Intelligent Systems and Control

Prof. Laxmidhar Behera

Department of Electrical Engineering

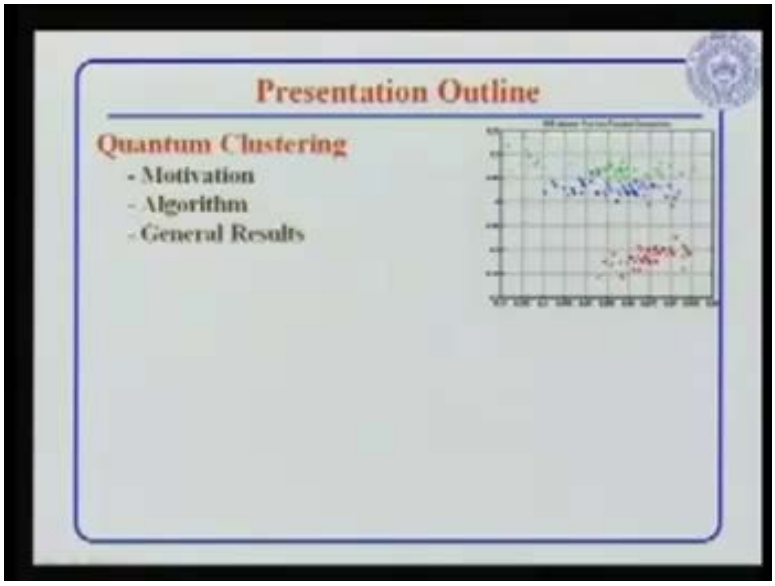
Indian Institute of Technology, Kanpur

Module – 3 Lecture - 8

Visual Motor Coordination Using Quantum Clustering

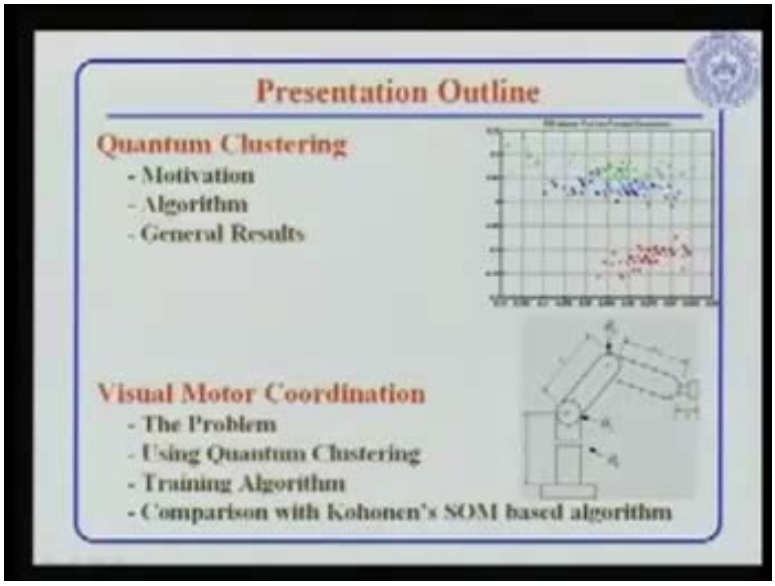
The topic for today is visual motor coordination using quantum clustering. Last class we discussed visual motor coordination using a different technique that is the Kohonen SOM (Self Organizing Map). Today we will present a different approach, little more introspective. It may be a little bit difficult for you to follow, but when you go through the class, you may be interested to do something in this area if you have a liking for it. This is the eighth lecture on this component neural control, which is module 3 of our course on intelligent control - Visual motor coordination using quantum clustering.

(Refer Slide Time: 01:16)



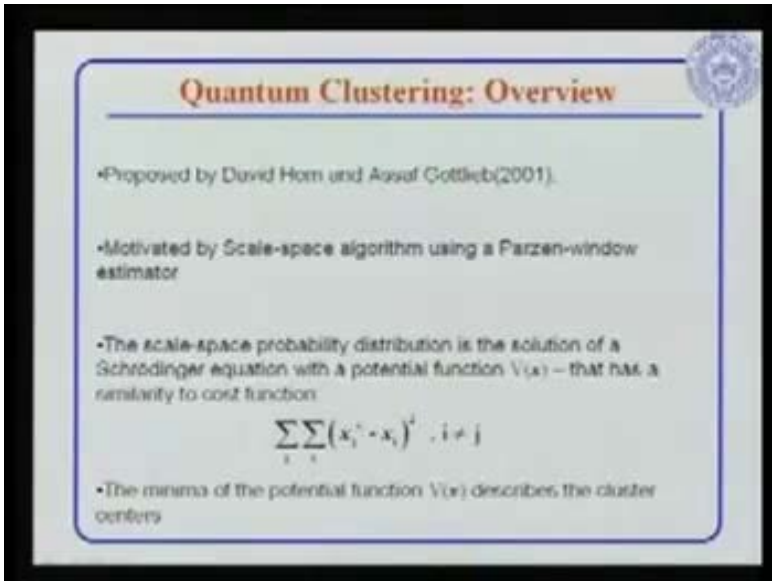
Presentation outline: We will be talking about Quantum Clustering. Earlier, we did the clustering using Kohonen self organizing map. We have also discussed what the meaning of clustering in general notion of clustering is. We will now be talking about quantum clustering, the motivation, the algorithm and then general results.

(Refer Slide Time: 01:39)



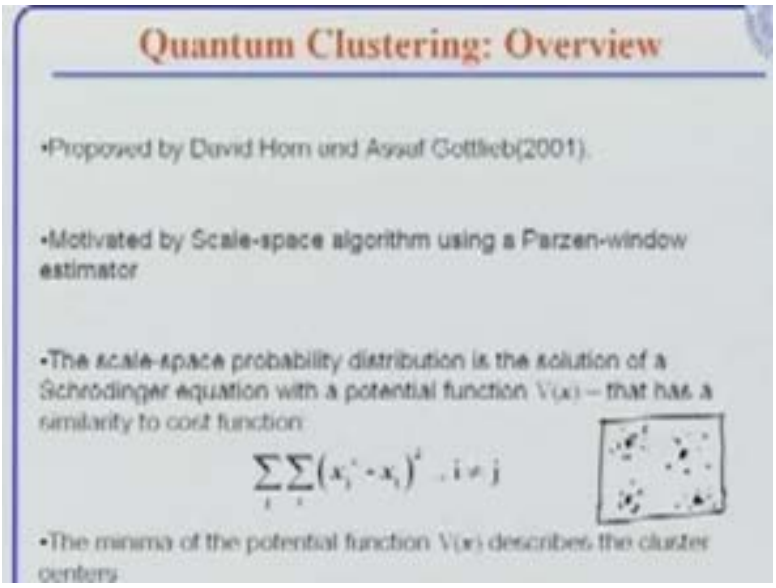
We will apply this quantum clustering to visual motor coordination which we have already discussed in the last class. The problem of visual motor coordination, using quantum clustering, training algorithm and comparison with Kohonen SOM based algorithm.

(Refer Slide Time: 02:01)



Quantum clustering and overview: those who want further information about quantum clustering I would suggest you read the paper published by David Horn and Assaf Gottlieb - The Method of Quantum Clustering, 2001. This particular quantum clustering algorithm is motivated by scale space algorithm using a Parzen window estimator. The scale space probability distribution is the solution of this Schrödinger wave equation with a potential function V(x) that has a similarity to the cost function.

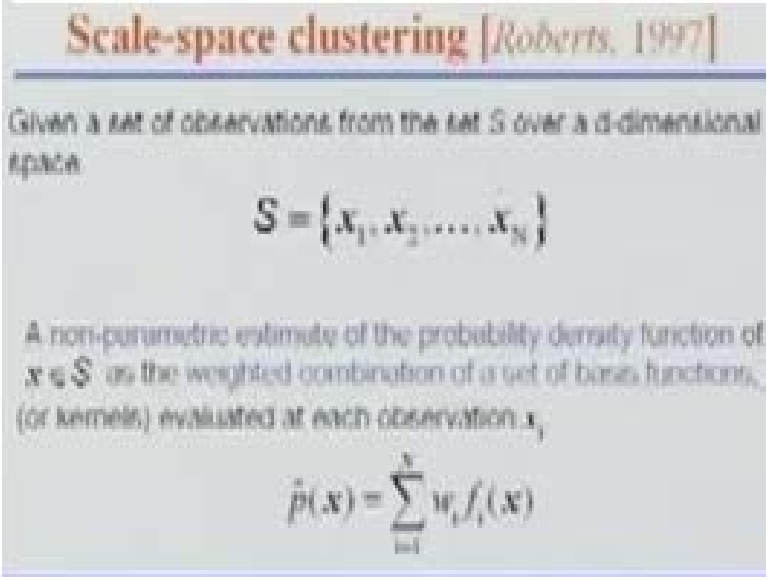
(Refer Slide Time: 04:28)



When we do any kind of clustering, the cluster center and the data point should always minimize the cost function. We always try to minimize the cost function. Let us see the 2-dimensional work space having some data points here and there, some sparsely there and lot of data point again here. Obviously what I would do is find one cluster point here; another cluster point here; another cluster point here and may be another cluster point here. The objective x_j c is the cluster point and this j would vary from 1 to 4 and x_i here represents all the data points. We would try to find out how far are these various data points from these cluster points and the objective of clustering is to minimize this distance (x_j c minus x_i). x_i represents all the data points in the data space and x_j c are the few clusters we have selected. In this case j is 1, 2, 3, 4 and so 4 clusters. The clusters must be selected in such a way that this particular cost function is minimized. The meaning of the potential function V(x) describes the cluster centers. In Kohonen network

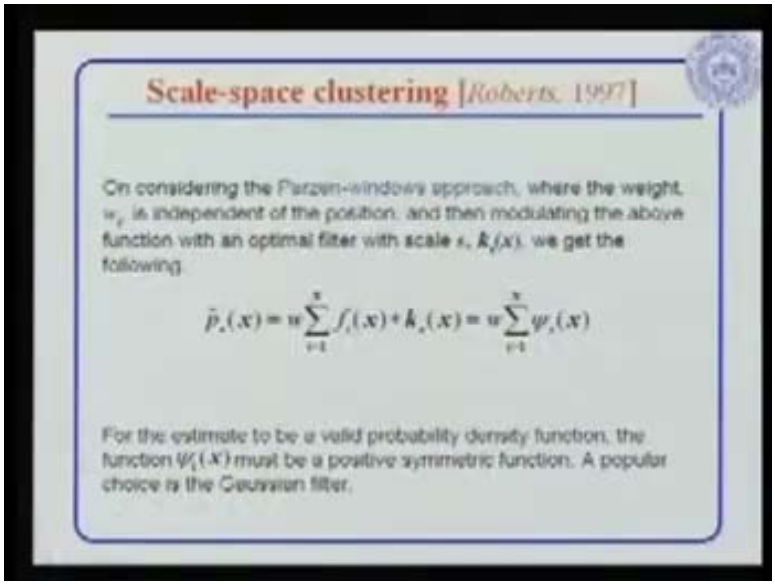
model that given a work space, and a data space with innumerable data, we try to represent those data with minimal representatives and the whole purpose of a clustering is to how to find these cluster points and that we have seen how to solve that problem using Kohonen self organizing maps. Scale space clustering, you can refer to the papers by Robert published in 1997.

