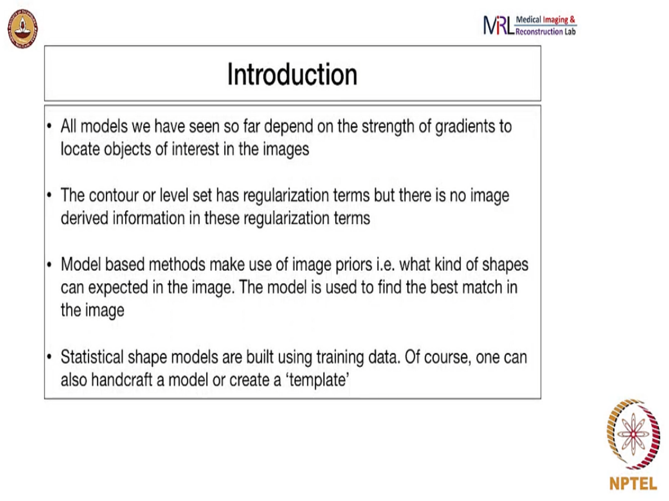
Medical Image Analysis
Professor Ganapathy Krishnamurthi
Department of Engineering Design
Indian Institute of Technology, Madras
Biomedical Engineering Design

Lecture 33

Active Shape Models

Hello and welcome back. So, in this video we are going to be looking at a so called Active Shape Models in a very powerful tool for image segmentation.

(Refer Slide Time: 00:28)



Introduction

- All models we have seen so far depend on the strength of gradients to locate objects of interest in the images
- The contour or level set has regularization terms but there is no image derived information in these regularization terms
- Model based methods make use of image priors i.e. what kind of shapes can expected in the image. The model is used to find the best match in the image
- Statistical shape models are built using training data. Of course, one can also handcraft a model or create a 'template'

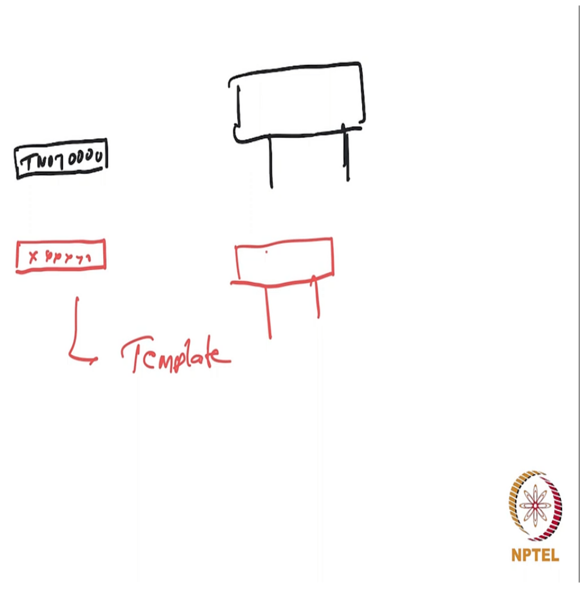
So far, we have looked at models, segmentation models that depend on the strength of gradients to localize objects of interest in the images or for instance, organ segmentation in the case of medical images. So, if you consider both active contours as well as geodesic active contours, the classical snakes and the geodesic active contours, even in the levels set formulation, the localization is done mostly on the base mostly on the fact that the edges, at the edges of the objects of interest have high gradients.

So, in order to regularize this problem, we had smoothness terms, so that the contour does not get too wiggly or too oscillatory. However, what we did not have is an idea of the shape of the object. So, there are no image priors that we have incorporated into these methods. So, we are only looking at, okay, the contour should be smooth and ideally it should be on the edges of the organ of interest, primarily because that is where the strength, gradient strength is very high.

But it would be nice if we can incorporate some knowledge that we already know from prior knowledge in order to help the segmentation. So, the idea behind this Active Shape Model is to incorporate some form of image prior, in the sense, exploit the fact that we do have some idea about what the object looks like. And how is that possible? Here, first we are going to run into this so called training data wherein we have a collection of images, which contain the object.

Now, that object might not have the same force, orientation or size, they might have different scaling factors but they are all the same object. And from those training sets of images, the idea is to construct a model, a shape model or so called statistical shape model, which we can then use as a template. Now, why would we need this? We can just create a template like a toy template that you typically do. Here is a difficulty there.

(Refer Slide Time: 02:34)



So, for instance, let us say you have like car number plates. If you have number plates in a car, so you can have things like this, I know. This box, this is a typical number plate with TN 07, some, some number. So, then you can look nice. Now, if you are going to look for number plates in a car or for instance sign boards and I know as you are driving around, let us you have a sign board something like this, etcetera.

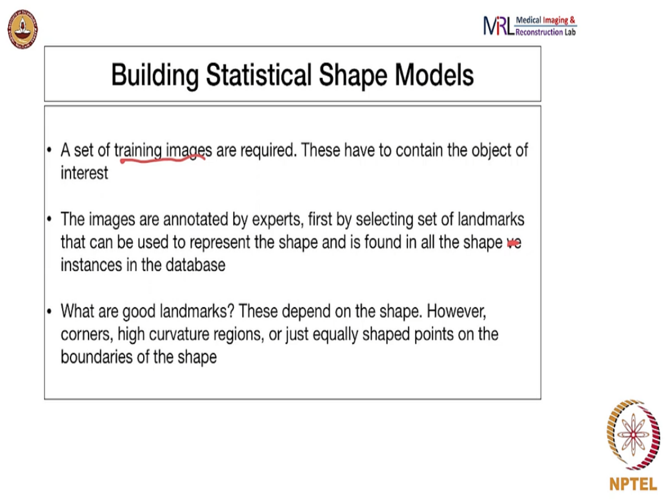
So, then you can always construct a template which has this shape. So, this has something like this, some shape and then you can have some identification here. Similarly, for this you can construct some template. So, this is our template that you can construct and then once you are given an image look for this, you can just scan for this template by doing a cross correlation. And wherever there is high cross correlation that is where you do localization.

However, this works only if the objects are a fixed size and orientation and these are mostly very rigid objects. Now, in many cases, where we are going to do image segmentation, we are looking for slightly more flexible objects where for instance, face detection. Not all faces look the same; they are all of slightly different shapes and sizes.

So, we should be able to incorporate, take that into account. So, that is another thing that we have to worry about. So, in the event where there is a huge variability, when I say huge variability within the limits of what we can may model. So, we cannot have like very amorphous shapes, etcetera. But for instance, like you said face is a good example, organs are good examples of shapes, which have some variability, and we cannot use a fixed template.

Now, in those cases, what we are looking for is a model that encapsulates the range of shapes for that are possible. Range of shapes and deformations that are possible on a basic shape. And that model if you have then somehow we use that model to find out where that object is in the picture in terms of the shape parameters and then eventually if I know achieve the segmentation of localization. So, that is the idea that is the general idea behind the active shape model paradigm.

(Refer Slide Time: 4:58)



Building Statistical Shape Models

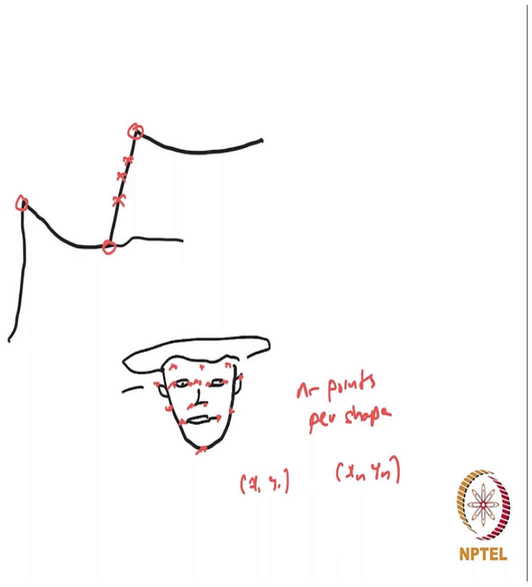
- A set of training images are required. These have to contain the object of interest
- The images are annotated by experts, first by selecting set of landmarks that can be used to represent the shape and is found in all the shape we instances in the database
- What are good landmarks? These depend on the shape. However, corners, high curvature regions, or just equally shaped points on the boundaries of the shape

So, let us get into the process right away. The first step is to build the statistical shape model. Now, how do we go about building this so called shape model. So, first, we need a set of training images that is absolutely required as a training set or you can say example set. In this case, I would really call it training set and I am borrowing freshology from machine learning literature. But we should call them example images, a bunch of example images where we have ideally have cropped out the object of interest.

