Computational Geometry

Prof. Pankaj Aggarwal
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

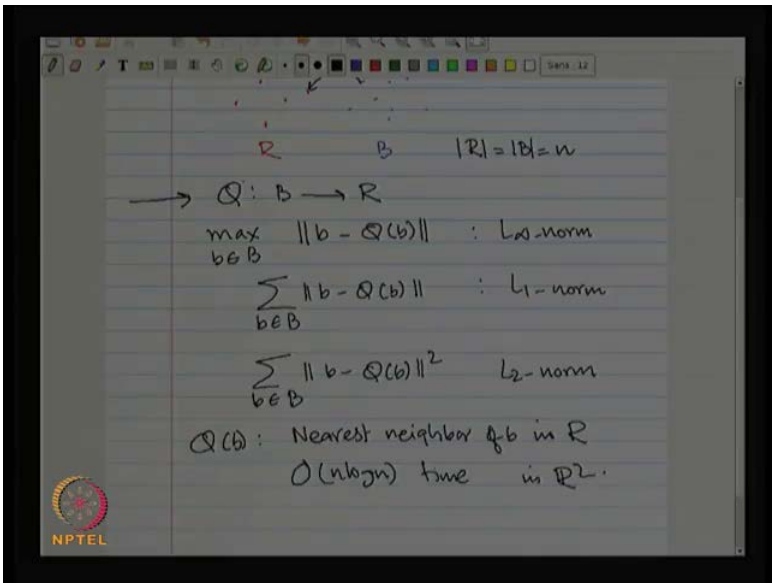
Module No. # 14
Shape Analysis & Shape Comparison
Lecture No. # 02
Shape Comparison.

Ok, I will continue where I stopped last time. So, in the last lecture, we talked about the present three dimensional shapes, focus on sampling, collecting the data, developing three dimensional models and then, simplifying those models. So that, you do not move such a crazy, but you can gain efficiency.

So, now, let us, assume that we have such models, then the next stuff you want to do is, you want to compare two such models. So, how do we compare two shapes? All of you have seen, algorithms for comparing a strings, but now, we are no longer in one dimensional world. We are sitting in multi-dimensional world. So, how we compare two dimensional or three dimensional shapes.

So, I will start by a very basic problem. And, slowly build up on and give you set of more and more difficult versions of the problem. So, let me, start with the simple version.

(Refer Slide Time: 07:38)



That, suppose, you have two sets of points, set R and then, you have another set B. So, here are two sets of points, red and blue. And, the goal is you want to see, how similar they are? That is what you have to measure, whether they are similar or different.

The first question is how we define similarity? And, there is no unique definition. And, let me tell you a general paradigm that is used to measure similarity. So, the first step is if you want to see if they are similar, then you have to do a mapping from the red points to the blue points or vice-versa. So, keep one point fix and let, I will try to map one point to others, one set to other set and see, how well I can do the mapping.

So, the first step will be to define a map. If I lets, B from B to R; and I will talk about it this map, what is, which maps make sense? Now, once you have this map; so, it means each point in the blue is said being mapped; each blue point is being mapped to a red point. Then, what you want to do is for; you want to look at the distances. So, suppose, this blue point is mapped to this point, then you want to look at the distances between these two points. Right.

If the two points are identical, then if you do the right mapping, then what will happen? This distance will be 0. Now, you do this, you want to do this map point, this look, at this value, then there are different norms you can take, you can for example, you can look at the maximum value that is one way of doing it. This is called L infinity-norm because L infinity-norm is when if you have vector, the L infinity-norm gets maximum value.

All you can take L 1-norm. The other norm that is, used a lot is L 2.norm. And, that depends on the underlying application, which norm you should choose. The max is guarantees at all the points, what the value I give you, within the distance in mapping. But, then it is a susceptible to out layers. Other extreme, it takes L 1- norm, it takes the averaging. But, then you may not want to like this, if all the points are very close to map the revel but, one point map very far is smoothing will not recognize that.

And then, some people will do L 2-norm. And, L 2- norm has certain other properties, nice properties. So, these are three norms that I used, quite a bit in practice. Now, let us talk a little bit about this map phi. What you think a most of obvious choice for this map?

If you have point B blue point, which red point you should map to? Where you will do furthest? Because you are trying to look at the similarity; so, suppose, if you have two points, let us say, I have two images, try to put them together, then the mapping you like to do is, like to see, what sort of closest you can go to. Right. So, the phi is typically that phi of b is a nearest. I will show you that there are problems with this one also. I will show.