(Refer Slide Time: 05:09)



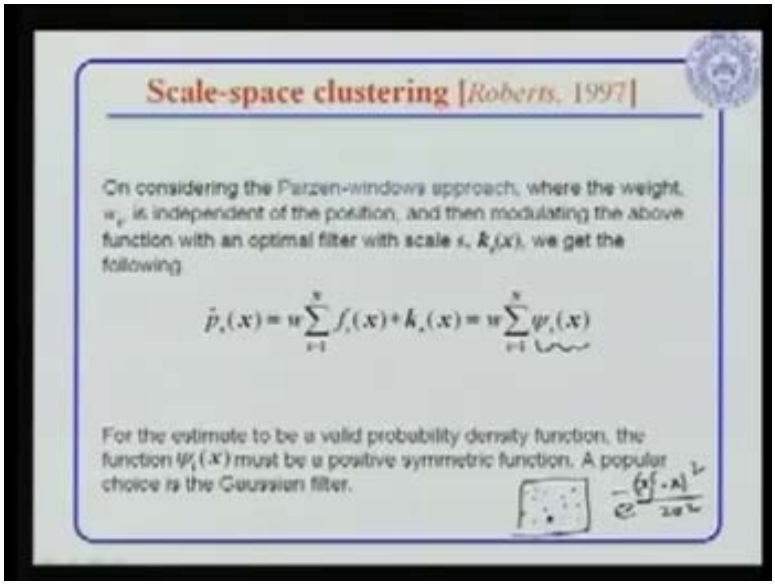
Given a set of observations from the set S over a d-dimensional space, S is defined as x_1, x_2, until x_N. A non-parametric estimate of the probability density function of x belonging to S as the weighted combination of a set of basis functions f_i or kernels evaluated at each observation x_i, the probability is given as $w_i f_i$ (x). When we look at the data space, what is the probability that this data is likely to be at a specific point in the data space?

(Refer Slide Time: 06:15)



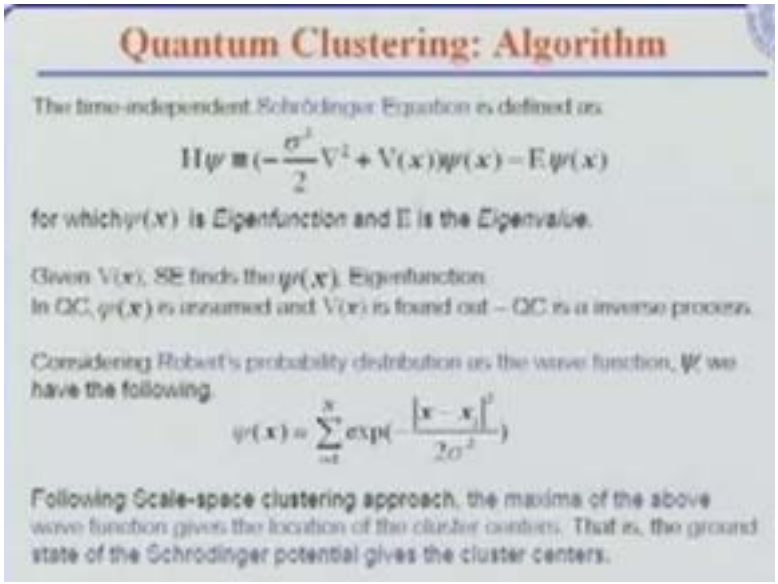
Considering the Parzen window's approach, where the weight w_i is independent of the position and then modulating the above function with an optimal filter with scale s, k_s (x), we get the following. This is the probability distribution function for scale space clustering. For the estimate to be a valid probability density function, the function $(psi)_i$ x must be a positive symmetric function and a popular choice of this is Gaussian filter.

(Refer Slide Time: 07:27)



What is normally done is that the given the cluster points – how do I find when datas are everywhere? Find out a cluster point that represents the data, we have a Gaussian function e to the power minus x_j c minus x whole square by 2 sigma square. This is the normal Gaussian function and what we would like to do is that we should maximize, because when this is minimum, this exponential function to the power is negative. Obviously, that has to be maximum and if this is minimum, ideally this should be 0, then e to the power minus 0 is 1 and that is the maximum value.

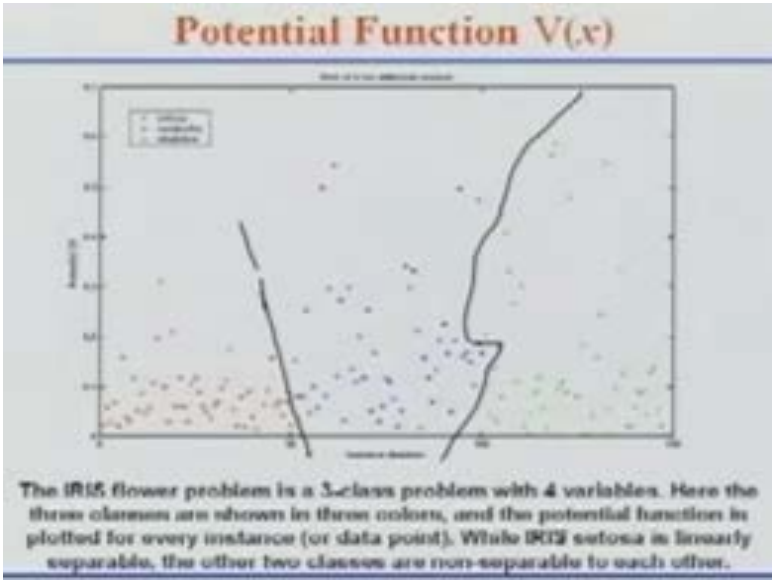
(Refer Slide Time: 08:06)



So in quantum clustering algorithm, a time independent Schrödinger equation is defined as H psi (H is Hamiltonian) minus sigma square upon 2, gradient square plus V (x) psi (x) is E psi (x) where psi (x) is the Eigen function and E is the Eigen value. Given V (x) the potential function, Schrödinger equation finds the psi (x) of the Eigen function. In quantum clustering, psi (x) is assumed and V (x) is found out. Normally in Schrödinger equation, we supply the potential function and see what is psi(x). But here given psi (x), V (x) is to be found and Q C is an inverse process. Considering Robert's probability distribution as a wave function psi, we have the following (Refer Slide Time: 09:12) and we assume our weight packets- to half Gaussian nature. So following scale space clustering approach, the maxima of the above wave function gives the location of the cluster centers. That is, the ground state of Schrödinger potential gives the cluster centers.

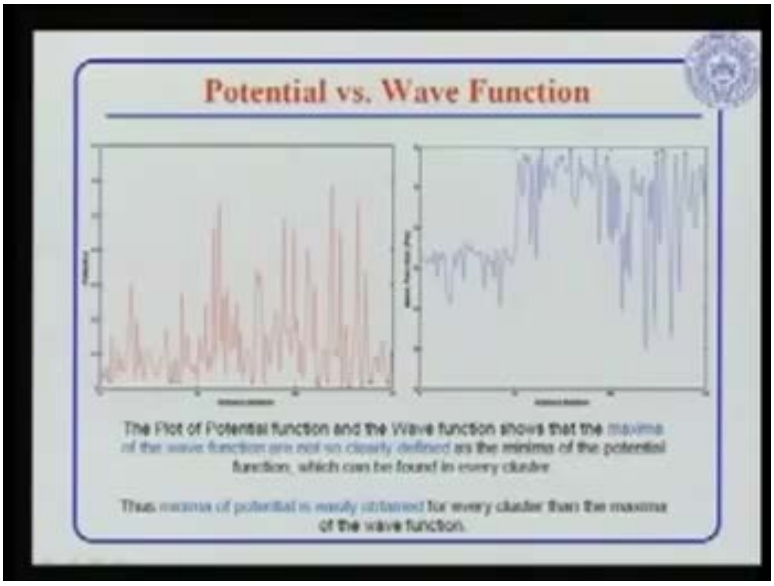
The maximum of this is the cluster center. In the same way the minimum of this is also the cluster centre.

(Refer Slide Time: 10:41)



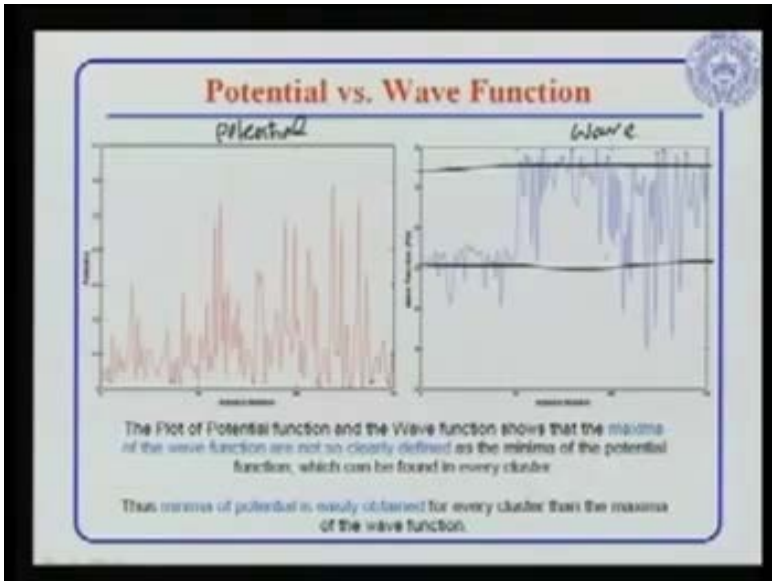
A potential function can be discussed as follows. IRIS flower problem is a 3-class problem with four variables. The three classes are shown in three colors: red, blue and green. The potential function is plotted for every instance or data point; where IRIS setosa is linearly separable and the other two classes are non-separable to each other. In this case, the setosa is the red one and I can place a line by which it is separable from the other two classes. But I cannot actually put a line because it is a potential I have plotted for each data point and the potential function V (x). With some raw data, I map them in such a way that each class of data can be separated and so that they occupy different zones. The raw data are all mixed and by doing some kind of operation on them, can I separate those classes? This is the point of clustering.