And of course, needless to say they of course contain the object of interest but where we also zoomed in and cropped it out, let us say. And the images are annotated by experts. Here, the annotation is twofold, one is like, of course, we are localized object, second in the picture and crop it out and second is to select a bunch of landmarks, control points if you want to call them that can be used to represent the shape.

And these control points or landmark should be found in all the shapes, shape instances in the database. There is a spelling error here, delete that. So, then what are the good landmarks? How do you decide what these landmarks are? Again basically, they depend on the shape and with whether these are sharp corners, high curvature regions, sometimes we just put equally shaped points on the boundaries of the shape.

(Refer Slide Time: 06:28)



So, let us just look at typically, I am going to just draw some a boundary of a shape and just tell you what, how we can annotate them. So, for instance, if you have something that goes like this, you have a sharp point here and then maybe something of this sort and you have this extending. Let us say, this is some shape, the boundary of some shape, it does not matter what it is.

So, then for annotation, we might have to choose these points, these points. And maybe because here we want to make sure that there is a connection, we can just choose some intermediate connection points etcetera. An easier example is if you have a face something like this, you have somebody's face and maybe and wearing a hat, there is eyes.

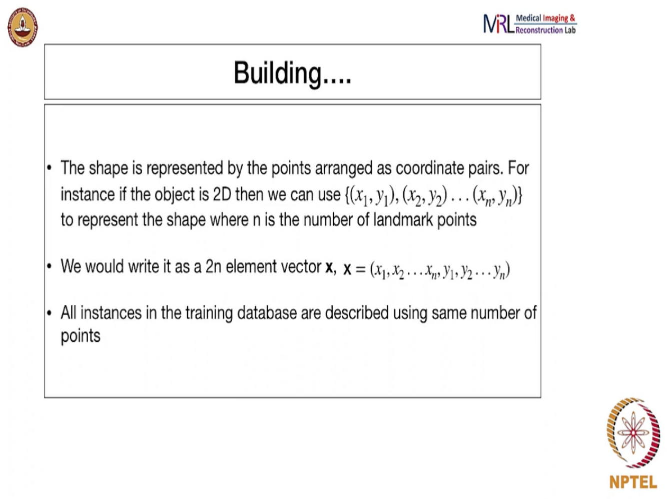
Then you might have to do landmarks. The landmarks will be something like, let us say here, there, here, there maybe we can have some points here, the tip of the nose, maybe some points there, be some some near the temple region, we have here on ear. I have not put in the ears here.

So, somebody who has some decent ears kind of not very human but still now edges of these ears etcetera. So, that will give you a nice template, a bunch of, a bunch of points that tells you, then you can also do something around the face. So, this gives you the control points that describe a shape.

So, first you need to have this annotation done so that you can have, for every shape, all these points should be present. So, that also for instance, in our case let us say, for this, in this lecture we are going to consider like the say n points per shape. So, basically, points of the form x1 y1 so on and so forth to xn, yn.

So, so that, but then all shapes in your database that you have with your so called example data or training data, for every point in the training data you need to, for every shape in the training data you need to have n points annotated on them and they should all correspond to the same landmark. So, that is the idea.

(Refer Slide Time: 08:52)



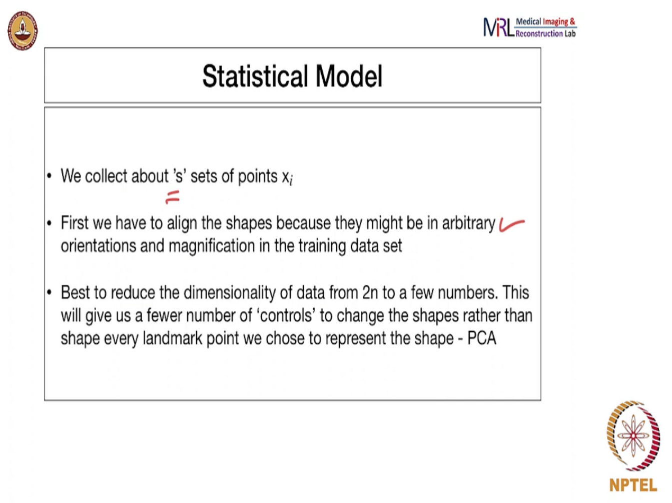
Building....

- The shape is represented by the points arranged as coordinate pairs. For instance if the object is 2D then we can use $\{(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)\}$ to represent the shape where n is the number of landmark points
- We would write it as a 2n element vector **x**, $\mathbf{x} = (x_1, x_2 \dots x_n, y_1, y_2 \dots y_n)$
- All instances in the training database are described using same number of points

So, like we saw the shape is represented by the points and is arranged as coordinate squares. So, for 2D objects, it is what we are going to talk about in this lecture, we can use in this kind of an ordered pair approach x1 y1 x2 y2 up to xn yn, we represent the shape and we typically use small n as the number of landmark points or for the sake of analysis, we typically write this down as a 2n element vector. That is x1 to xn and y1 to yn.

And of course, like we said all instances in the training database, once again you can call it an example data, should be described using the same number of points. So, make sure that the landmark points are present in all instances of the shape in your database. So, that is very important. So, once you have this, now once you have this ordered pairs, let us say you have s data points and you have 2 n, for every shape you have 2 n element vector.

(Refer Slide Time: 10:03)



Statistical Model

- We collect about 's' sets of points x_i
- First we have to align the shapes because they might be in arbitrary orientations and magnification in the training data set
- Best to reduce the dimensionality of data from 2n to a few numbers. This will give us a fewer number of 'controls' to change the shapes rather than shape every landmark point we chose to represent the shape - PCA

Then next, we move on to how we build this so called statistical model. So, the first algorithm here, the first step here is we have to once we collect the set of points. So, for every shape, we have so here, once again I am going to be abusing notation here a bit. So, ideally, we need to have 2 indices, one to index the points in a shape another one to index the instance of the shape.

So, but depending on the context I hope that you will understand and I will also make it a point to explain. So, we collect all the points corresponding to all the instances of the shape. So, let us say there are s data points in our training data set, for example data set. Then first step we have to do is to align the shapes. Why is an alignment required?

Because these shapes may be in arbitrary orientations or pose and magnification that is scaling in the training data set. So, we do not want that. So, what we want this is a variation of the shape but we do not want to have them in different poses. The idea is we can always estimate the pose and the scaling.

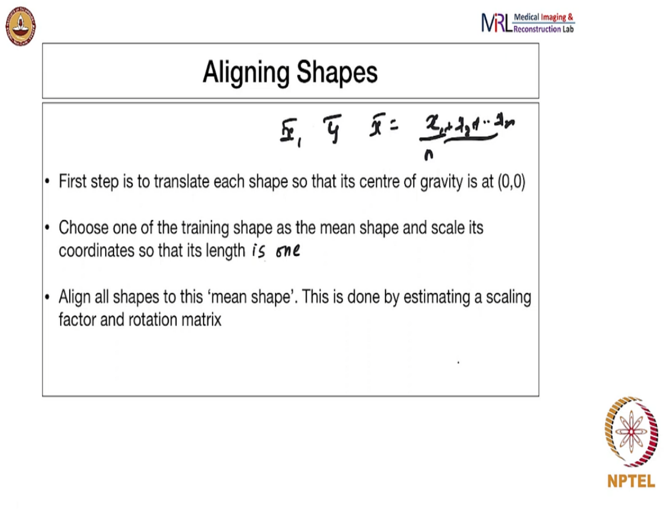
What is hard to estimate is the natural variability of the shape. So, we want our model to encapsulate just the variability of the shape itself and not its pose and magnification or scale. And that is the one step that we have to do. So, this is the first algorithm. Once we have a model, statistical model then it is best to reduce the dimensionality of the model. So, what do you mean by reducing the dimensionality?