So, let us, start with this one; nearest neighbor of b in R. So, that let us say, I do with this. Now, if I know this, if that is what, I do now, how quickly I can compute these three norms? Now, the main thing is I need to compute a given point b and it needs to compute phi b. Once, I do that and then the rest of it is pretty easy. So, how will you compute phi b? Come on, this is you all, you have seen this in class (( )) how will do Delaunay triangulation.

So, what have you learnt for doing nearest neighbor? How do you find nearest neighbor, closest point?

Plus.

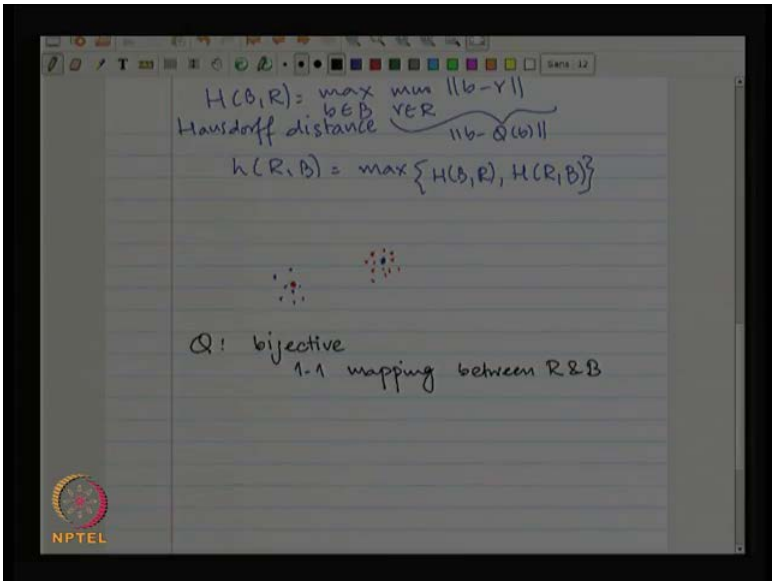
No. You do the Warnier/Orr diagram, but how do I find the nearest neighbor for a point, plus point location? So, it is Warnier/Orr diagram plus point location. Right. So, compute the Warnier/Orr diagram of red points and preprocess for point location, then when you get a blue point, using the point location find the Warnier/Orr it contains and that gives a nearest neighbor. So, that will take you. Let us assume, for simplicity that both… on

points. I just say, it is not important. But, this will simplify my notation little bit. So, then this takes n log n time in two dimensions because that is the story.

Now, this has problems in higher dimensions because Warnier/Orr diagram has largest size. And, you have to do something else. But, in two dimensions that separates it will do. Now, what do you see the problem with this map? Precisely. So, one problem is that you may have situation that the following. Let us see. Right.

So, you have, let us say, they need not to be a circle; they are some cluster. Actually, if you have such blue points, certainly, red and blue points look very different. But, in this, all the blue points will map to a single red point. And, the distance between the blue point, its nearest neighbor is pretty small. And, that does not usually be small, similarity, but they are not similar. Ok.

(Refer Slide Time: 13:46)



This, I can quickly fix. And, that is what is, so, this is so, let me, go a back little and this is called Directed Hausdorff distance and this is, I will used to notate B, R. Now Hausdorff distance is what you notice.

If I expand this business, phi b; so, if you want to write, let me, write it here, H (B, R) is maximum b belongs to B. And, phi b basically is the nearest neighbor. So, you can write it minimum. So, this is basically, b minus phi b by definition. Now, we have defined it. Am I right. So, taking the nearest neighbor; so, now, but you notice is that this is not

symmetric for b and r. And, intuitively, if a similar to b, then b should be similar to a. Because it does not make sense to say that, a is similar to b, but b is not similar to a. And, if you look at the problem, the problem with this one is that if you can do the mapping of blue to red that is very small. But, you try to map red points to its nearest neighbor and then the distance is large.

So, if you have done by miss mapping by missing, then you will not get the distance to be small. You will get the distance to be large. So, what you define is h (R, B) the Hausdorff distance is maximum distance to triple H. So, at least, the function is symmetric now. And, in this case, if you take the Hausdorff distance between red and blue points, it will be large. Ok.