(Refer Slide Time: 11:36)



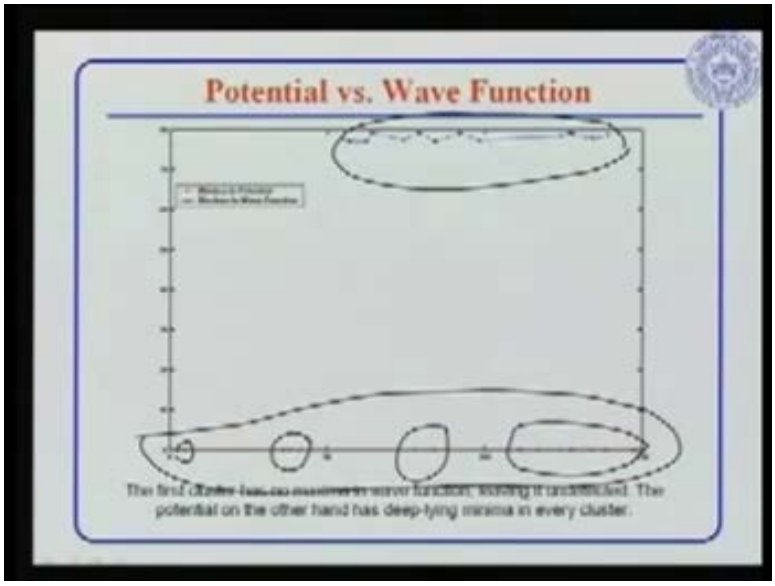
Potential versus wave function: As we said, normal approach is to find out the Gaussian mixture model of the data points. While trying to find out the cluster points, we maximize the Gaussian mixture model while deriving the cluster centers. We showed that it is the same as finding the minima of the potential function. The plot of potential function is a wave function source and that maxima of the wave function are not so clearly defined. The minima of the potential function which can be found in every cluster is actually the minimum and these are the maximum points of the same IRIS data and you can easily see that the maximums here over various data points. If I put a threshold here, these maximas will appear. In this case, if I put a threshold here, I am getting all these cluster points; because they represent the minimum values. Similarly if I go on, you have to compromise a lot to find some cluster points that means I have to put a threshold in this level. So while putting threshold in this level, you are compromising that means all data remain under the cluster point. So what you are doing is that if I put a more reasonable threshold, using this wave equation and a potential, the cluster points for every data point we are computing, what is the potential function using the formula?

(Refer Slide Time: 13:44]



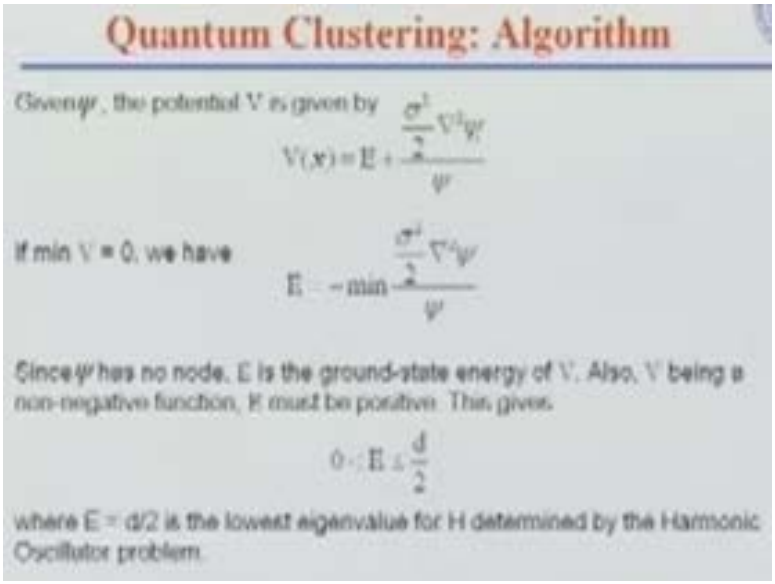
We are finding out the Gaussian mixture model function, we compute that for each data point we plot that and there is a fluctuation. How do I find a cluster point? I find that as a cluster point for which the potential function is minimum here and in this case the wave function or the Gaussian function is maximum.

(Refer Slide Time: 14:57)



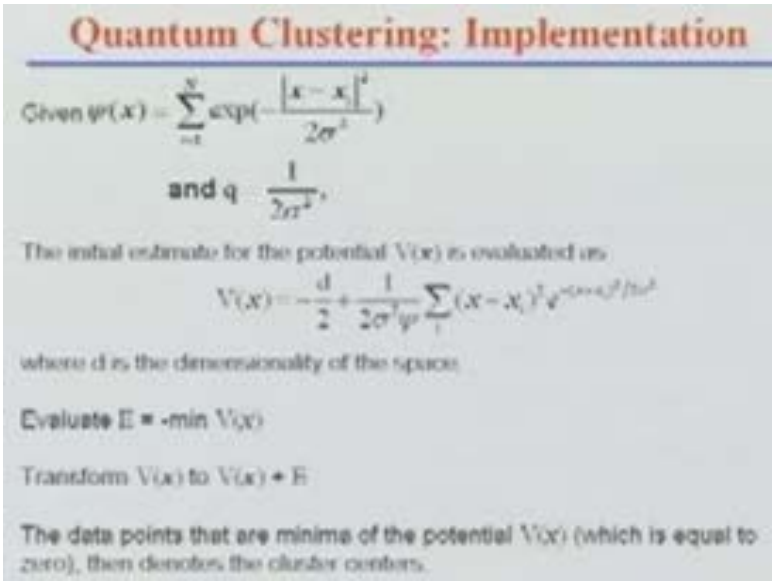
I put a threshold for this wave function here and threshold for the potential function over the entire range of the data, I have cluster centers. Here there is 1 cluster, 3 clusters, 2 clusters and multiple clusters, whereas if I look at the wave function side, I have only clusters in this zone. I have no clusters in this zone and this entire zone is unrepresented.

(Refer Slide Time: 15:05)



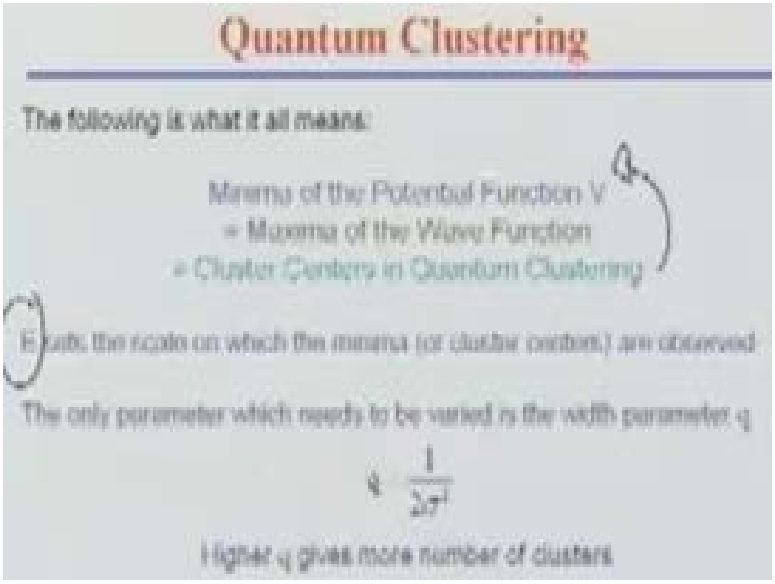
In Quantum clustering, given psi the potential function V is given by V x is E, which is the Eigen value and sigma square upon 2 into gradient square into psi upon psi. If V is 0, then the minimum is 0. If we have E equal to minus min sigma square upon 2 into grade square psi upon psi. Since psi has no node, E is the ground state energy of V, V being a non-negative function and E must be positive. This gives E between 0 and d by 2; where E is equal to d by 2 which is the lowest Eigen value of H determined by Harmonic oscillator problem.

(Refer Slide Time: 15:56)



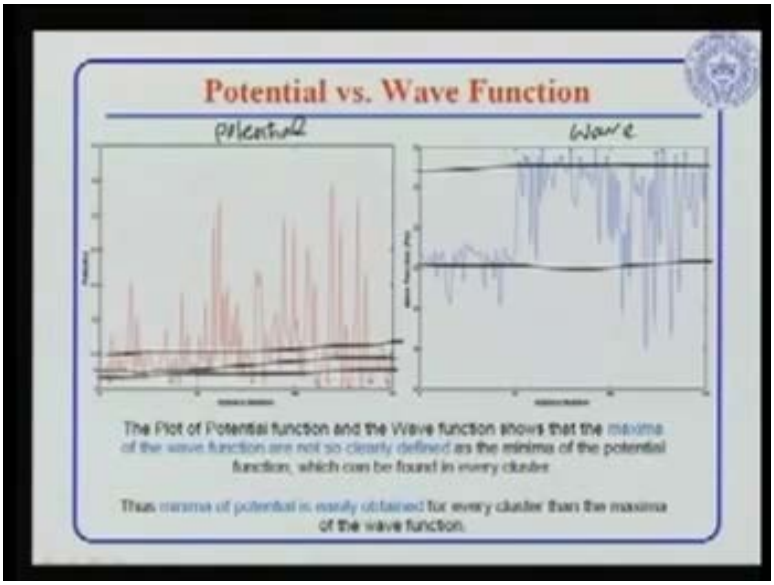
Quantum clustering implementation: This is a wave function, Gaussian function; where q is 1 upon 2 sigma square, because this gives the width. The initial estimate of the potential function V (x) is evaluated as minus d by 2 plus 1 upon 2 sigma square psi sigma i x minus x_i whole square and this function e to the power minus x minus x_i whole square by 2 sigma square where d is the dimensionality of the space. Evaluate E equal to negative of minimum of V(x). Transform V(x) to V(x) plus E. The data points that are minima of the potential function V(x) which is equal to 0 denotes the cluster centers. So we add this E to find out the cluster centers.