So, here think about it. So, you have a shape, which has n data points. So, which is basically 2n numbers are used to describe that shape. N data points, each point is an x comma y. So, but if you want to let us say modify the shape a little bit, say one of them deform it a bit then we have to work on all these n points. So, all 2 n numbers have to be changed.

And of course, we have to, even if you want to make minor minor say even if you change one point at a time maybe you will not be able to express yourself better, so, you might have to change a bunch of points at the same time. So, to make this job easier, we have to do this dimensionality detection, so that we have a very few numbers that we can fiddle with, you can call them shape parameters, so that moving, changing one of the numbers will lead to some tuneable shape transformation. So, we will see that later on.

But the idea is again; if you understand this basically, you have a lot of x y coordinates describing your shape. And if you want to change the shape or deform the shape a bit then we have to change all the n x y coordinates. Instead, if you reduce the dimensionality of that data then you do not have to change so many numbers but change only a few numbers to get the kind of shape deformations we want.

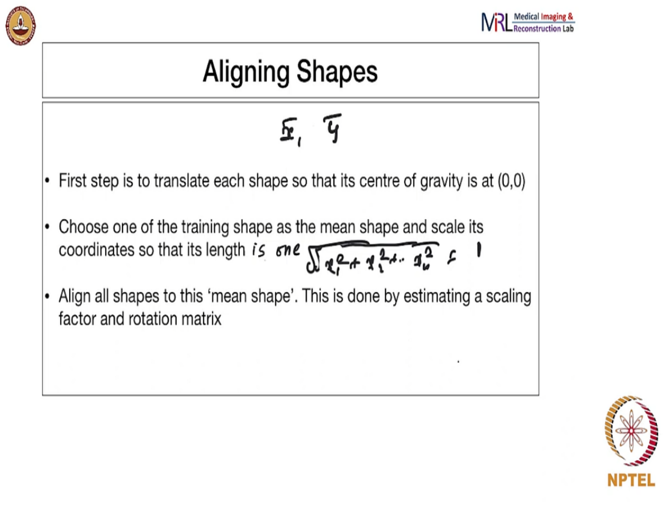
(Refer Slide Time: 13:19)



Aligning Shapes

$\bar{x}_1$, $\bar{y}$ $\bar{x} = \frac{x_1+x_2+\dots x_n}{n}$

- First step is to translate each shape so that its centre of gravity is at (0,0)
- Choose one of the training shape as the mean shape and scale its coordinates so that its length is one
- Align all shapes to this 'mean shape'. This is done by estimating a scaling factor and rotation matrix



Aligning Shapes

$\bar{x}_1$, $\bar{y}$

- First step is to translate each shape so that its centre of gravity is at (0,0)
- Choose one of the training shape as the mean shape and scale its coordinates so that its length is one $\sqrt{x_1^2+x_2^2+\dots x_n^2} = 1$
- Align all shapes to this 'mean shape'. This is done by estimating a scaling factor and rotation matrix

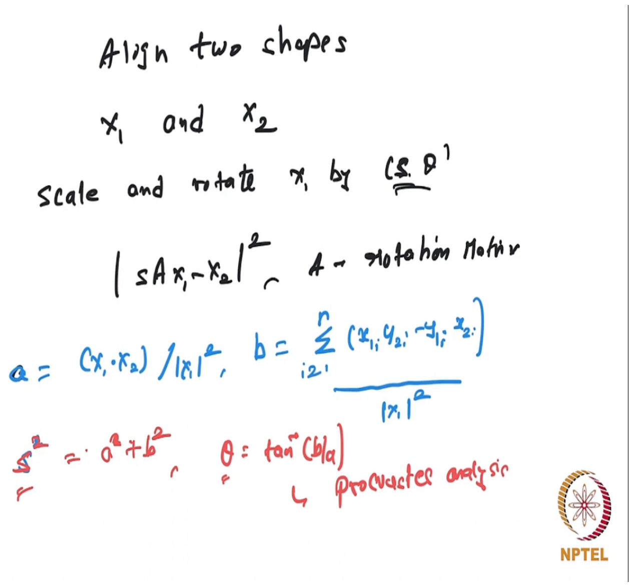
So, the first step in aligning shapes is to translate each shape, so that its centre of gravity is at 0 comma 0. So, basically, the way to do that is to estimate the centroid by just calculating. Let us say x bar and y bar for each shape which is nothing but, so, x bar would be the average of all the x coordinates total by total number of points.

So, x1 plus x2 plus up to xn by n, we can write this better in a second. So, this is x bar and y bar are calculated. So, that x bar and y bar are calculated as the mean of the x coordinates and mean of the y coordinates and then you subtract that out from the individual coordinates. The second step is to choose one of the training shapes as a mean shape and scale its coordinates, so that if you calculate its length, its unity.

So, basically what we want is the square root of… for this particular shape, we can do something like square root of x1 square plus x2 square plus so on up to, there are n points, xn square. This equal to 1. So, you want to do a normalization like that. So, what we are doing here is kind of scaling the shape.

So, take the scale factor out. And the last step is to align all the shapes to this mean shape. And this is, again this alignment is done by estimating a scaling factor and a rotation matrix. And this alignment actually we can oppose in the form of an optimization problem.

(Refer Slide Time: 15:23)



Align two shapes

x_1 and x_2

Scale and rotate x_1 by (s, θ)

$|sAx_1 - x_2|^2$, A - rotation Matrix

$a = (x_1 \cdot x_2)/|x_1|^2$, $b = \dfrac{\sum_{i=1}^{n} (x_{1i} y_{2i} - y_{1i} x_{2i})}{|x_1|^2}$

$s^2 = a^2 + b^2$, $\theta = \tan^{-1}(b/a)$

↳ Procructes analysis

So, let us look at how we can align 2 shapes and then all you have to do is if we have applied this to all the shapes in the database and take, choose one of the shapes as a mean shape and align all the other shapes to it. And then we will just look at how we can do 2 shapes at a time.

So, the idea is we have the 2 shapes, which are centred at the origin that much certain and we wish to scale one of the shapes, let us say x1 by s and then also rotate it by angle theta so as to minimize the loss function. So, let us just write this down. So, we have 2 shapes x1 and x2, 2 shapes x1 and x2 and we want to scale and rotate. Scale and rotate x1 by s comma theta.

So, a small s, theta. And the idea is that way and how do we estimate this (s, θ). It is by minimizing this loss function, $|sAx_1 - x_2|^2$. So, here A is the rotation matrix that much. So, it turns out, then the solution to this optimization, if you solve this optimization problem by basically what we have to do is minimize this loss function with respect to these parameters that we want to estimate and we will end up with the following formula. So, a is, I will use a different colour. So, $a = (x_1 . x_2)/|x_1|^2$, $b = \sum_{i=1}^{n} (x_{1_i} . y_{2_i} - y_{1_i} . x_{2_i})/|x_1|^2$.

So, if you look at this formula, you have both this factor a and b. We are here, the i's refer to the coordinates of the individual points for shape x1 and n shapes x2. So, when you say y2, y2 refers to shape 2, the y coordinates of shape 2 and i is the specific y coordinate of, let us say landmark point i.

So, that is how you have to interpret it. So, the i's actually subscript the individual points in a particular shape. So, if the estimate is a and b, from here on its the scaling factor is on top of this, estimated as, if you want to use a different colour then $s^2 = a^2 + b^2$ and the rotation angle is $\theta = tan^{-1}(b/a)$.

So, if you have 2 shapes x1 and x2 and if you want to bring x1 in alignment with x2 then you have to scale x1 by this following, this s and rotate by theta. So, all of this is, of course, when I say scale the shape etcetera, what we are trying to do is to apply these transformations to the coordinates which describe the shape.