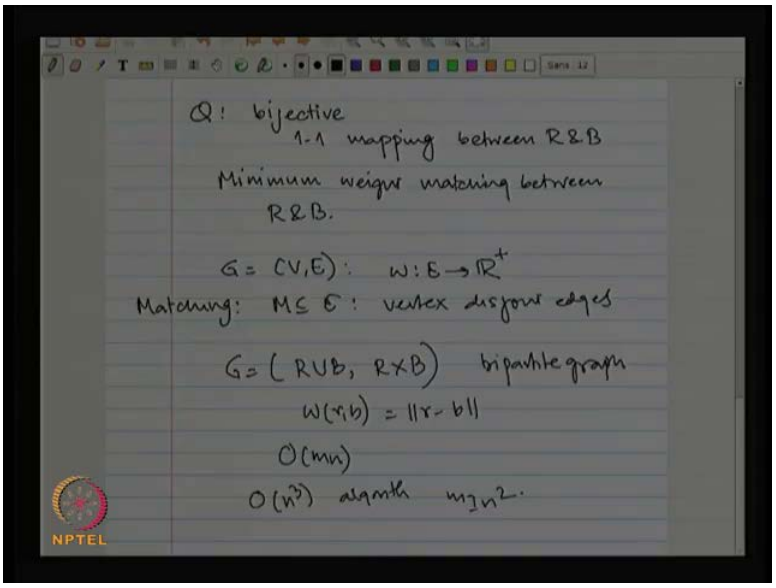
But, still one can make an example, which is it is not pretty good, is that if you choose, let say, now, if I do this point set, again they are not very similar, but again if, but if I look at the symmetric and the Hausdorff distance that is also small. So, and this specific problem is that has some of you said, is the problem is at many points or one set mapping is many to one. Many points are being mapped to a single point and that is what, you like to avoid.

So, the other extreme for phi is the following. That you see that phi is a permutation of phi is bijective; so, it means you want to do is, if the points are at the similar, same size, they are R and B, what you want to do is, you want to do One-to-One mapping. Ok.

So, it is a One-to-One mapping between R and B. Now, one has to be little more careful. What you mean by One-to-One mapping? If R and B of different sizes, then one has to more careful. But, let us, for now, let assume that R and B are the same size. So, I want to avoid that technicality. When we say that we want to do matching, so, we restrict that it should be matching. So, you like to again map to compute the function phi.

So, that each blue point is mapped to the nearby red point. But, you want to insist that no two blue points match the same red point. And, you complete the match. So, this basically, says that you want to compute. Now, if you are looking in the maximum norm, listen this is called Bottle neck matching. If you learn in the graphs, you see that basically, this is called the standard minimal weight matching. L 1-norm is called minimal weight matching and this is basically, called the RMS matching.

(Refer Slide Time: 22:08)



Now, we have seen, have you seen, computing a minimum weight matching in a graph or maximum weight matching in a graph? Have you seen in any class? How many of you know how to compute a matching? No. Come on, do not be shy. So, roughly half of you know matching. So, how many of you know what a matching is, assume that every one of you know what a matching is. So, matching is in a graph, it can be a non-map too. Here, it is a graph is going to be bipartite, but in general it is a non-bipartite graph.

So, if you have a graph, matching a vertex, so, if you want to choose a subset of edges, so, that no two edges share a single vertex. They have vertex that are disjoined. You can ask the various versions of people. You can also assign a weight, let say, may be weight limit positive x. And, either, if your graphs are unweighted, the question goal becomes finally, larger size matching. Find the matching in the maximum number of edges in a graph or if it is a graph, either, you can talk about the maximum weight matching that find the matching of the maximum weight or you can say that find the largest matching as the matching. But, among them find the matching with the minimum weight.

So, there are different versions of problem. Now, so, this is one of the earliest combinatorial optimization problems that were studied. And, when Edmonds came up with this algorithm, what is called the “Hungarian method” and what was surprising at that time was this is in fifties.

So, you have to think about, before most of you are born, if not, all of us were born was that algorithm was just an infancy people. Had not really thought about big one notation did not exist. And, we are not doing it. And, knowledge is, I think if you want to find some, like matching. Knowledge is people thought about were just doing a boot force exhaustive search.

