Algorithms for Protein Modeling and Engineering

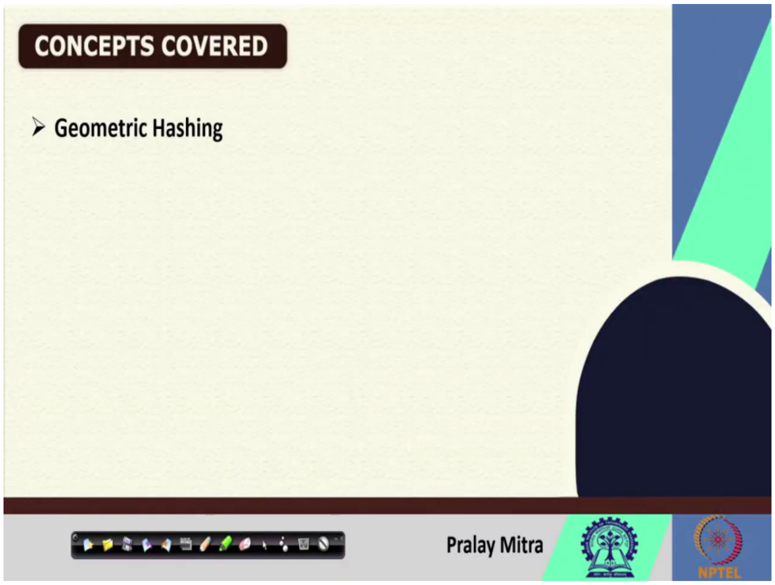
Professor. Pralay Mitra
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No. 12
Geometric Hashing (contd.)

Welcome back. We are discussing geometric hashing. And in this context what we started to discuss is if we deal with all the atoms in a protein molecule, then it is time-consuming. Gradually you will be able to understand what I mentioned in one situation when you are generating all the possible transformations and storing that in a database then some storage is required.

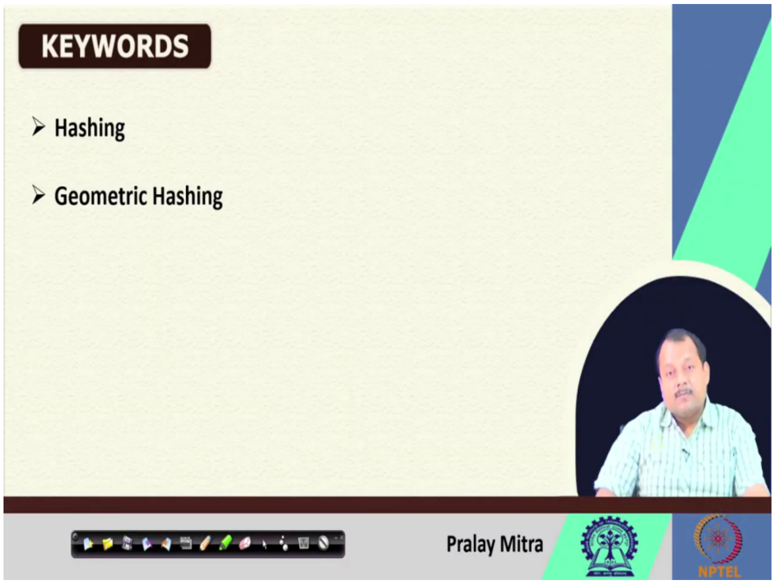
So, it will be directly related to the amount of space and also if the size of the database increases and you do not have a specific search technique, then the search time will also increase. But, I also mentioned that since we shall go by the hashing technique, then searching may not be a concern, but storage might be a concern. That is why we started to work on reducing the number of points which in the context of the protein molecule is the number of atoms of amino acid molecules.

(Refer to Slide Time: 1:48)



CONCEPTS COVERED

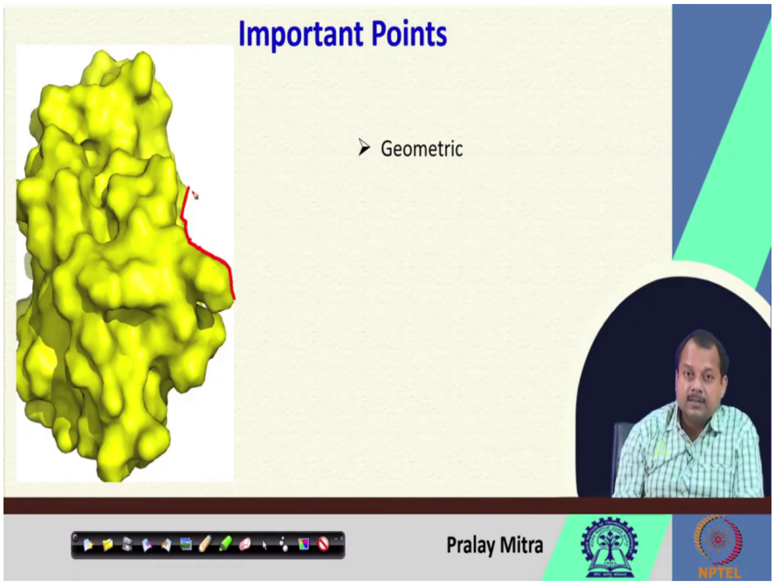
- Geometric Hashing



KEYWORDS

- Hashing
- Geometric Hashing

Pralay Mitra



Important Points

- Geometric

Pralay Mitra

First of all, I mentioned that it will be on the surface, and even on the surface all the atoms are not necessary. We mentioned that following some physicochemical properties as suggested by the chemist or biologist, we shall consider only those points which might be relevant for anchoring the orientation.