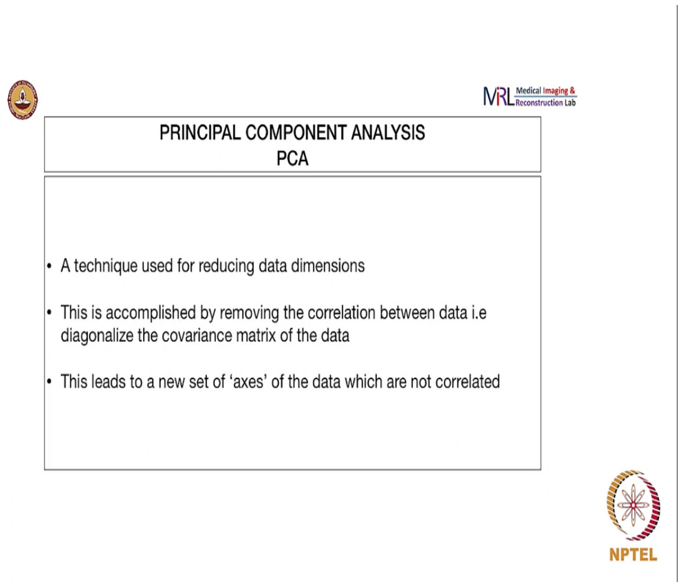
Now, we know how to do that because we have seen some of this in our image registration. So, this is something like image registration but we are only considering a specific set of transformations and this typically applies to situations where we have like a bunch of points describing a shape and this is often used, this is referred to as a Procrustes analysis.

So, once again to reiterate, why do we need to do this? It is because when we are annotating our training database with these landmark points, so all the points in all the data in our training database, so, let us say images basically, will have the object that we want. But they will all be in different poses and scales.

We are not looking to estimate, we do not want to have a distribution over all possible poses. That is impossible but our scaling factors for us. That is not what we want. What we want, again, is to capture the natural variation in the shape of the object that we are trying to segment or localizing the object. So, this is the first step.

So, once we have annotated then we take those points and bring them all into a common coordinate system that is precisely what this is doing. So that now, we have for every control point, they are all in alignment all the control, corresponding control points in every shape that we have annotated is in alignment now. So, that is what you would refer to as a point distribution model.

(Refer Slide Time: 20:52)



PRINCIPAL COMPONENT ANALYSIS
PCA

- A technique used for reducing data dimensions
- This is accomplished by removing the correlation between data i.e diagonalize the covariance matrix of the data
- This leads to a new set of 'axes' of the data which are not correlated

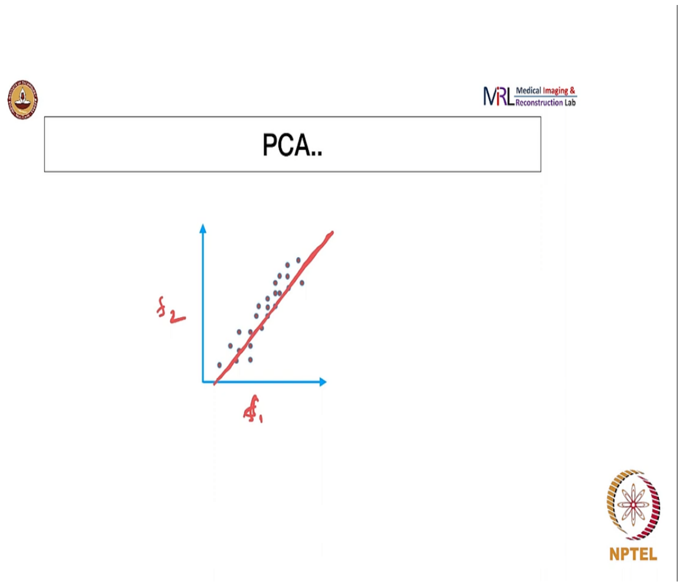
The second step. Now, the second algorithm that is used is the Principal Component Analysis. Now, that we have all the shapes in a common coordinate system. What we want to do is to reduce the dimensionality of the data. Now, we only know that every shape is represented by n points.

And then if you want to deform the shape in any way, we have to deform all the n points that becomes a little bit hard. So, it will be nice if you can, if you know the variations or the variations in the shape that can be controlled by a few parameters. So, by varying those parameters, we can control the shape very effectively.

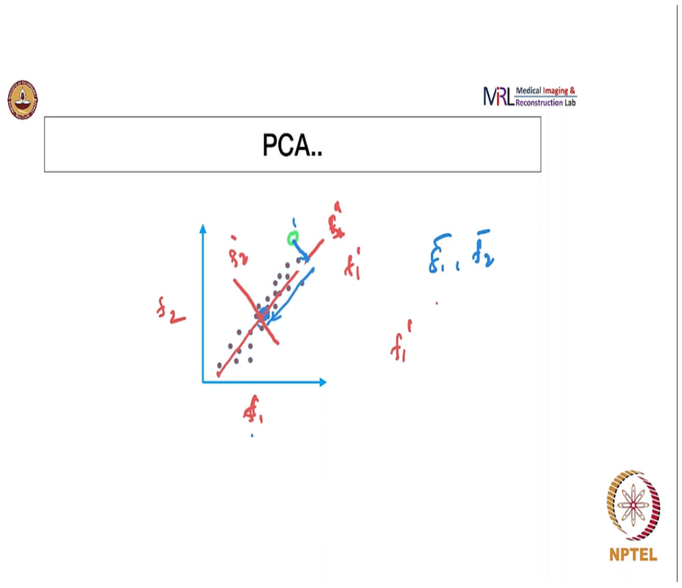
So, and this is accomplished by you using something called the principle component analysis, a technique. In fact, this should qualify as one of a most older machine learning technique and this is typically used to remove correlation between features in data. So, the other way of putting it is to it is used to diagonalize or diagonalize the covariance matrix of the data.

So, when you diagnose the covariance matrix of the data, what you do is you will remove the correlation between the features in your data. So, and in the process, what happens is that you will figure, estimate a new set of axis for your data and you can also interpret the process of principle component analysis as the rotation and translation or the translation followed by rotation of your coordinate system that you are in, into another coordinate system, so that the correlation between the features are removed. So, we look at an example in the next slide, so that understand this a particular scheme better.

(Refer Slide Time: 22:37)



PCA..



PCA..

So, for instance let us look at this data here, which is plotted. This is a toy plot. So, I am going to call this just to not confuse people, let us say there are 2 features. So, I am going to call them f1 and f2. So, these are some in, so many data points, each data points has 2 components f1 and other component is f2. Now, we see that this there is a lot of; there is correlation between f1 and f2. So, we can draw this correlation.

Let us say I will use red, let us say something like this. So, this line kind of captures this correlation between f1 and f2. So, if you know f1, you are kind of estimate f2. So, that is what this means. So, but if we actually have a line like this that goes through I am just hand drawn this. This is very crude.

I have that you know that, that goes in this direction and then you have another fee… another line, let us say which again in this case, it is not drawn properly but let me redraw this so that let us make this clear. So, I am going to indicate with a very large blue thing.

Let us say this is the mean, this is the mean of your data, mean of this is, this point is f1 bar, f2 bar. So, I am going to draw a line that goes through that, another line it goes to this. So, we are going to call this f1 prime, f2 prime. So, this f1 prime f2 prime is a new coordinate system. And if you see, what it does?

What it has done is that this f1 prime, f1 prime captures the maximum variance in your data set. So, because there is a lot of variability in the data across, if you go from, if you go along the f1 prime axis, while the variability along the f2 prime axis is very low. So, now, if you look, this also is a translation and rotation. So, because we have drawn the axis through the centroid of the data set.

So to speak, we have calculated f1 bar f2 bar and that is the centroid of the data set and we have move and we have drawn our axis through that. So, it is a translation wherein we have subtracted out the mean from each of the data and then the rotation of this axis. So, if we do this transformation then we see that we can actually describe our data with just f1 prime.

Because the variability across f2 prime is so low, we just need f1 prime to tell where the data is. So, for instance, if you wanted let us just look at this particular, let us say some data here, I will draw separately, let us say there is a data point here. And if you want to describe that data point then all we need is some projection of this data point on to f1 prime axis.