And, then there are exploration numbers of options. And, people were surprised that you do not have to do all exploration number of options. You can just do polynomial number, doing that was a big major breakthrough at that time. People did not think that he can do this. So, that was the major break though this matching that time; Polynomial time algorithm. So, this was before NP, the concept of NP existence. So people are not thinking in terms of NP verses P. No. It is that means it is generally graphs. Yes. So, Kuhn's Hungarian method is done in basically in primal dual method for…it was done.

Ok. Now, in our case, the graph is R union B. And, the edges are R cross B. So, it is a bipartite graph. And, the weight of, in edges is basically, Euclidian distance. And, now, so, you can use a graph matching algorithm to compute the minimum weight matching.

Now, what is a running time of a matching or minimum weight perfect matching in a graph? People who know the matching algorithms? Some talk the minimum weight perfect matching not maximum matching. Pardon. Order and you will be very famous if you can do that. So, what is the best known algorithm of computing minimum weight matching, minimum weight perfect matching in a bipartite graph? So, there are no flowers and blossoms, anything here because this is bipartite graph.

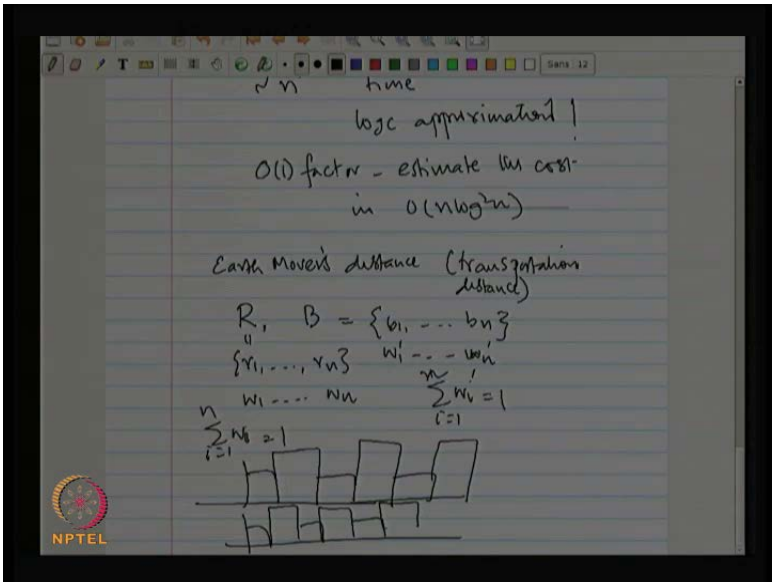
Well, we are talking about maximum matching. I am not talking about minimum weight perfect matching. So, each maximum weight matching is the minimum weight perfect weight matching or minimum weight matching? So, most of you are thinking about computing a maximal edges, maximum matching and for an unweighted graph, finding the matching with the maximum number of edges. It is not we have edge cost function and also, use Hungarian method. So, best note is m n if you know the log factors, as some log factors of m n, order of m n is a best one. Where m is number of edges and n is number of vertices.

So, if you do this, so, there are number of papers. I do not show all the, of top of my hand, had all the details. So, for bipartite graph, it is easier that was we known for a

while. But, for the non-bipartite graphs, in this paper by, it is a paper by I think galil and silver mechali and Vijay…Vijay was one who did a lot of this stuff (( )). And, Shashi do you remember that all the bipartite graph. So, this is galil, mechali… was (( )) right those. Yes. Non bipartite it is earlier Vijay… I do not remember that. Yeah. So, that is the best known whether, you can do better that is an open question.

Now, here the graph is not arbitrary. It is a geometric graph. In the sense, weights are equal distances. So, the question is what is the best one can do? And the best known algorithm for this particular set, so, if I just do that then this leads to order of n cube algorithm because number of edges is because m is n square in our case. You are correct (( )). So, questions can we do better than n cube?

(Refer Slide Time: 22:21)



And, the best-known algorithm, so, for roughly I will put tilda; These are all to know polygon factors so, roughly quadratic times it takes near the quadratic time. And, there is no lower bound. Of course, we do not know, whether one can do better than this. There is a sub quadratic algorithm to compute the minimum weight matching.

Now, this is for L 1-norm because minimum weight is some of the weight of the edges. No one can ask the same question for L infinity- norm. Right. Find the matching, so that, the largest edge in the matching is as small as possible. So, if the longest edge is as small as possible that is called Bottle neck matching. There you can do better the running time

is I think n to the power four-third they are sub quadratic algorithm. For L 2-norm is still it is the n square is best norm.

