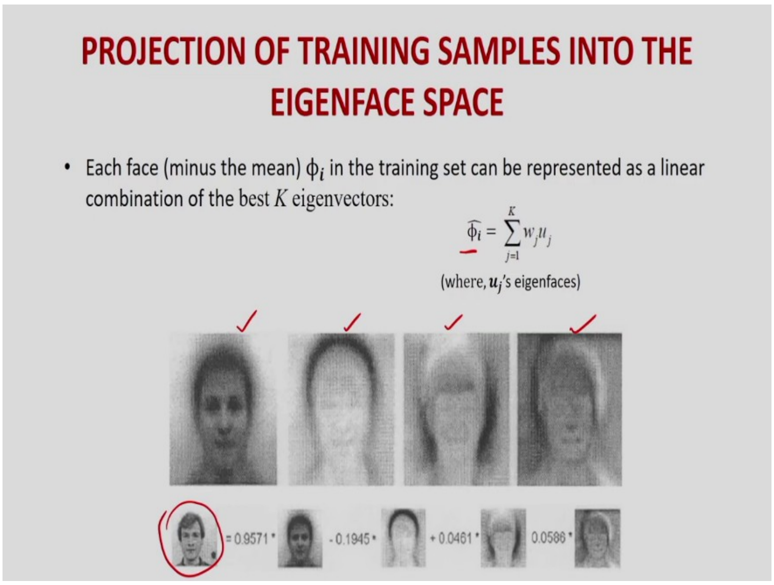
$$\hat{\phi}_i = \sum_{j=1}^{K} w_j u_j$$

(where, u_j's eigenfaces)

= 0.9571 *
- 0.1945 *
+ 0.0461 *
0.0586 *

So, you can see here, so each face minus the mean in the training set can be represented as a linear combination of K eigenvectors. So, here you can see this is the representation. So, each face minus the mean can be represented by a linear combination of the best eigenvectors. So, I have the eigenfaces like this. So, from the eigenvectors I can determine the eigenfaces, these are eigenfaces. So, each face, this face is represented as a linear combination of the K-based eigenvectors.

(Refer Slide Time: 56:53)



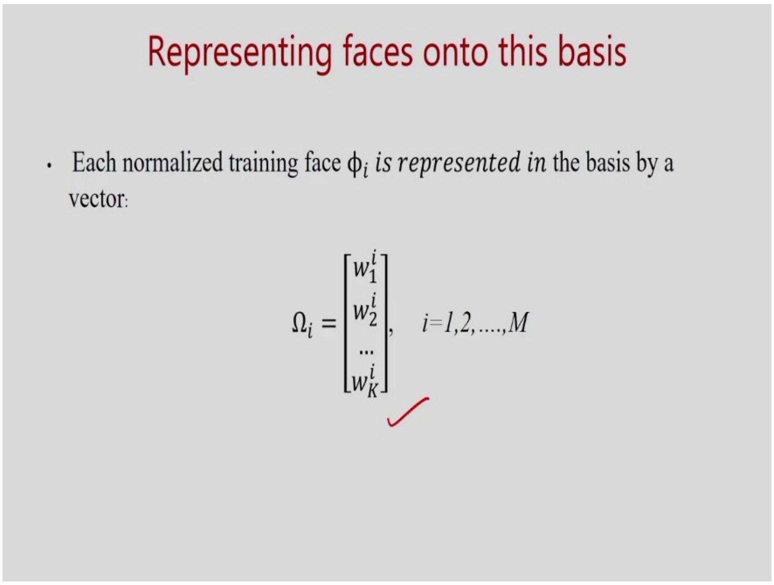
PROJECTION OF TRAINING SAMPLES INTO THE EIGENFACE SPACE

- Each face (minus the mean) ϕ_i in the training set can be represented as a linear combination of the best K eigenvectors:

$$\underline{\hat{\phi}_i} = \sum_{j=1}^{K} w_j u_j$$

(where, u_j's eigenfaces)

= 0.9571 * - 0.1945 * + 0.0461 * 0.0586 *



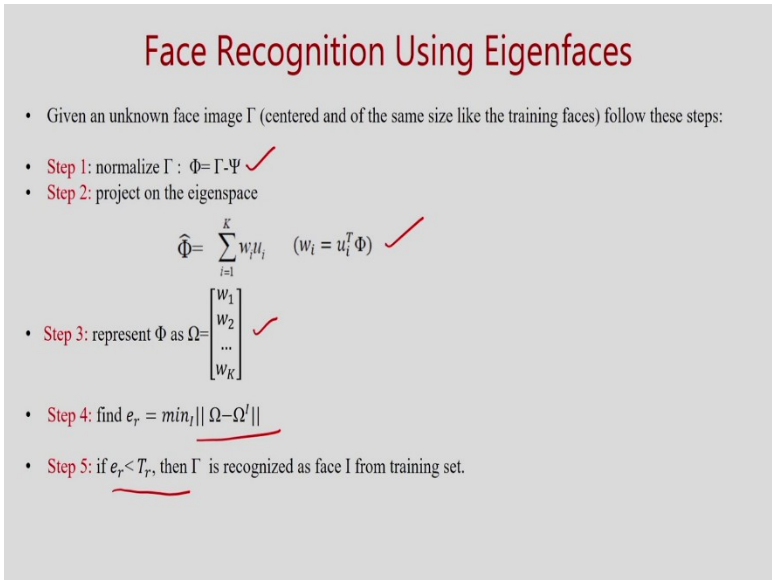
Representing faces onto this basis

- Each normalized training face ϕ_i *is represented in* the basis by a vector:

$$\Omega_i = \begin{bmatrix} w_1^i \\ w_2^i \\ \cdots \\ w_K^i \end{bmatrix}, \quad i=1,2,\ldots,M$$

So, here in this case that means, if you see this slide here these are my widths. So, these are my width vectors. So, representing faces onto this basis.

(Refer Slide Time: 57:06)



Face Recognition Using Eigenfaces

- Given an unknown face image Γ (centered and of the same size like the training faces) follow these steps:
- Step 1: normalize Γ : $\Phi = \Gamma - \Psi$ ✓
- Step 2: project on the eigenspace

$$\hat{\Phi} = \sum_{i=1}^{K} w_i u_i \quad (w_i = u_i^T \Phi) \checkmark$$

- Step 3: represent Φ as $\Omega = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_K \end{bmatrix}$ ✓
- Step 4: find $\underline{e_r = min_l \| \Omega - \Omega^l \|}$
- Step 5: if $\underline{e_r < T_r}$, then Γ is recognized as face l from training set.

And for the testing what I have to do? For a test unknown image, the same procedure is repeated. So, first I have to do the normalization and after this, I have to do the projection on the eigenspace and this is my widths corresponding to the test image, and for a face identification what I have to consider? Only I have to compare the widths, the widths that already I have explained for the training and for the testing also this width I have to compare.

Because already I have the eigenfaces and if this error is less than a particular threshold, then a particular face is recognized or identified. So, this is the face recognition by the principle of PCA. So, in this class, I discuss the concept of the KL transformation, a very important concept the KL transformation and after this, I discuss the concept of the PCA the principal component analysis and after this, I discussed on application. So, how to recognize face by using the PCA? So, let me stop here today. Thank you.