(Refer Slide Time: 17:24)



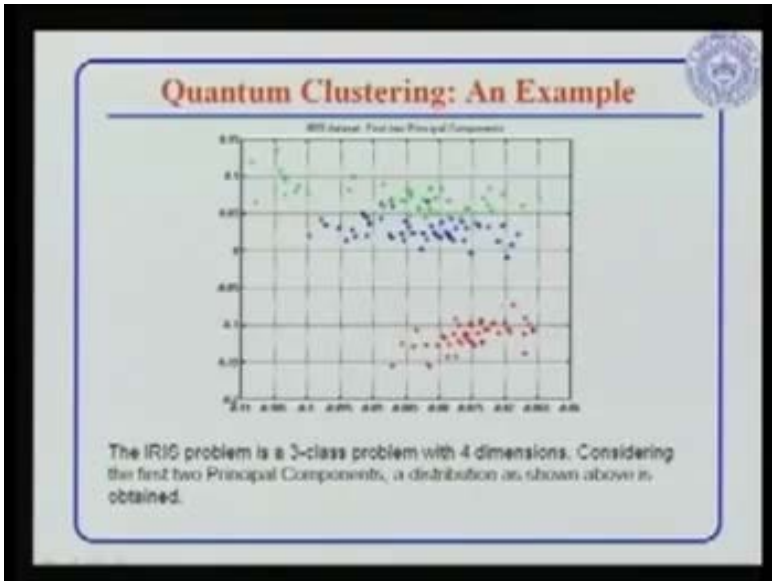
In essence, the minima of the potential function V is the same as the maxima of the wave function and based on that, cluster centers are decided in quantum clustering using the minima of the potential function. E sets the scale on which the minima or the cluster centers are observed; where to put the threshold is selected by E. From earlier concept it determines where to put the thresholds, here or here.

(Refer Slide Time: 17:29)



If I am putting the threshold at the low value, I have fewer clusters and the moment I increase more and more, I have the number of clusters more. This is adaptive. Depending on E, our cluster centers will change and E will depend on 1 upon 2 sigma square. The only parameter which needs to vary is the width of the parameter and q is 1 upon 2 sigma square. Higher q value gives more number of clusters. So if we want more number of clusters, we increase the value of q.

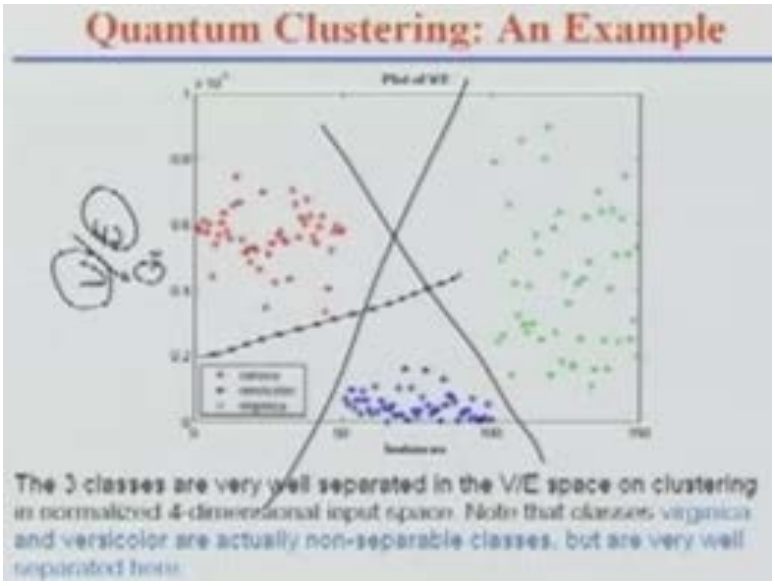
(Refer Slide Time: 18:24)



Here is an example: If you look at here the IRIS data set, first two principle components you plot and you have these 3 different classes.

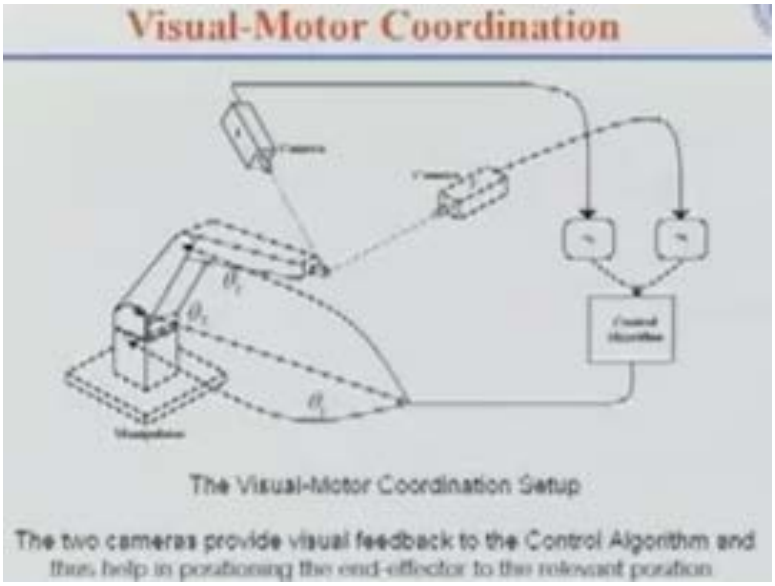
We find one class is completely separable from the other class, but these two class are all mixed. So we have to create a method by which we can linearly separate the green from the blue. The IRIS problem is a 3-class problem with 4 dimensions considering the first two principle components, a distribution as shown above is obtained.

(Refer Slide Time: 19:12)



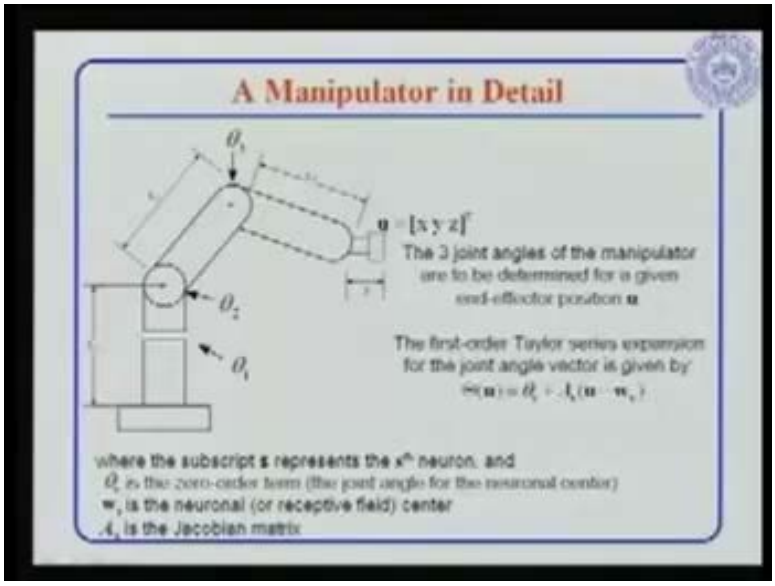
By mapping these data points, by transforming them through their potential function and the threshold E; V is the potential function and this is V by E; where E decides the threshold, we have a a line here that separates this class from these two classes. Similarly we put another line that separates the green from blue and red thereby the three classes are now distinctly separable. The three classes are very well separated in V E space. On clustering in normalized 4-dimensional input space, note that classes virginica and versicolor are totally non-separable classes but are very well separated here like the red and green. If we look at the previous color red and blue, they are inseparable and now the red and blue are completely separable. We can put a line by which they can be separated.With this idea on quantum clustering, we apply it on visual motor coordination. We have introduced some new concepts also to this visual motor coordination.

(Refer Slide Time: 20:41)



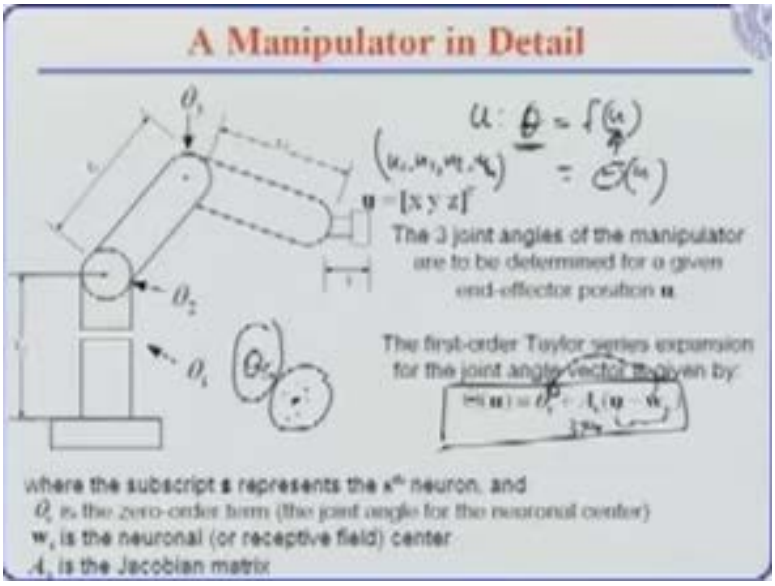
As discussed earlier, the visual motor coordination has a robot manipulator and this robot manipulator whose end-effector is being observed by two cameras, camera one and camera two. If you have a target point and if this end-effector wants to reach that point, the target point is also in the camera focus and also the end-effector. The objective is how to reach this target point through learning. Imagine I am holding a pen here and the other hand wants to reach that point. This is a visually guided motion. In this visual guided motion. I am not aware of the arm dynamics and I do not take into account any kind of dynamics from my hand; rather the mechanism is such that it is like a learning mechanism that guides my hand to a target point. If you see a child, he learns as he grows by various trial and error methods. Similarly, in this situation by hand eye coordination, we do many things dexterously.

(Refer Slide Time: 22:25)



Here is the manipulator in detail dynamics: We have three joints; one is the base ,this is the joint for the first link and then another here, the joint for the second link. So the base, first link and second link comprise a 3-link manipulator. The three joint angles of the manipulator are to be determined for a given end-effector position u; where the coordination principle is, camera finds the end-effector position. When you take a photograph of a 3 D point, camera which has only 2 D point, as u_1 u_2 and camera 1 and camera 2 is u_3 and u_4. You have this 4-dimensional input vector for a given target, a given end-effector point. Given a target, the objective is that the control algorithm should find $theta_1$, the base angle and $theta_2$ which is the joint angle one, two, $theta_2$ and $theta_3$ such that by orienting $theta_1$ $theta_2$ and $theta_3$ properly, this end-effector will exactly reach this target point.