This projection here and the using this point from the origin, what is this distance, we can describe this green point. So, instead, but then previously to describe this green point we needed both f1 and f2. Now, with the new coordinate system where you are translated and rotated, we just need f1 prime. So, just we have reduced from dimensionality of 2 to dimensionality of 1. This is precisely what a PCA does.

Another thing it has done is to decordinate, in the sense; f1 prime is not correlated to f2 prime at all. In the first case, f1 and f2 are correlated but now, f1 prime and f2 prime are not correlated at all. In fact, it just has figured out the axis. So that I have drawn the axis so that maximum variance is along f1 prime axis f2 prime axis has very little variance.

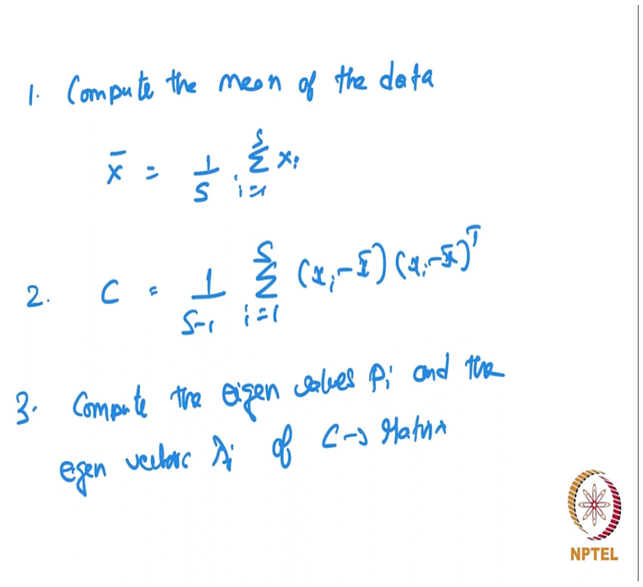
So, this is what in general a PCA accomplishes. What it does is, if you have a coordinate system, the coordinate system once again need not be spatial. In our case, it is a spatial coordinate system. We have a point cloud; we have a bunch of points describing a shape. And we have a point cloud describing a shape and what we want to do is to reduce the number of points that describe our shape.

So, in general terms, we have a bunch of features describing our data and it would be nice, if we can reduce the number of features used to describe your data by removing the correlation between the data. So that we only keep the features or the directions which have maximum variance and we can actually ignore directions where the variance is low because it does not add any information, any new information. So, this is the function of PCA.

And we are going to do this for our shape model. Remember, our shape model is nothing but a statistical shape model, nothing but collection of x comma y coordinate pairs drawn from different instances of the shape that we are trying to describe. So, for every landmark point, there are a bunch of x comma y coordinates.

So, since, we have that, how do we now reduce those. Let us say we have n points, 2 n numbers actually, n points, 2 n numbers. Can we reduce this to a number of, smaller number of parameters that can be used to control the shape? And that is exactly what PCA accomplishes.

(Refer Slide Time: 28:02)



1. Compute the mean of the data

$$\bar{x} = \frac{1}{S}\sum_{i=1}^{S} x_i$$

2. $C = \frac{1}{S-1}\sum_{i=1}^{S}(x_i-\bar{x})(x_i-\bar{x})^T$

3. Compute the Eigen values p_i and the eigen vector λ_i of C-matrix

So now, how we are going to see, how we are going to actually calculate this the Eigenvalues and Eigenvectors. So, I am just going to walk you through the PCA calculation steps very very briefly. I urge you to again look up some text. There are lots of videos online that you can consult to understand this actual computation. So, the first step is to compute the mean of the data.

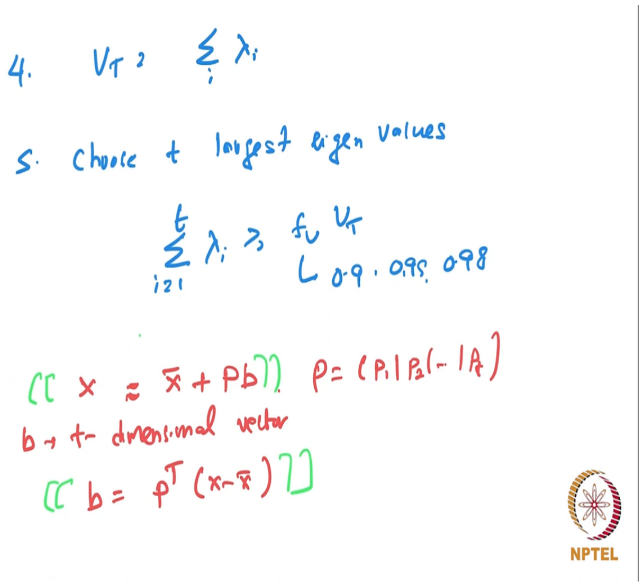
So, we have to subtract the mean out for this to make sense. The reason is because we are trying to look at oscillations around the mean, not the absolute values. So, we calculate this x bar as if you sum over. So, once again there is some abuse of notation, s, we use for scaling factor also. So, I am just going to use a slightly maybe a larger S.

This is the mean of the coordinates. Then again, if you compute the covariance of the data, so, a covariance of data I am going to call it C. It is 1 over S minus 1 summation i equal to 1 to S, large S and you have x i minus x bar times x i plus x bar transpose. So, now, the third step is to compute the Eigen values p i and the Eigenvectors lambda i of C matrix. So, this is the calculation that you have to do.

So, once you have this C matrix, this is the covariance matrix of your data, like I said the idea behind the calculation of the PCA is to make sure you diagonalize your covariance matrix. So, then you are trying to actually estimate axis where which are not, which will make sure the data does not appear correlated. So, that each of the axes are uncorrelated.

Now, the next step. One have the Eigen values and Eigen vectors. The idea is that each of the Eigen values tells you the variance of the data about that axis, about that axis from the mean, about the mean. So, that is why we subtract the mean out otherwise the computation would not make sense. We subtract the mean out and we only have this variation in the data from the mean and that is what the Eigenvalue gives. So, you can calculate the total variance.

(Refer Slide Time: 30:59)



4. $V_T = \sum_i \lambda_i$

5. Choose t largest eigen values

$$\sum_{i=1}^{t} \lambda_i \geq f_v V_T$$

$\llcorner$ 0.9, 0.95, 0.98

[[$x \approx \bar{x} + Pb$]] $P = (P_1 | P_2 | \dots | P_t)$

$b \rightarrow$ t-dimensional vector

[[$b = P^T (x - \bar{x})$]]

So, the fourth step is total variance, VT is the summation of all your Eigenvalues. And ideally, we want to choose the t largest second value. So, in the example I showed you, I just it was in 2 dimensions, we had f1 and f2 and then we did f1 prime and f2 prime. So, that we have 2 prime but very small oscillations and uncorrelated f1 prime. So, we ignored it. We threw it away because the variance there was very little.

So, we do something similar. There are more dimensions to choose but what is the criteria? We choose the t largest, choose t largest Eigen values which capture, let us say, very large percentage of the variance. So, the way you would write it down is you would say summation i equal to 1 to t lambda i is greater than some fraction of the variance. But this you would set it as let us say 0.9 0.95 0.98, depending on what gives you good accuracy.

Typically, let us say 98 percent is what is used. Even at that high threshold, you lose a lot of the axis when it is not necessary. Now, once you have this, what is the point? As I said, we saw that when you did the 2D example, we saw that once we have the f1 prime f2 prime axis, we are able to express every point just in terms of the f1 prime axis by taking a projection of any arbitrary point on to the f1 prime axis.

So, that is pretty much what we are going to do is we are going to project every x onto the t axis that we have left. So, and that, once we have projected them that gives you the sets of coordinates. We call them b that is, that can be used as shape parameters. So, the idea is that we can approximate any training set data x as approximate as the mean plus p times b. What do these mean?

The p is nothing but this particular matrix where each column is an Eigenvector but it has only t number of columns. So, it has t Eigen vectors and b is a t dimensional vector. So, b is just like I said, b is very similar to that f1 prime we saw. When 2D, it was easy to see but in more dimensions, it is much harder to see. So, b is a t dimensional vector give and b is a t dimensional vector. And how did we get b?