So, for L 1 and L 2-norm can we do better? That is not known. And, then ask a question that well, I do not want to spend quadratic time is a bastian algorithm. If I relax the condition to find the minimum weight, is there a faster, is there fast approximation algorithm? And that sort of is… and then the situation becomes really murky.

Yeah. So, for L 1-norm for minimum weight matching, there is an algorithm that can find constant factor approximation in a linear time. But, the running time depends on the constant. So it is something like, for it is a, so, what is sort of is, let see, I forgot the running time. Running time is something like n to the power 1 plus 1 over c time. You can find log c approximation. So, what is not known is can I do factor two approximation in n log n time or something like that would be nice. But, if you do not insist on finding the matching, if you just want to estimate the cost of the matching, then you can do the constant factor, estimate the cost in I think, in I do not remember, n log n or n log square n time. Let me, write n log square n time. So, you can within n log square n time, you can estimate the cost of minimum weight matching within cost and factor, but we do not know, how to compute it. That is, it is a strange, but that is all it is.

For L 2-norm, for this norm the best-known algorithm, if you want to find within cost and factor, the running time is center path three half. It is an open question, whether you can, in L 2-norm you can estimate the cost, even the cost or compute the cost matching with the cost and factor in nearly linear time n log or n square, something better than three half that is not a problem. So, these problems are all open for graph. So, what I am pointing out to make is the very simple problems, we do not have very good algorithms. This is generalization of matching. So, that is called the Earth Mover's distance or also, it is called Transportation distance. Which is used nowadays is a following. That you have let say, let me, describe with the, for the graph or for points are it does not really matter. That is, we have two sets R and B, let us, see that points and the name of the same size, but it does not matter.

And, then you have given a weight here, let see and their weights w 1, w 2, up to w n. And, so, that let us, just normalize it for the simplest this is 1 and this is 1. So, what we should think about is that you have some material that you want to transport and what

you can sort of, know is you can take only this much from w i and you want to sent it some distance to some other node. So, transport it. And, of course, all the material that is sent from r i should be w i. And, all the material that should reach to the b i, b j, it should be w prime j. Am I right. So, that is what, you want to do.

And, you send it along the edges. And, total transportation cost of sending, whatever x item is proportional to x times the distance. And, you want to minimize the total transportation distance. So, why is this sort of comes naturally? It has become naturally in comparing, in measuring similarity between two distributions or two images. So, think about that you have two distributions which represents as a histogram. So, let say, this is one probability distribution and here is another probability distribution that is it. And, I want to… how similar they are. So, what you think about that, this is a histogram, that I have built, is a pile of sand and what you want to move the sand from here to the places; so that, you get the destination distribution. Ok.

And, the question is how much sand you have to carry? And, that is why, it is called Earth movers distance. So, it is used a lot for measuring the…, as you working more and more data with a certainty machine distributions that is used quite a bit and factors and also comparing images. Suppose, you have two grayscale images and you normalize them that the total level of the gray level is one, some more the stuff and you want to see, how similar the two images are. And, this is as hard as matching problem. So, again the question is finding exact solution of…, what kind of approximation one can do.

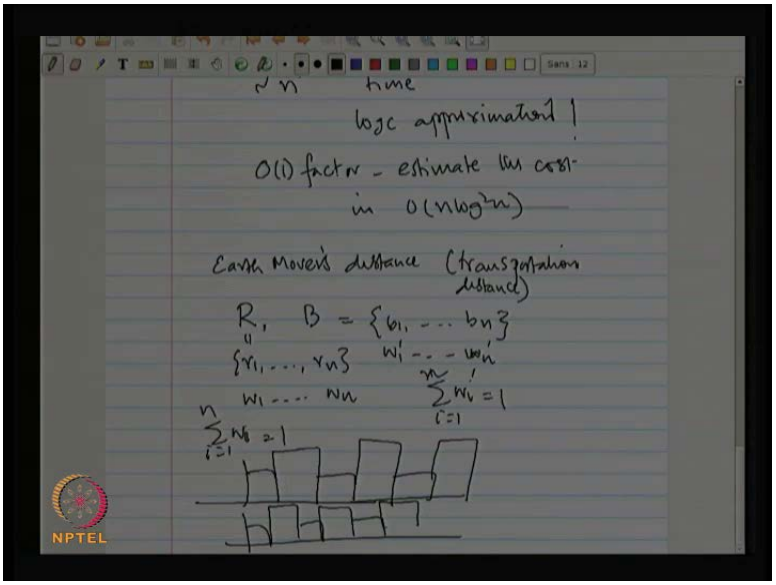
So, distributions are, two distributions are given to you and of course, you can either have deterministic algorithm or you can have random …Montec or Las vegas algorithms or both type of algorithms are there. Does that answer a question?