(Refer Slide Time: 24:08)

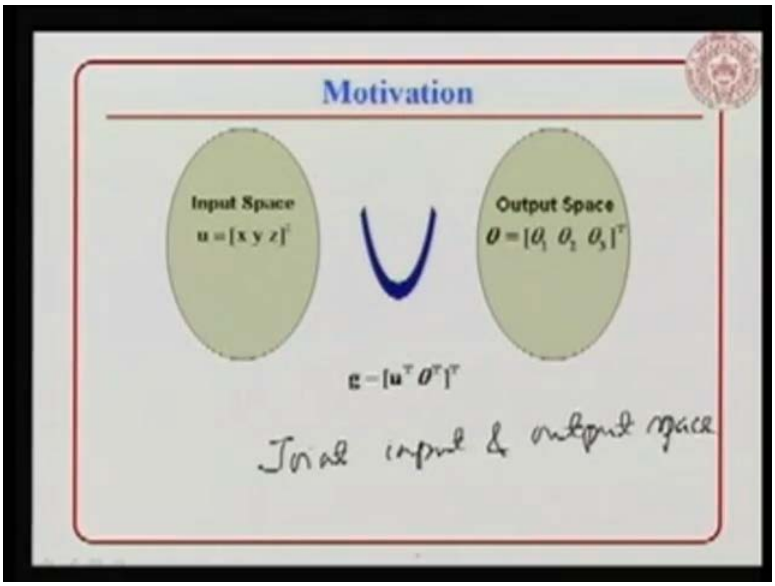


What we are trying to map u, theta is f of u. We would find out theta given u through learning and one can use easily the inverse kinematics of this manipulator and can find out that.

This principle is different; we need not go to the manipulator inverse kinematics and simply learn to find out what is theta equal to f(u). The objective is that this is just a curiosity that how learning can be so powerful and we can ignore the inverse kinematics of the actual model and still we can reach any point in the robot work space. What we do is that this function of u can be linearized around a specific formula which is in a local zone. If I know theta_s for response to do, this point in the Cartesian space point corresponds to theta_s. So then any point near to this Cartesian point will be very close to theta_s. Thus we can find this expression and the relationship by simply linearizing this non-linear function around theta_s. This is linearized around theta_s plus a Jacobian matrix which is obviously 3 by 4 matrix and u is the input vector and actually input is not x y z, this is u_1, u_2, u_3, u_4 and u represents Cartesian space, but this is actually the input coming from the camera. This is 3 by 4 and u is the actual input that is the actual data point around the thick dot and w_s is the neuronal center that is when I localize or discretize this entire work space as we did in the Kohonen SOM algorithm. When I create a discrete cell, each cell is assumed to be represented by a neuron, so that neuron is associated with

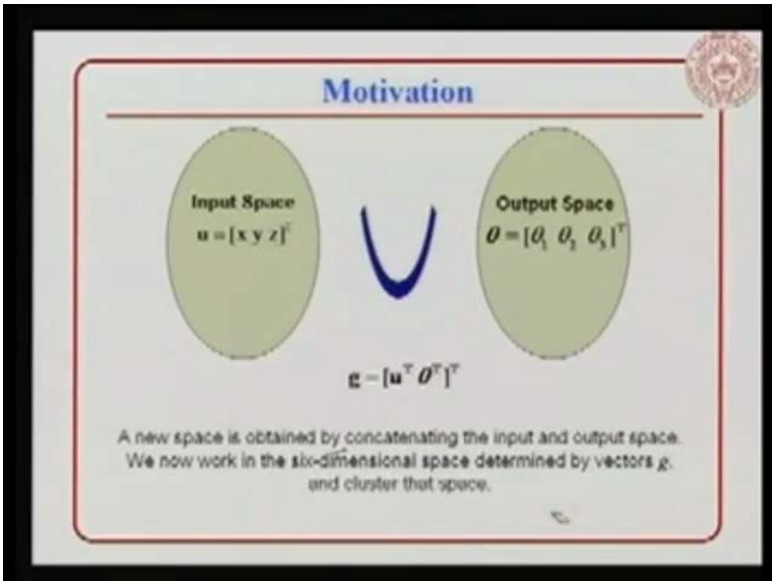
a specific w_s and the meaning of that is, this w_s actually corresponds to this $theta_s$. If given any other point u very close to w_s, how do I represent the exact theta? This is my actual theta and so my actual theta is a local zone in which I have a special point in the 3D coordinate and corresponding to this coordinate, I have $theta_s$ as the actual joint position $theta_1$, $theta_2$, $theta_3$. They take this end-effector to this point which is the representative in the camera coordinate which is w_s. We can write at a given point which is very close to this point in the Cartesian space, the joint space theta can be written as $theta_s$ plus A_s u minus w_s. So, u represents a new data point around the old point which is w_s. This is the meaning of linearization around a small work space.

(Refer Slide Time: 29:32)



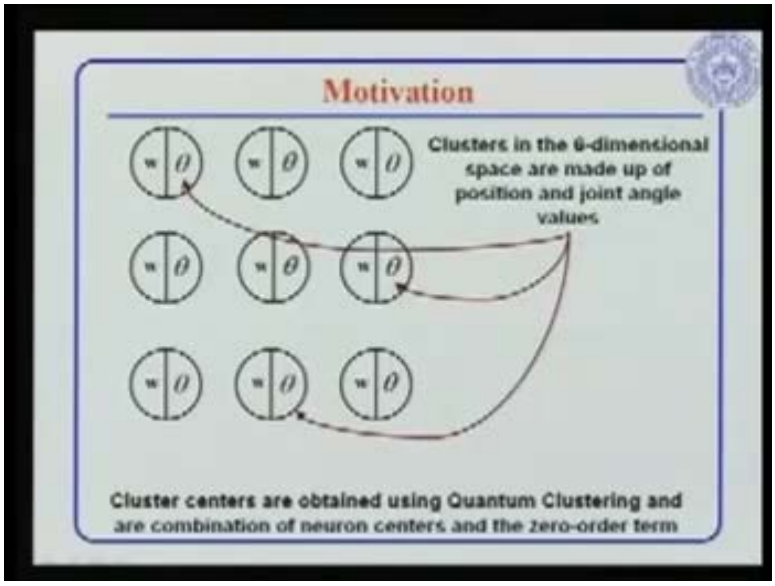
What we are now doing is that we have an input space and an output space; input space is 4-dimensional (29:37) and we can make the dimension 3. In a two camera coordinate system, we get 4 points and the Cartesian dimension is 3, but we get 4 points, Output space is $theta_1$, $theta_2$, and $theta_3$. If we create a joint called joint input and output space, it is u transpose, theta transpose, transpose, a 6-dimensional vector.

(Refer Slide Time: 30:24)



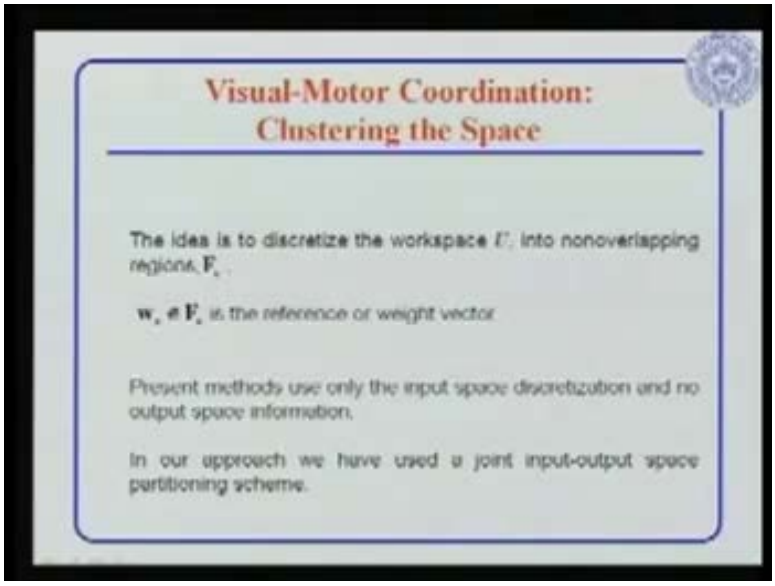
A new space is obtained by concentrating the input and output space. On working in the 6-dimensional space determined by vector g and cluster space, we created a cluster only in the input space and now we are creating cluster both in input and output space. Now we created a cluster and have a work space and in the work space, the input-output space is divided into small cells.

(Refer Slide Time: 31:02)



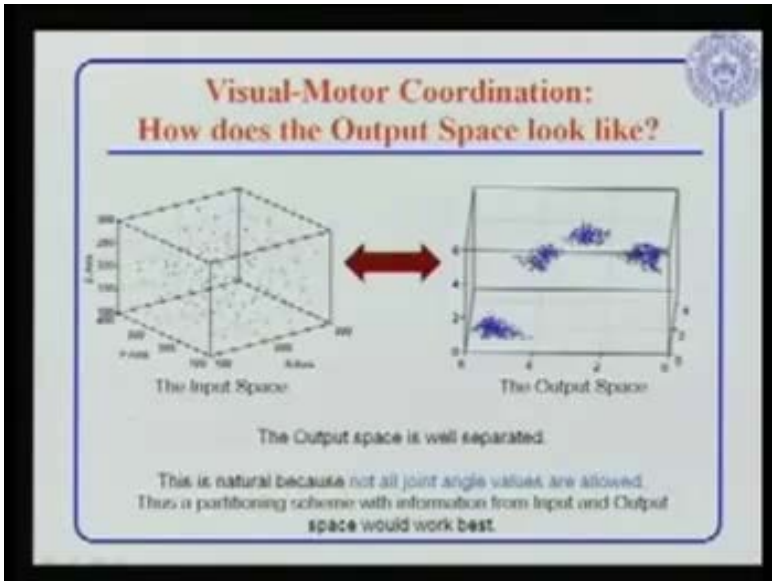
The clusters in the 6-dimensional space are made up of position and joint angle values. Corresponding to each special position w, we have a specific w theta and these clusters are obtained using quantum clustering technique rather than the Kohonen self organizing map. The cluster centers are obtained using quantum clustering and are combination of neuron centers and the 0 order term.