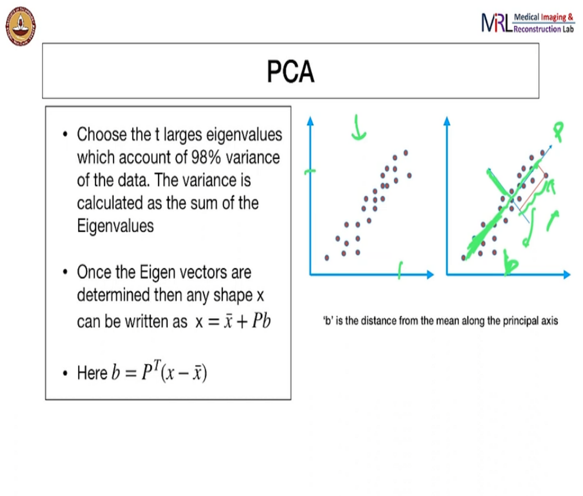
In that case, in the example case, we got this f1 prime by projecting an arbitrary point on to the f1 prime axis. So, similarly, we would have to project any arbitrary x onto the t axis and that is done by the following expression. p transpose x minus x bar. Of course, this is done after doing the, after subtracting out the mean.

So, if you see that this could be consistent dimensionally. This is a good exercise I would not do it. Now, but I would urge for you to look at what will be the dimensions for each of them. Make sure that they are dimensionally consistent make sure it is meaningful. So, the Eigenvectors, what this p Eigen vectors represent is a rotated coordinate system and the parameters b or the new points that those are the models.

So, the if you change b, you can change in the models. So, they are the most significant coordinates of the shapes, in this rotated shape. So, this is the purpose behind doing the PCA. Because now, we have this ways to, by you can look at each one of these b's as some kind of button that you can press up or down on and so that you can change it.

By changing its value, you will change an aspect of the shape. So, let us understand, it is slightly better, we have looked at the f1, f2 prime example. But we can also I mean just to reinforce this; we look at it one more time.

(Refer Slide Time: 35:28)



PCA

- Choose the t larges eigenvalues which account of 98% variance of the data. The variance is calculated as the sum of the Eigenvalues
- Once the Eigen vectors are determined then any shape x can be written as $x = \bar{x} + Pb$
- Here $b = P^T(x - \bar{x})$

'b' is the distance from the mean along the principal axis

So, we will consider this picture. So, for instance, this was our original point cloud distribution and we had constructed the new axis. This is our p Eigen vectors, this is our p. So, any arbitrary point, if you look, let us say this is an arbitrary point that we want to project. f is some x. This we can project to here to x prime. And what is b?

B is nothing but the distance along this axis. It is a coordinate of that point in the new axis. So, this is this distance. This distance I have indicated here, that is your b. So, in your new system of coordinates and what is your coordinate values that is exactly what this does. So, it is very, in 2D so, you have to understand, of course, when you go to higher dimensions, it is not easy to visualize.

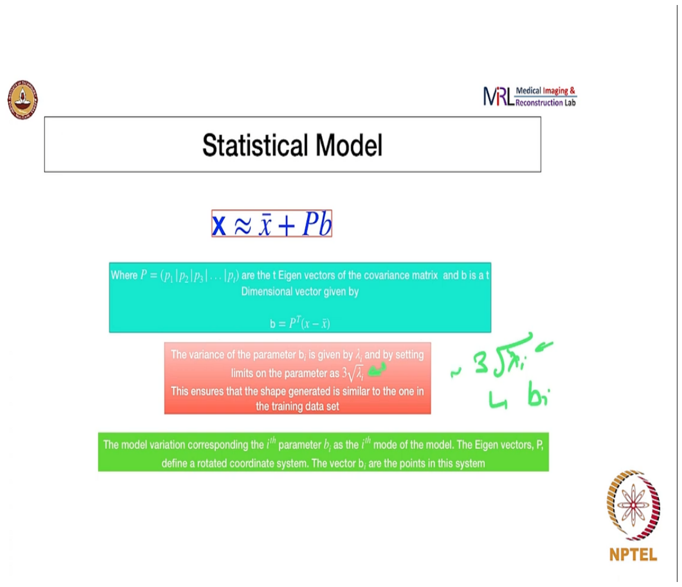
But this is exactly what it is doing. So, in the problem, if you look at it from the point of view of diagonalizing the correlation matrix, if you think of it this way, if you look at this, this situation, there is correlation between these 2 data sets. Because you can always draw a straight line and you can use that to predict one from the other. So, that is possibility.

But here if you look, since our axis is aligned along this, along the maximum variance, if you take this axis, right here this axis passing through the origin, through the mean of the points, this is uncorrelated with this axis. And the variance along this axis is much much lower compared to variance along this axis.

So, we, in the sense, when I say this axis, this horizontal axis right here is where the maximum variance is. And we can safely ignore the other axis without losing too much information. So, that is the idea behind doing the PCA. So, now, we have done 2 things. We have taken… we annotated the training data with control points. We have aligned all the annotated training data to a common coordinate system.

And then we have done PCA to reduce the dimension of your data so that we can now describe the shapes using a fixed set of parameters, call them b, if we call them the shape parameters. Now, how do we use this to identify objects in a new testament that is what we are going to see in next few slides.

(Refer Slide Time: 37:53)



Statistical Model

$$\mathbf{x} \approx \bar{x} + Pb$$

Where $P = (p_1|p_2|p_3|\ldots|p_t)$ are the t Eigen vectors of the covariance matrix and b is a t Dimensional vector given by

$$b = P^T(x - \bar{x})$$

The variance of the parameter b_i is given by λ_i and by setting limits on the parameter as $3\sqrt{\lambda_i}$
This ensures that the shape generated is similar to the one in the training data set

$\sim 3\sqrt{\lambda_i}$

$\hookrightarrow b_i$

The model variation corresponding the i^{th} parameter b_i as the i^{th} mode of the model. The Eigen vectors, P, define a rotated coordinate system. The vector b_i are the points in this system

Just to recap a bit, this a couple of points that I would like to reiterate here. So, now, we have the shape parameters b corresponding to each of the axis, that new axis that you have estimated from the PCA. Now, we know that the lambda i which are the Eigen values corresponding to each of those new Eigenvectors or the coordinates.

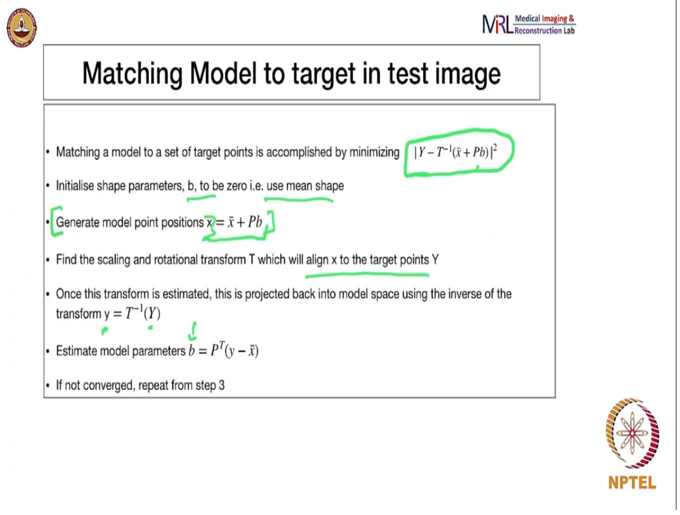
They describe the variance along that axis. So, what this PCA also gives you is this parameter, $3\sqrt{\lambda_i}$ which, I will write this here $3\sqrt{\lambda_i}$ for every b i corresponding to a Eigenvector, we can actually estimate the $3\sqrt{\lambda_i}$ as some kind of a… this is actually 3 times standard deviation, if you think of it that way.

So, that we know how much b i can vary. So, that gives you some limit. So, we cannot arbitrarily vary b i. So, you make sure that you always vary b i within these limits. So, that is a good way to actually adjust shape. So, this also tells you a drawback or the constraint of the method because this 3 square root of lambda i comes from the original data which is our x and y coordinates.