So, the other measures also, for distributions, but in many applications, this is, people are using Earth Movers distance to measure similarities between two distributions. The statistics has come up with. Has defined many different other measures also, for measuring distributions. They are in the probability theory, there are other measures. I will not go into there because this is specially used for algorithmically; If, any of you know, any much about Embedding? There are some results that Earth Mover's distance can be embedded to some high dimensional space and then, you can compute it. So, they

are some Monte Carlo algorithms, for measuring the Earth Mover's distance, for example.

So, this is how the issues are mentioned so far is about, what the function phi should be? Now, the second question that comes up which I have not discussed so far. But, assume the co-ordinates of points R and B, are fixed. Right. But, in the practice, what happens is that let say, you have two objects. You had, each of them was generate in its own coordinated space. Right. So, you will, let say, I have these two pens, so, let say, they are sitting here and let say, I have fixed the coordinates. Now, if I use this existing coordinates and they look very different because we do a matching identical. So, what let me, able to do is I should be able to translate it and find that it was translated; so that, it tries to align them as much as you can or this may be a different poses, they are not fixed. So, I should be able to rotate that. Right. So, the question you can ask is the following.

(Refer Slide Time: 33:08)



So, R, B is, and let me do the following. Let me introduce a notation, so far, in this format that, I talked about as general. Let me, define R, B as a whatever choice, you choose to measure the distance, similarities of the distance between R and B two points to the fixed embedding, embedded in the space. Now, what I can talk about is the following. Given that mu, find so, translation plus notation that is called the rigid motion.

So, look at the space rigid motion. So, if especially, someone taken any question completed graph, if you must seen this a lot about the, how we measure the rigid matrix. But, in two dimensions, rigid motion can be represented by, three by three matrices, so, the sort of, all the rigid, all not; any motion should have set a property. Not, every matrix corresponds to a rigid motion, but you can talk about the space of rigid motions. All possible rigid motions, you can think about as a three by three, set of three by three matrices or normal matrices plus translation that we do.

Rotation is also normal matrices. Basically, they are, that is what, corresponds to. But, I will not go into that. So, let us say, let me call, phi is a set of all rigid motions. So, this we should think about, all three by three matrices that correspond to a rigid motion. Then what I want to do is, compute minimum pi. So, I am going to keep one set fixed and other set, I will allow translation, rotation. So, what I am sort of, saying is that, we are given a point sets and you have chosen your favorite distance measure or similarity measure for the two points sets. Now, what the game I want to play is, one set, I keep one set fixed, other set I try to translate and rotate and align it.

So that, the similarity as small as possible. Now, there are many examples of this concept. Let me, just give you one, which is the permanent nowadays is with that comes in Molecular Biology. So, ours, one of the basic structures in our cell or in our life or in our body is said as proteins. Am I right. And, the one of the (( )) discussion is to understand, how the system works is, to understand how the proteins interact. And, the proteins, how the proteins bind each other?

And, there are sort of, various issues, but one of the other things is that, you want to generate new proteins and generate new molecules. But, an underlying problem, so, this last, I will say, for may be twenty to twenty five years, people have basically, said three dimensions structures of proteins. They are too complex. I will just work with sequences and that is why, Genomics are focused on. Now, what may happen is that, Genomic, when you work with the sequences, you capture only certain similarity.