(Refer Slide Time: 31:36)



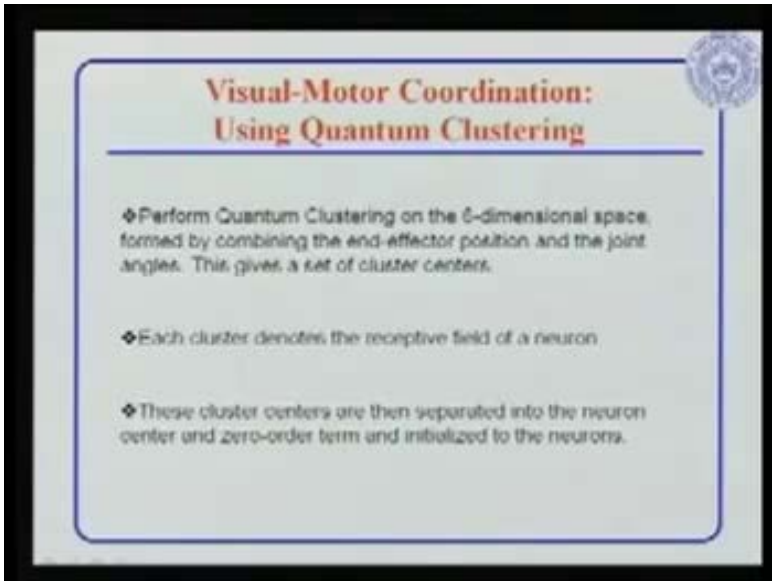
The idea is to discretize the workspace u into non-overlapping regions F_s and each one of these are all small discrete cells belonging to F_s. w_s belonging to F_s, which is the reference or weight vector for each discrete cell. This is your reference vector associated with the each discrete cell. Present methods use only the input space discretization and no output space information and the Kohonen self organizing map can also be incorporated and this is independent of the other. You do quantum clustering or Kohonen SOM; this is not a defect of Kohonen SOM. It is just that we are utilizing that feature. In our approach, we have used the joint input-output space partitioning scheme.

(Refer Slide Time: 32:40)



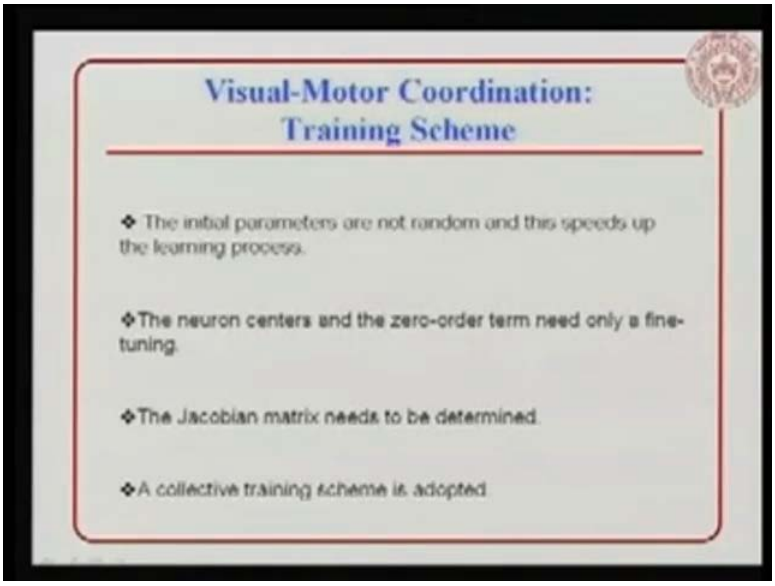
This is 3D work space in which the robot is working in this and any random point in the work space, the robot manipulator can reach and corresponding to each point in joint space, we have a specific $theta_1$, $theta_2$, $theta_3$, but if we look at the output space, they are not uniformly distributed. The output space is well separated; it means it is natural because not all joint angle values are allowed. When I manipulate my hand, I cannot probably create certain angles like a robot manipulator. Thus, a partitioning scheme with information from input and output space would work best.

(Refer Slide Time: 33:49)



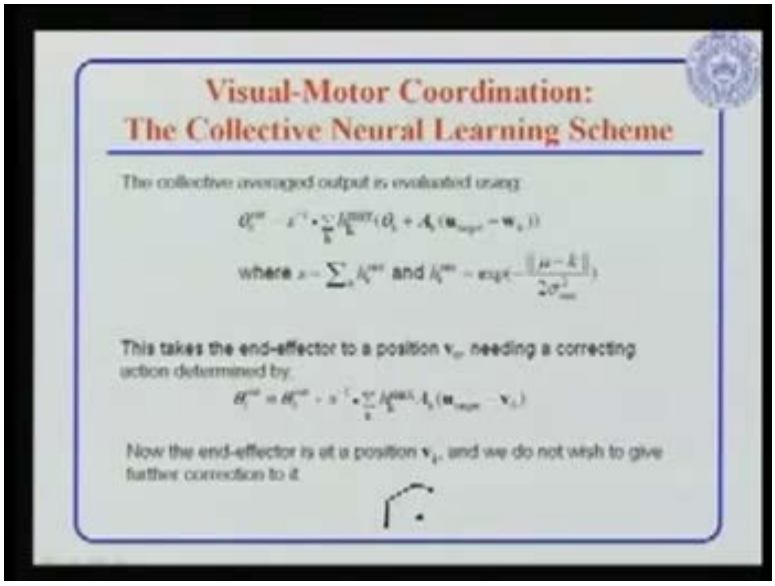
The first part is performing quantum clustering on the 6-dimensional space formed by combining the end-effector position and the joint angles. This gives a set of cluster centers, each cluster denotes the receptive field of a neuron and these cluster centers are then separated into the neuron centers and 0 order term end initialized to the neuron.

(Refer Slide Time: 34:16)



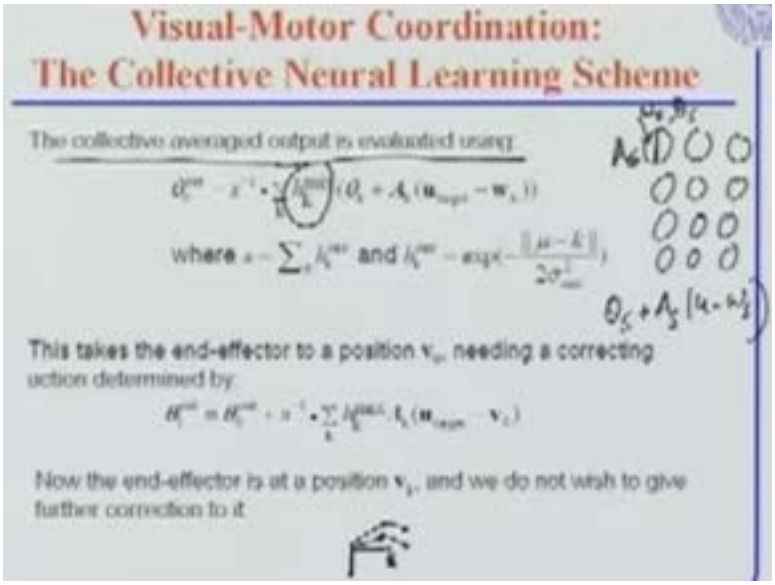
The initial parameters are not random and this speeds up the learning space process .The neuron centers and 0 order term need only a fine tuning and the Jacobian matrix needs to be determined. A collective training scheme is adopted which means we create a cluster. Through clustering, we find out w_s as well as $theta_s$. Now $theta_s$ and w_s are given and with new input u, how do I learn this s? How do I also refine the relation between $theta_s$ and w_s? This is key question being asked.

(Refer Slide Time: 35:07)



The objective of this training scheme can be explained. For example, this is my robot manipulator; this is one link; this is another link and this is another link. For example, we have to come to this target point and what happens is that this algorithm gives some random variables, $theta_1$, $theta_2$, $theta_3$ -in the beginning, when it was not learnt to various links and then the end-effector goes to some point. The learning principle is, how I can with some minimal steps bring this end-effector?

(Refer Slide Time: 38:08)

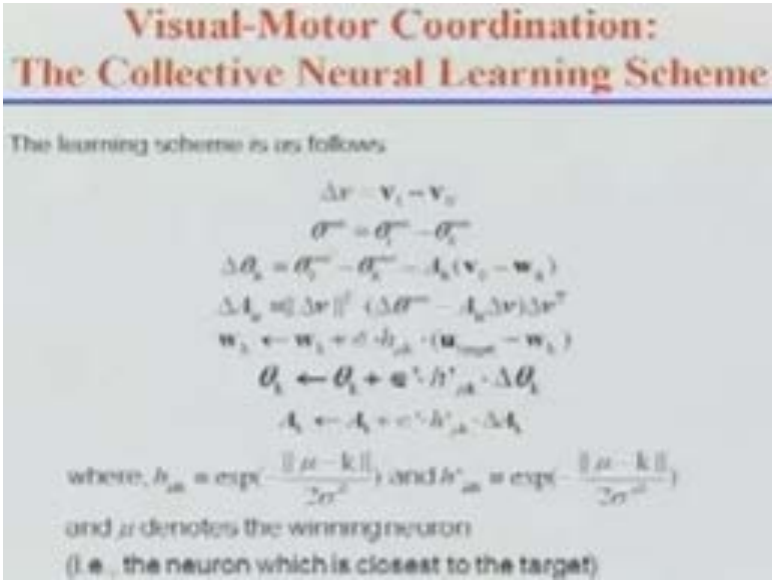


I can bring from this point to this point through training and what is happening here is that you see that I have many neurons in the workspace and these neurons are actually a 3D lattice. In the case of Kohonen network, it is a 3D lattice, but in the case of quantum clustering, we do not create a lattice kind of thing. But each neuron represents a specific discrete zone of the input, output space and while clustering you have already the coordinates associated with this neuron which is w_s and $theta_s$. Once clustering is done and for each neuron A_s is defined, then obviously given u, each neuron output would be $theta_s$ plus A_s u minus w_s by each neuron. But instead of allying only one neuron to take a decision, we allow a group of neurons to take decision which is collective averaged output.

The collective averaged output is that k represents each neuron and this is within a specific neighborhood and this H is the distance that is given input as specific neuron is the winner according to cluster. The winning neuron will have a maximum role to contribute to the decision making process and others will have less and less say. Each neuron is associated with three parameters and the output of each neuron is this. If this is the winning neuron, then the maximum contribution comes from the winning neuron and lesser and lesser contribution. The neurons are away from winning neuron and their ability to contribute to the decision making process dies down that is, their participation

is to the degree how close to the winner. So the collected averaged output in a given input u_{target} and this is the distance M u is the winner and k is the specific neuron what is the distance between them and so we also put this neurons in a lattice and find out the end-effector to a position v_0, needing a correcting action determined by this $theta_1$ out is $theta_0$ out which is the initial plus this is the correcting that is instead of w_k we now put it here v_0. Now the end-effector is at a position V_1 and we do not wish to give further correction to it; because, if this takes it to V_1, then we do not need it.