So, what it means is that you cannot have shapes that are too different or outside the bounds described by your training data. So, if you have like an extreme version of your shape, whatever that, however you qualify extreme and then it is not going to be, this method will not work.

So, but another way of looking at it is that it gives you a very systematic way of imposing constraints on b i. How do you know, its maximum this is 3 sigma is here. So, now you know, how much, what kind of constraint to put on b i. Because some point we will be estimating the b i because that is what tells you what the shape is. And we want to make sure that it would not go beyond these limits.

(Refer Slide Time: 39:53)



Matching Model to target in test image

- Matching a model to a set of target points is accomplished by minimizing $\boxed{|Y - T^{-1}(\bar{x} + Pb)|^2}$
- Initialise shape parameters, b, to be zero i.e. use mean shape
- Generate model point positions $x = \bar{x} + Pb$
- Find the scaling and rotational transform T which will align x to the target points Y
- Once this transform is estimated, this is projected back into model space using the inverse of the transform $y = T^{-1}(Y)$
- Estimate model parameters $b = P^T(y - \bar{x})$
- If not converged, repeat from step 3

This last piece or say the penultimate piece is to match the model to a target in test image. So, first let us just look at this problem, as how would you match a model to a set of target points in the test image. Now, we assume that target points are given. We will see how to get those target points later.

First, we will see that there is a bunch of target points that you have somehow figured out in the image which hopefully will correspond to the shape that you are looking for and you what you want to do is take your model. Your model is now described remember by b. You want to do some rotations and translations on your b.

So that the shape that you have, that model that you are working with matches the y in the data. So, basically, you are trying to solve this optimization. The way to solve this maximization problem is the following, is basically you take a shape parameter. Initialize your shape parameters by setting b equal to zero.

So, that is you start off with the mean shape and you generate the model point position using this expression. You have the p axis. Now, we do this. Now, find a scaling rotational transform which will align x to the target points y. So, you will do scaling rotation and also translations, both all 3 of them. I think I left out translations here. Please, add them.

So, this is a sum of, this would be a rigid body transformation that you are looking at but you are doing it to a point cloud rather than to objects. So, find this rigid body transformation that will align x to y. So, once you have estimated it, you would want to project this back into the model space.

Now, you have taken your model b, b is your shape model. Now, you are projected into x space here, put it back in the image. Now, aligned it with the target points y using and that you do by doing some transformation, estimating a transformation. And then you take the transformation, apply it to your y point, so that you get, you are now putting this into the model space.

So, in the model space, now you estimate b. And this b describes your shape. If this is a, and you repeat this till convergence. I still repeat from step 3. So, you have to start from here. So, for this b, generate a new shape and then put it back in the picture, align, estimate the transformation, which will align the shape with the target points and we keep doing this again and again so, till you converge.

So, that way now, what you have done is you taken your model shape and arrived at another shape, which is similar to what is there in the picture that has been given to you. So, now, you can take that model parameters and now, we have a description of the object in the test image in terms of the b's. So, that is the power of this technique. So, now, the question, I am sure in your mind would be, what is this y, where, how do you get this y? So, because intuitively what are you expecting? What do you want to do?

(Refer Slide Time: 43:16)

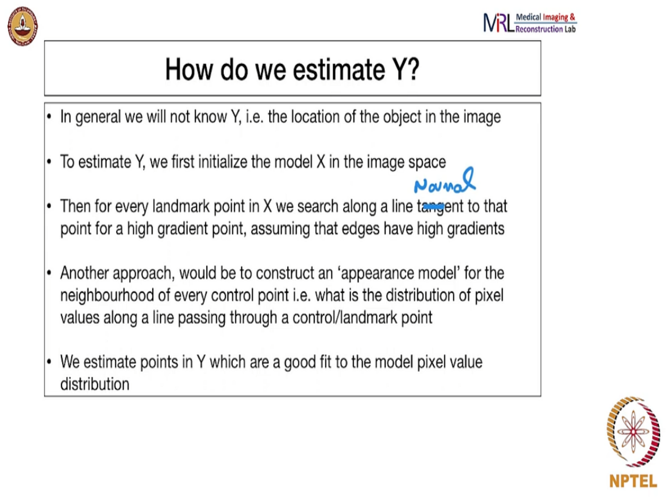
Let us draw this so that we all with the same page. So, ideally you have this picture, big picture, let us say a bunch of, whatever some name, birds flying something and there is a cloud, etcetera and then there is a guy standing there with a face that says Roy's face. So, you have this guy here. So guy we drew earlier v as a different ear and this is guy and we want to.

Now, we have our shape model, we saw that we did our shape model. So, let us say our shape model looks something like based on this b. We have now projected. Now, let us say our shape model looks like this. This way, this is our shape model, so it is more like a monkey but it is okay. This is a shape model.

And now we want to align this to here by estimating a rotation and translation. That is what we do. So, we would align this. Now, how do we know where this is? We do not know. We would not know where this red face is in the picture. So, where would we put this guy, so that we can estimate, rotations and translations and scaling.

So, this is very similar to image registration except that now like I said we are trying to align a point cloud to another set of control points. So, there are, this set of control points here are estimated by or given by y and we want to know how to get that y. So, that is the next step.

(Refer Slide Time: 44:49)



How do we estimate Y?

- In general we will not know Y, i.e. the location of the object in the image
- To estimate Y, we first initialize the model X in the image space
- Then for every landmark point in X we search along a line ~~tangent~~ Normal to that point for a high gradient point, assuming that edges have high gradients
- Another approach, would be to construct an 'appearance model' for the neighbourhood of every control point i.e. what is the distribution of pixel values along a line passing through a control/landmark point
- We estimate points in Y which are a good fit to the model pixel value distribution

So, then how do we estimate y. Now, where in the picture is the object, we do not know. Because, if we know the object is in some location in the picture then we will take our b's, transform it into image space and initialize it somewhere close by, so that it is easy to estimate the rotations, translations and scaling.

But we will not know that because we would not be able to see that some in a semi-automatic way, we can probably know but otherwise we would not know. So, first we initialize the model in the image space. Then for every landmark point in x, we search along a line normal, not tangent, sorry, we search along a line which is normal and for that it is basically draw a tangent, you search along normal.

So, it is around normal. There is an error in the way I wrote it. So, please, take that into account. So, then for every landmark point, we search along with normal to that point for a high gradient point assuming that the edges have high gradient. So, this is again, we are still again, once again back to using gradient, which is like a bad thing, which you want to avoid but there is no other good marker right now.

So, we look for, so, I will, let us draw this in the next page, so that next slide, so that you know what I am talking about. So, for every landmark point, you draw a normal to that point. What do you mean by normal? The normal is defined by it being 90 degrees to the tangent at that point and we have obtained once again how do you do the tangent, we assume that, that the shape model can be seen as a boundary of an object.

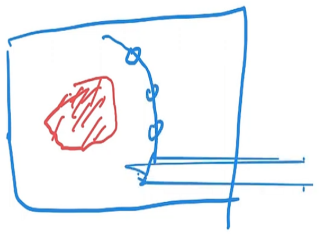
So, it is like a curve. So, we can have a tangent at every point and to the tangent, we draw a normal. So, you search around the normal till you hit the high gradient pixel and that is your y. There is another approach, which is again slightly more sophisticated than this. So, now, we have this shape model. We can also do this so-called appearance model.

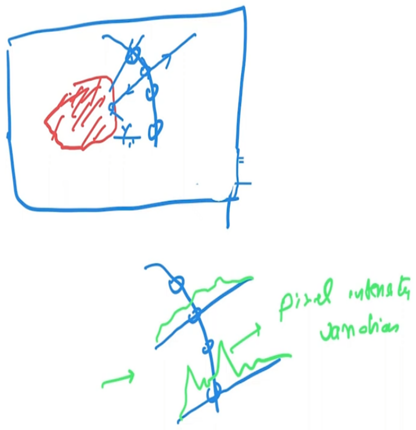
So, let us for every landmark point in the neighbourhood of the point, so, you draw a small line, which is again normal to the tangent at that point. You take a line and you look at the pixel profiles there and then you can construct a statistical model of the profiles around the control points. This here, when I say the profiles, I am talking about the intensity profiles.