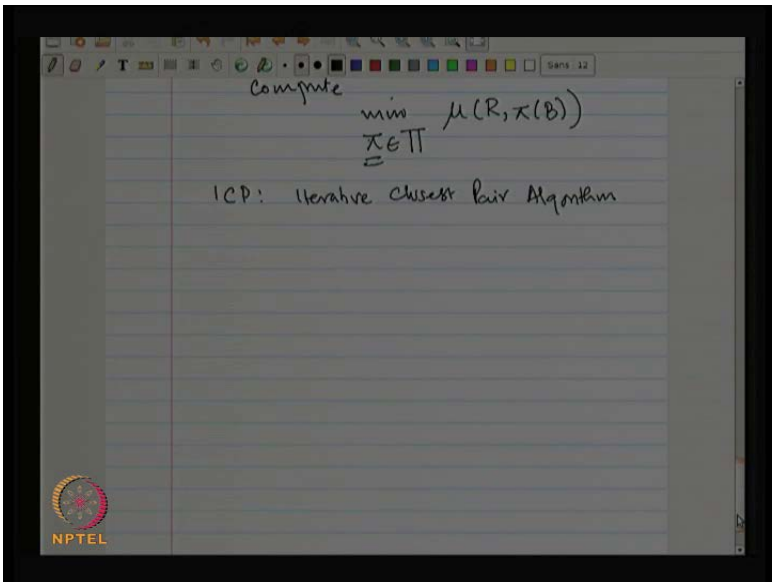
You may have two proteins, if there are very different sequences, but their structure might be similar and they interact with each other. So, just comparing sequences is not enough. You have to look at the structures. So, last, I will say about last ten years, there has been much more work in comparing a structures, molecular protein structures. And,

if you know about any protein, there is a backbone and the site chase for protein. And, the backbone, you can think about polygonal change sitting in three dimensions. And, what you want to do is, you have one backbone chain and another backbone chain. You want to take the one backbone chain and translate and rotate it and see how fast you can align it. That is what, of its aligning and translating and rotating it.

Ok. So, that facility given application comes. And, there are many other applications one can give, where you want to do this. Now, let us, look at the some other very basic stuff and then, see what we can do. Let me, sort of, even though, we do not, we did not like this, let us, take nearest neighbor and let me, just take L 2-norm. And, I will not say why, but L 2-norm has max properties, lot of convexities, exist in the function.

So, this is the major of a mu. Basically, each point is major to its nearest neighbor. And, I look at the sum of the squares of distances. Now, that is mu. Another question, I will ask is, whether, how you compute this function under rotation, under rigid transform or rigid motion. And, the problem is not known, whether it is P or in NP or NP-hard? That is not, there is no polynomial (( )) to find the optimal solution.

(Refer Slide Time: 39:51)

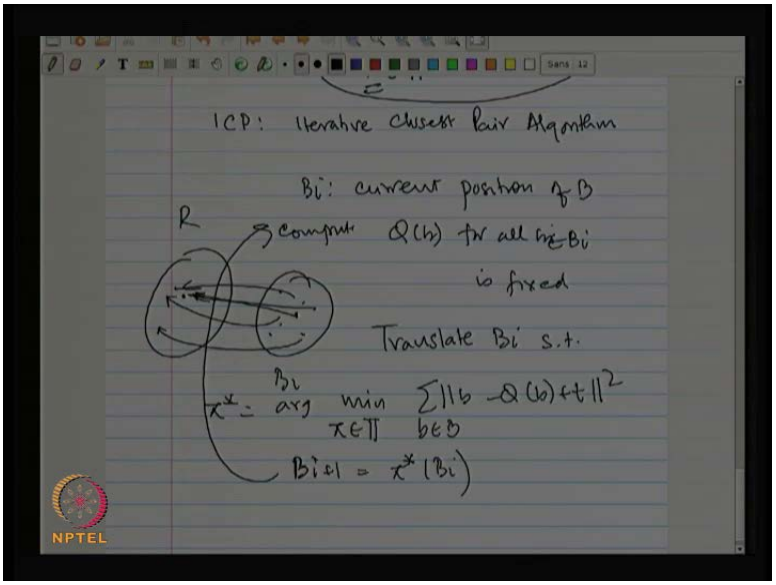


There are approximation algorithms one can do and there are some heuristics, the one very widely used is, that is called ICP: Iterative Closest Pair. And, here is an algorithm so, there so, if you look at this algorithm, there are two choices. There are two. You think about, there are two transformations; one is pi, which are alignment, rotation and

translation. And, other is map phi, which is mapping each point to its nearest neighbor. So, algorithm, what it does is it iterates, it is, by as a name suggests, it is an iterative algorithm.

So, you have current configuration of red and blue. Let us say, red points are fixed and blue points are also fixed. What you do is, for each blue, this function keeping the mapping fixed, keeping phi fixed.

(Refer Slide Time: 40:39)



So, let say, that R, so, R is let say fixed; only B, I am trying to translate and rotate. So, let say, B i is a current position of B. So, it means that B, I have fixed that transformation, it sits in the plane. It is the (( )) B i position.