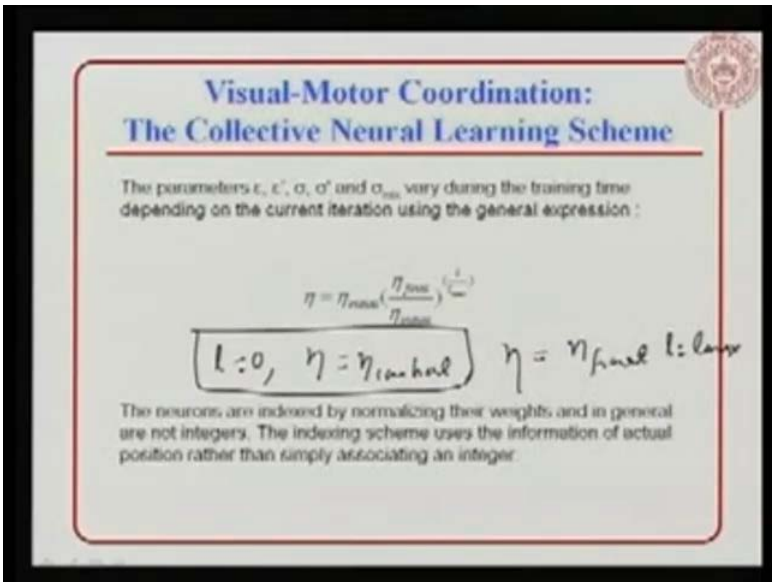
(Refer Slide Time: 40:28)



The learning scheme is actually the gradient distance. Delta V is V_1 minus V_{naught}. I discussed all these things in the last class; theta out is $theta_1$ out minus $theta_0$ out. $Theta_0$ out minus $theta_k$ out; because, this is a kth iteration minus A_k V_0 minus w_k. This is the update law for the Jacobian matrix which is a norm delta V square this delta theta out. You can actually derive this algorithm from using gradient descent. Finally, each weight of the neuron is updated as w_k using clustering algorithm -just like we did in the Kohonen clustering algorithm; where w_k is w_k plus epsilon $h_{\mu k}$ u_{target} w_k, $theta_k$ is $theta_k$ plus epsilon dash $h_{\mu k}$ dash delta $theta_k$. My new weight associated with a neuron k is the old weight associated with neuron k plus epsilon is the learning rate. This is the distance factor; how far is kth the neuron from the winning neuron and u_{target} minus w_k? This is a normal learning that is done in clustering. Similarly for $theta_k$ the old $theta_k$ plus the

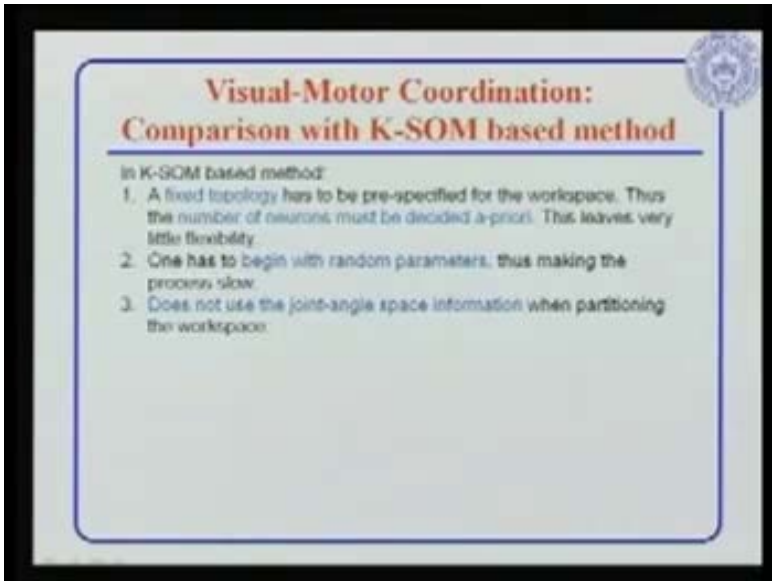
learning rate and another distance function and delta theta$_k$ .Similarly for the Jacobian, we train with old Jacobian. In the beginning, the beauty of this entire learning process is that we start from absolutely no idea. Our neural network possesses no knowledge of the workspace and everything is initially, randomly initialized without taking any consideration, the workspace size -nothing is taken into consideration. The moment you give the data, the neural network is associated or excited by the data the learning starts and then beautifully the mapping of the topology of actual work space gets mapped to the neuronal structure and nice mapping results. This is a distance measure; Mu denotes the winning neuron, the neuron which is closest- to the target.

(Refer Slide Time: 43:45)



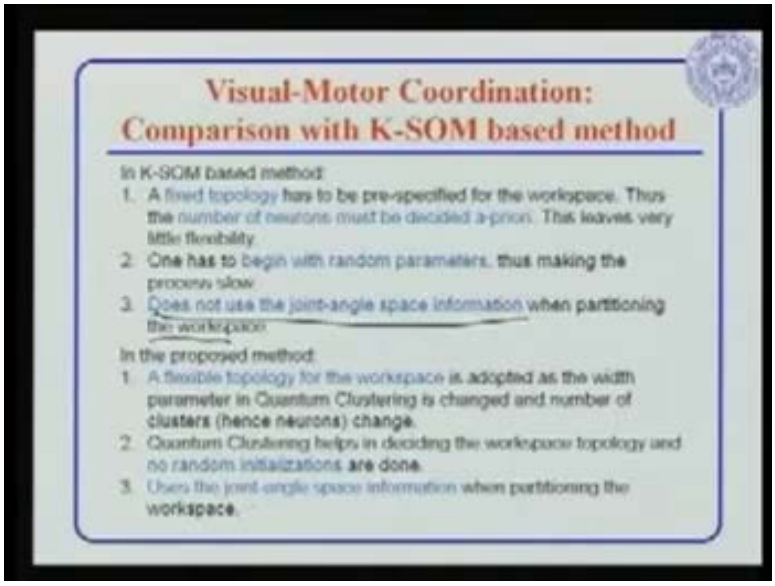
The parameters epsilon, epsilon dash, sigma, sigma dash and may vary during the training time depending on the current iteration using general expression Eta is Eta$_{initial}$ Eta$_{final}$ upon Eta$_{initial}$ l by l$_{max}$, this is the iteration . When I have not started the iteration, then this is 0. Eta is Eta initial; because this to the power 0 is 1. When this l is l$_{max}$, this is 1; so this becomes Eta$_{final}$ by Eta$_{initial}$ cancels out, Eta is Eta$_{final}$. Eta in the beginning is Eta$_{initial}$ and Eta is Eta$_{final}$ at l equal to l$_{max}$ and this is 1 equal to 0. The neurons are indexed by normalizing their weights and in general are not integers. Indexing scheme uses the information of actual position rather than simply associating an integer.

(Refer Slide Time: 44:59)



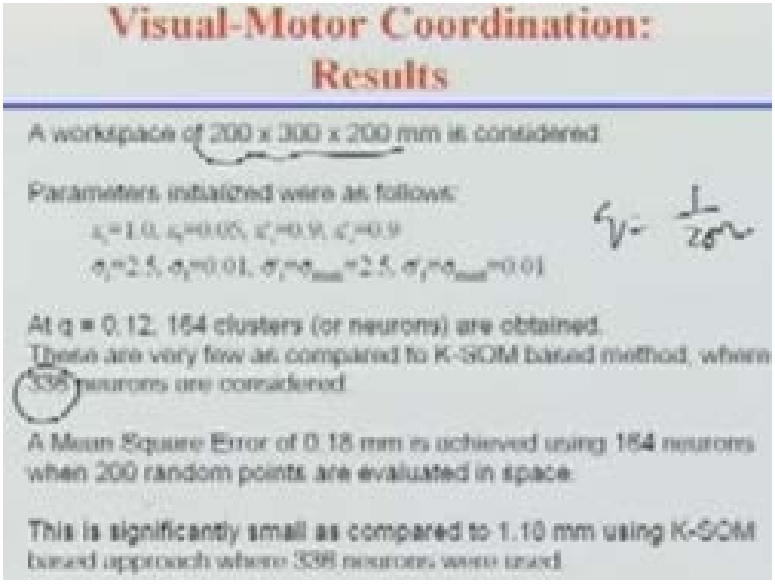
Now we would compare our results today to the last class. Last class we talked about Kohonen self-organizing-map-based methodology where we had a fixed topology in the beginning. We fixed the neurons and fixed topology has to be pre-specified for the workspace. Thus the number of neurons must be decided priori and this leaves very little flexibility. One has to begin with random parameters, thus making the process slow and does not use the joint angle space information when partitioning the workspace. But this is not confined to Kohonen sum; it is an additional theme because we can also do this for Kohonen.

(Refer Slide Time: 45:52)



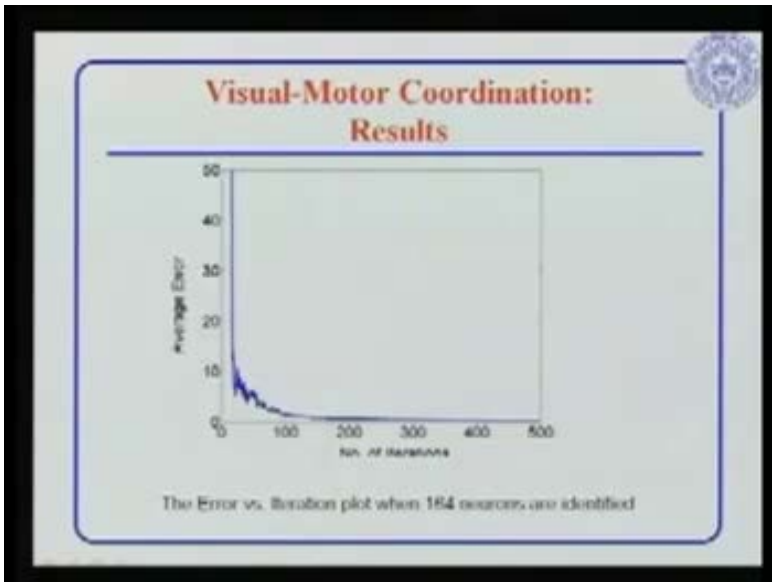
What we learned is that a flexible topology of the workspace is adopted as the width parameter in quantum clustering is changed and a number of clusters changed, quantum clustering helps in designing the workspace topology and no random initializations are done using joint angle space information when partitioning the workspace. Again this is not exclusive to quantum clustering and this can also be in the quantum clustering.