So, I will look at the pictures in the next slide. So, we have intensity profiles around every landmark point. And you search again along the normal till the intensity profiles match. So, that is another way. Instead of just looking for, let us say a strong gradient; you can look for similar distribution of pixel intensity values in and around the contour point.

So, that much you can do. So, once, we then we hit then we get this y. So, we can now have a bunch of y's or the y that or capital Y is, you call it, which describes a bunch of points, which is most probably the shape that we are looking for. So, once again, so, let me just draw this.

(Refer Slide Time: 47:46)





Let me know what you are talking about. Let us say, we have this is a picture this is our image. So, let us say there is an object in there. Let us use red. There is some object in there. This is object. This object we are looking for possibly. And then we have our shape model. Shape model is like this and these are our control points. So, from b, we have got this.

So, the idea is now to, let us say we put one more point here. The idea is now to look at, let us say there is a tangent here and then we look at the normal along this direction. So, you search along this direction, you can search on both sides, typically, either way is fine. So, you would go along till you hit, let us say this is a high gradient point. Now, you mark this as some y i.

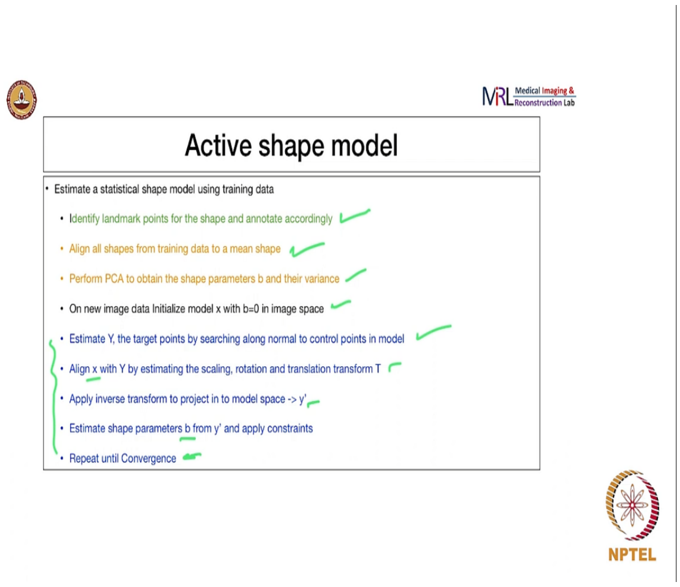
You do that for every control point. From here, you draw some normal. Similarly, from here. So, that you have an idea that from every point on your model curve, you draw a normal and then you search along that normal. That gives you an idea of pattern. So, the next step is, so, once again, if you are looking at this, other way of looking at it is, I was talking about.

You have these control points. Now, along the normal, you can also look at intensity variations. In the training data, you can look at intensity variation. So, this is again pixel intensity variation. Now, I am loosely using this six-pixel intensity variation term. Ideally, you will have to take for every point.

So, you have a bunch of training data sets. Right. Yes, in our case. Then you will have bunch of these intensity profiles, maybe all you can do is, you can in fact construct some distribution from this, statistical distribution from this like we did for the point cloud for the active shape model, the shape model and look where these things would match.

So, this is another way of doing it. So, here in this particular approach, you can get rid of the dependence on the gradient. On top of this, you can also look at high gradients and you can say, once I hit eye gradient I also look at whether the pixel profiles they are match. Pixel intensity profiles, they are match at all. That is another way of looking of finding out y's. So, now, we will put this all together.

(Refer Slide Time: 50:30)



Active shape model

- Estimate a statistical shape model using training data
  - Identify landmark points for the shape and annotate accordingly
  - Align all shapes from training data to a mean shape
  - Perform PCA to obtain the shape parameters b and their variance
  - On new image data Initialize model x with b=0 in image space
  - Estimate Y, the target points by searching along normal to control points in model
  - Align x with Y by estimating the scaling, rotation and translation transform T
  - Apply inverse transform to project in to model space -> y'
  - Estimate shape parameters b from y' and apply constraints
  - Repeat until Convergence

Putting this all together. So, estimate a statistical shape model using training data. And the way to do that is to identify landmark points for the shape and annotate accordingly and make sure that every shape you are annotating or every instance of the shape you're annotating in your training data set you have the same number of points.

Align all the shapes removing the, what can I call, scaling and post dependency. So, that, those 2 you have to remove. And perform PCA to obtain shape parameters. So, now you have a bunch of points. There is too many of them. You just reduce the size, the feature set and to boil them down to a few parameters b, we call shape parameters and we also have a bound on how much they can vary.

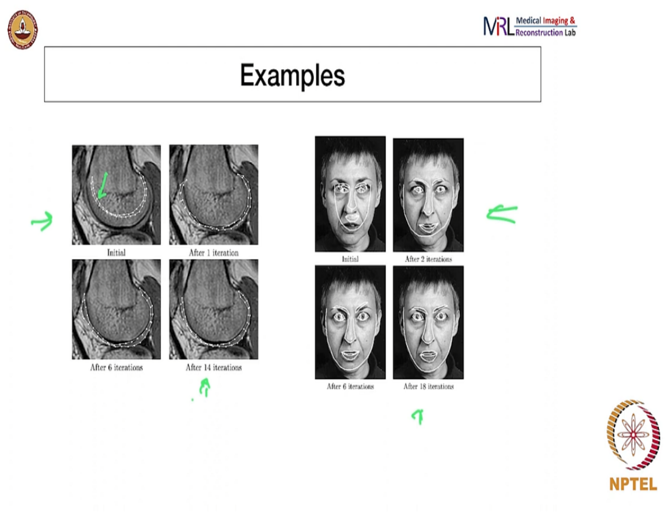
So, on the new image data, initialize some model with b equal to 0. So, take the mean, mean picture or one of the reference images, reference shapes that you have. And then put it on the on the image space. Estimate y, the target points by searching along the normal to the control points in the model. This, the different ways of doing this, we saw that.

And then you align x with y by estimating scaling, rotation and translation transform. Apply this inverse transform. Project to model space. Estimate shape parameters. Repeat this until convergence, especially, the blue plots. This is how you would do this. So, it is a kind of a complicated algorithm but I have, we put this in here because it is a natural continuation to what we have seen so far.

So, it says basically, the active contour models, the geodesic active contours and the level set based techniques, which all start from functional; they have some constraint on the curve, on the curve that does the segmentation. Either you can embed the curve in a level set or you can just work with the curve itself in the image space. But either way, there are some constraints on the curve, which are basically nothing, regularization constraint.

Now, just to make sure that your curves are smooth, etcetera. But there is no other constraint. Here, this has a shape constrained and also there is a trying data set concept introduced. So, it will be a nice algorithm. To know also very powerful algorithm used widely in its time. Even now, you can use it. There are now deep learning versions of this. If we will get time, we will cover that.

(Refer Slide Time: 52:54)



Here are some examples. So, this is for the knee in MR picture, right here. This is for the knee in MR picture. This is for face. So, now, you would, this is your initialization. This is the shape that you have estimated, point cloud shape. And you have now initialized it close to the curve, close to where the actual segmentation has to happen. And then after 14 iterations is kind of converges. You can see here.

Similarly, for the face. You have a model of the face given by the white curve. Put it on the face as close as possible and then you can convert after 18 iterations. So, there are quite a few freeware available for this, which you can download and play with, understand, what it takes to for this to converge.

So, it works in many situations but it would not work when you have like very tortuous shapes. This is not a good algorithm to use. And also, if you are trying to segment a lot of small objects in a big picture and then label them, multiple instances of smaller objects and not be a very powerful technique. Still, you still need to get the target shape because that problem is solved kind of by doing in a semi-automatic way where you can just have, somebody put the model shape very close to that supposed to be at.

And maybe that part of the problem is slightly solved but you still have to rely on some gradient information. So, this concludes our sessions on the conventional image processing techniques. Next week onwards, we are going to look mostly at machine learning based techniques but specifically deep learning based techniques and 3D CNNs and generative models. Thank you.