Now, compute phi b, for all b in and B i. Find its nearest neighbor. Now, I have done the mapping. Now, here is your set of red points and here, is your current B i and you have mapped each points. Now, you say that, I keep this mapping, I am not going to change the mapping, but let see, how much I can translate and rotate; So that, the sum of the squares of the distance is as small as possible.

And, surprisingly, it tends out to be, convex approximation problem. And there is a close form solution for this, which I have not told you. Because deriving is say, something, but let me, tell some of the following. If you forget the rotation, if I do only translation, so, you have let say two point sets and I fix a mapping from each point here to each point

here, and I can only translate this one; and so that, the sum of the squares of distances between the map points is as small as possible.

And, can you guess, what the translation will be? So, the question I am asking is, that phi b is fixed, translate B i, such that, minimize the translation. So, look at that, just try to translate the point set B i, keeping the mapping fixed; so that, the sum of the square of the distances between the map points is as small as possible. Let us, anyone to make, how will you find this translation? There is very simple answer to this. Pardon.

So, what is average mean? So, can you be more specified what do you mean by average? So, what is that called? What is the mean of, if you take mean of phi B i, what is that called? Whether there is a term for that? Centroid.

So, what you do is, you align the centroids. So, take the translation, you have a p points here, centroid here and your phi b, the centroid here. Do the translation, so that, the centroids map through the centroid. That will minimize the translation. And, I will not have time to do the derivation. When we do the rotation that describe (( )) to allow compute Eigenvalue of a 3 by 3 matrix.

This is standard thing. If you just do this ICP, on where, you will find the formula. All the derivation is a beautiful derivation. Doing it so, one can do. So, this is one can find the translation, rotation, pi, rigid transformation; so that, for a fixed mapping you do that. So, you do that. So, compute, so, let me, sort of, generalize it. So, find this translation. Am I right. Not translation, rigid motion. Map it there. Now, you have new B i. The B i, so, let say, arg min pi star is pi star B i.

Now, you got a new point set, the B i. What you did? You started B i, you moved it to the new position. Now you repeat again. You find the next, its nearest neighbor and do it.

If the nearest neighbor did not change, then you have done because it nothing changes. So, you are, you should be, but if it is different, then you repeat. What you can argue is that, you cannot argue that, this algorithm converts to a global optimum. But, you can argue is that, it will always converges to a local minimum.

So, what people do in practice is, do it few times to start from different starting points. You choose, let say, some by some, there are many heuristic, how you should start an initial staring point and from there do it basically.

Now, other thing is that is, now, recently few years ago, it was proved, that it does not necessarily converging in polynomial graphs. That was big open question, whether this algorithm always converge in polynomial number of stack, not known. It may take converge in polynomial number of stacks.

But, what happens is that, it is like (( )) method. So, it basically, makes a lot of progress and when you reach optimum, it just starts scrolling. And then, you can cut off at that point. In practice, when the gain is not much. So, this is used as quite a bit in this approach.

Now, when I can ask a simple question for matching and then, the problem is completely, widely open. So that, if you have a two point sets here; I find the rigid transformation, so, the matching, minimum weight matching is as small as possible.

And, we do not know, whether this is in P. We do not know, whether it is NP-hard and we do not have very good approximation algorithms either. Now, let me, sort of, give you, sort of, let us, start more ambitious. You ask, why stay stop at rigid transformation? For example, you are sitting right now, that say, but if you stand, the points that will look very different. Am I right. But, you are the same person, identical like this.

Suppose, if you are a security person, if I want to look for people, I should able to identify that, they should be same and they should be too sort of. So, one can talk about more complex transformations, doing good. And, there the people have just begun to think about, that problem that exits. The state of art is pretty bad, when you allow, what is called flexibility or non-rigid transformations doing it. If you just pick up the few sort of conferences, there is called symposium of Geometric progress, Processing Siggraph. There are some computer visions, you will just see some heuristic, and people have come up. People have just do not know and the problems becomes much harder. Some versions of this problem are known to be NP-hard.

But, it is a completely open area for research. How you do this? And, there is a beautiful method, technology that have been developed and trying to understand, what it means, to

handle knowledgeable transformations. Also, going back to proteins, proteins are not rigid. Proteins are flexible. And, there also, you want to do alignment that is a biggest challenge. You want to do alignment keeping flexibility into account. If you want to stop, ok. Thank you.