(Refer Slide Title: 46:23)



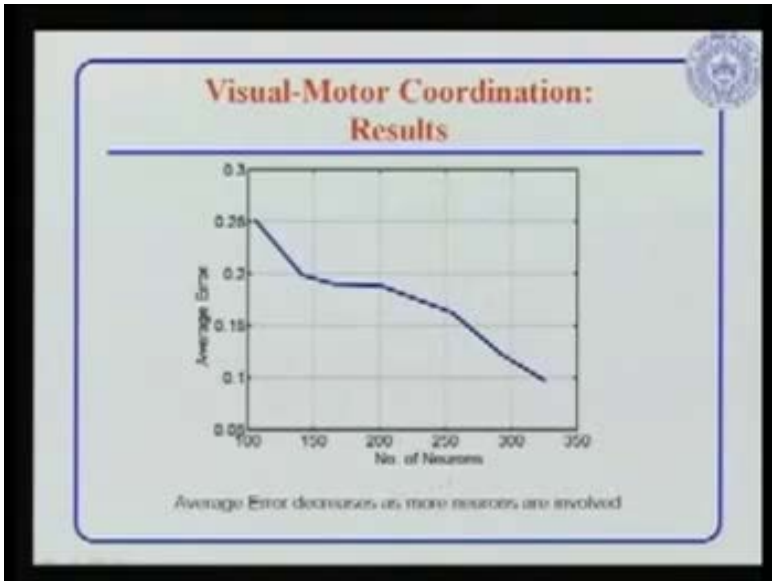
Our workspace size was 20 centimeters into 30 centimeters into 20 centimeters. The initial parameters were taken like this: (Refer Slide Title: 46:35). In the beginning, q is1 upon 2 sigma square; this was fixed as 0 point 12 and then 164 for clusters or neurons are obtained .These are very few as compared to K SOM based method where 336 neurons are considered. In a Kohonen cluster 7 into 12 into 4 many neurons were considered. Here, it is only 164 clusters, a mean square error of 0 point 18 mm is achieved using 164 neurons when 200 random points are evaluated in space .This is significantly small as compared to 1 point 18 mm using K SOM based approach where 336 neurons were used. This is the result we achieved in the last class using Kohonen quantum clustering algorithm.

(Refer Slide Time: 47:38)



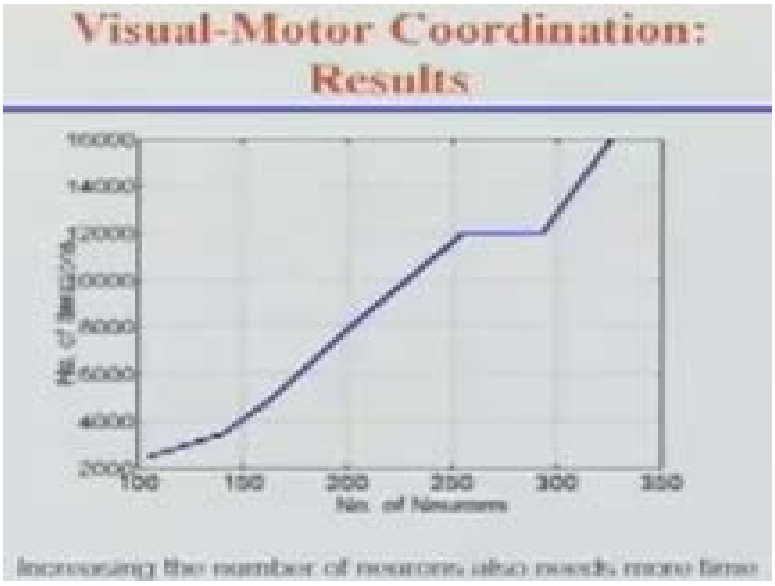
Here it is the average error and number of iterations, so what you are seeing is that the errors are almost in 200 samples iterations; whereas normally in Kohonen clustering it takes 5000 iterations.

(Refer Slide Title: 48:07)



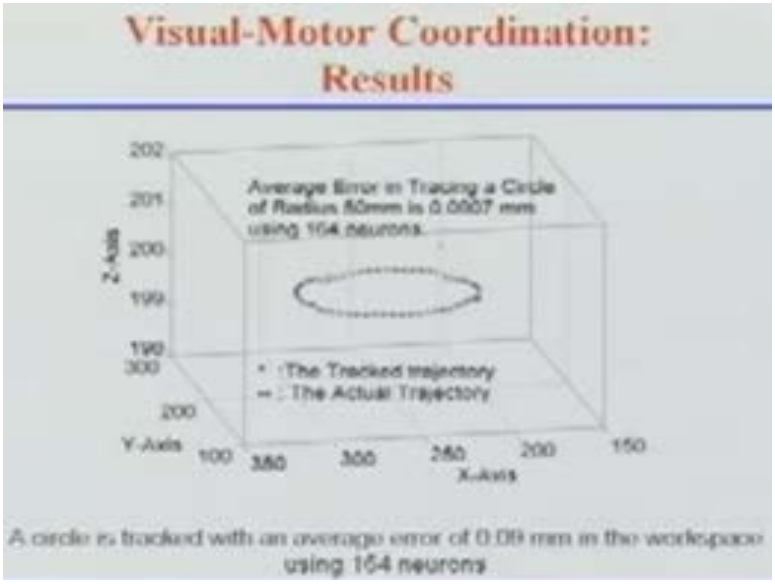
In this quantum clustering, I have put this as the average error and the number of neurons. This is an adaptive and flexible topology in the case of quantum clustering and the number of neurons are not fixed in the beginning; that means how many discrete cells the original workspace should be consisting of, we decide in Kohonen's self organizing map But in quantum clustering, these can be varied by varying the q; which is 1 upon 2 sigma square. If I increase the number of neurons naturally the average error decreases and this is the profile. You increase the number of neurons, (average error) decreases.

(Refer Slide Time: 48:53)



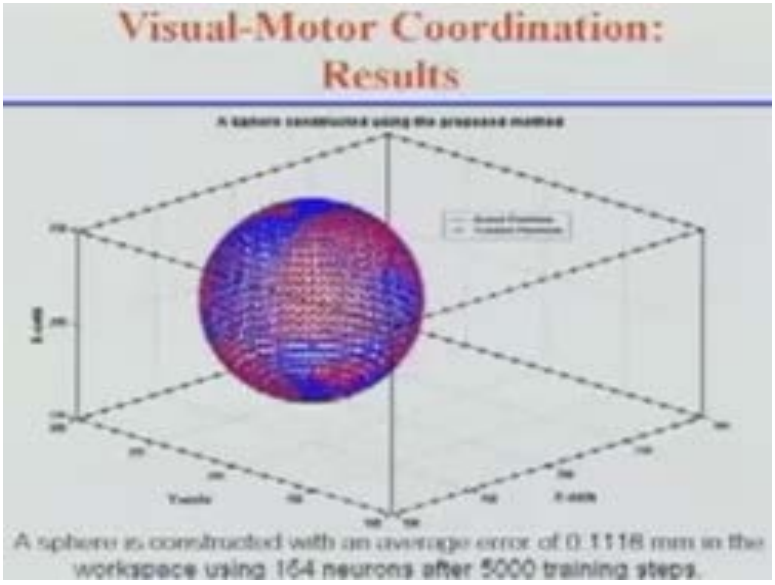
This is another interesting piece of result where we find the number of iterations we require for training and the number of neurons. If you increase the number of neurons more and more, the number of iterations required is in large numbers and the price for making this is high. In such a situation, if the number of neurons is small, less number of iterations is required. Increasing the number of neurons also needs more time.

(Refer Slide Time: 49:35)



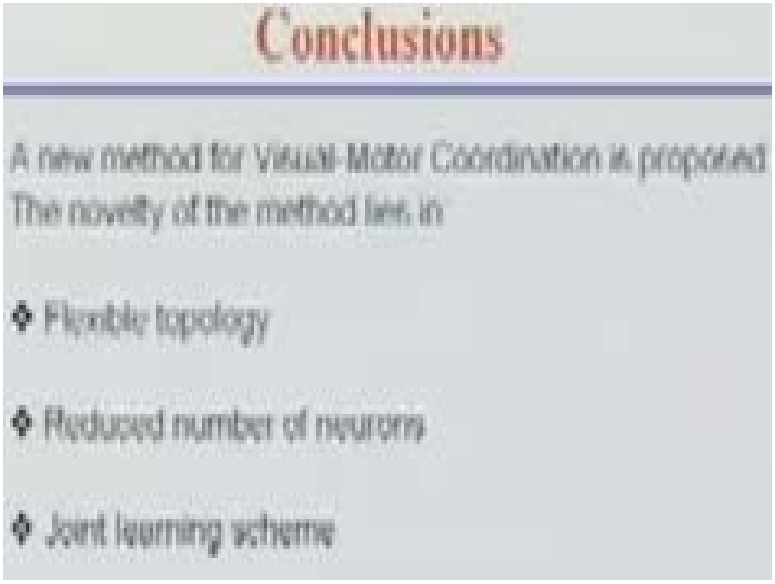
What you are seeing here is that we commanded the end-effector of a robot to track a circle, we find that the star positions, the points along which the robot have moved. The average error in tracing a circle of radius 50 mm is 0 point 0907 mm using 164 neurons. The star is the track trajectory and the broken line is the actual trajectory. A circle is tracked with an average error of 0 point 09 mm in the workspace using 164 neurons.

(Refer Slide Time: 50:18)



This is a sphere where the actual position is the red point and the blue point in the track position. We have placed the red point in the first and if we are able to see the blue point, we say the tracking is perfect. Whereas if the red is there and then the tracking is not good, a sphere is constructed with an average error of 0 point 116 mm in the workspace. That means our end-effector is able to reach all this points in the sphere with an error of 0 point 1116 mm after 5000 training steps.

(Refer Slide Time: 51:06)



Finally, we have presented a method of visual motor coordination using quantum clustering in which the flexible topologies where the number of neurons can be increased or decreased depending on the accuracy you want with time. If you want more accuracy, the number of iteration increases vigorously and the reduced number of neurons can give you equivalent and much better results when compared to fixed Kohonen SOM and joint learning scheme. It is proposed that let the clustering be done in joint input-output space instead of only input space. In this lecture, I have given you some dynamic model in the beginning and we discussed some control mechanisms. We discussed different application of learning methods to control the dynamic control. But as you know, a robot moving to any point, we are interested to know how the robot can go to any point through visual guidance. This is a little different aspect of control and we will start again some of these direct adaptive control schemes for robot manipulators as well as, we will introduce a nice experimental setup that we have in our lab which is an inertial wheel pendulum in our discussion. Thank you.