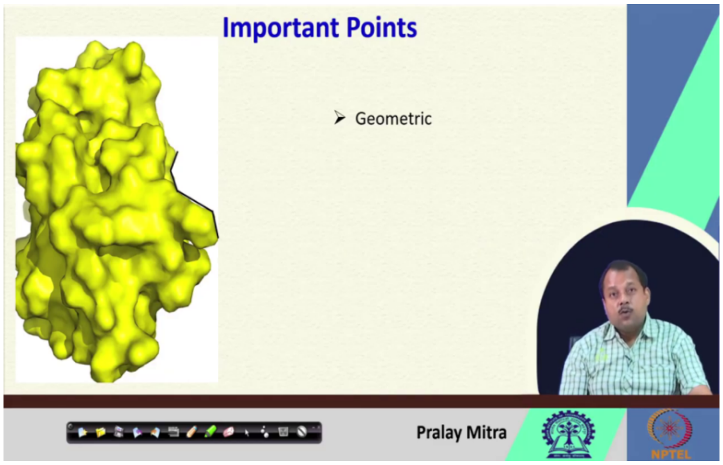
After anchoring that orientation following the geometric feature, we should look for the following physicochemical feature. We shall look for the geometric features also. Now, in the case of geometric features, I mentioned that if I start from here, then go by this, then go by this, then by this, then by this. So, every time I am making a change in the direction of the movement say going upward then moving to left or right or diagonally up or then diagonally up left or diagonally upright or say downward left, downward right.

In this way, if I change the direction, then I can consider that a particular point can be critical to declare the shape of the structure. The shape is also going to contribute when I am going for surface matching or generating the orientation. Geometric as well as physicochemical, combining that two information I shall select a subset of the atoms which reside on the surface of the molecule and then I shall work with that one or I shall restrict my generation in the searching for the different orientation using that one also.

But you should not misinterpret this definition with the fact that after generating the orientation when we are going for calculating the surface matching, then we have to consider all the atoms which are on the surface and all the atoms which are inside (to penalize). You remember from our previous class that if some penetration happens that is purely computational. Biologically penetration is not possible practically/experimentally.

But computationally, when we are modeling, then it may be because I am writing some code and that code allows me to go inside unless otherwise, some sort of penalty is imposed on that one. To make it practical, make that implementation practical we have to incorporate penalties. During the matching score, we have to consider the inside atoms, surface atoms, and all the atoms.

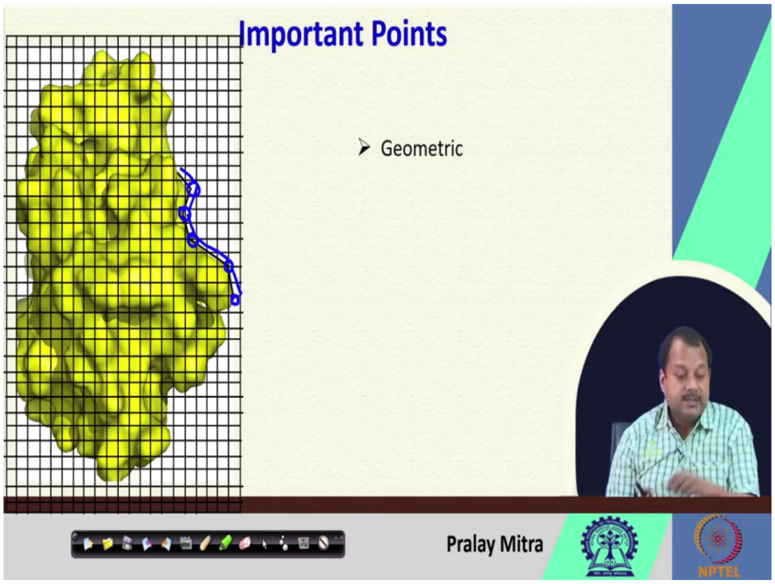
(Refer to Slide Time: 5:11)



Important Points

- Geometric

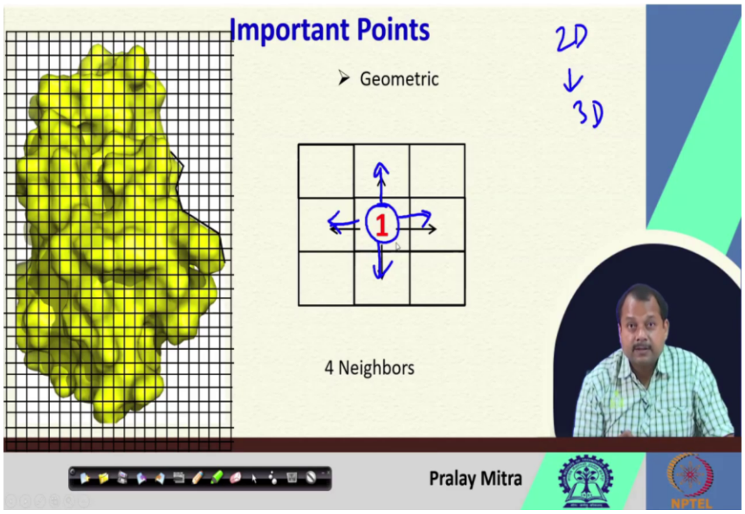
Pralay Mitra



Important Points

- Geometric

Pralay Mitra



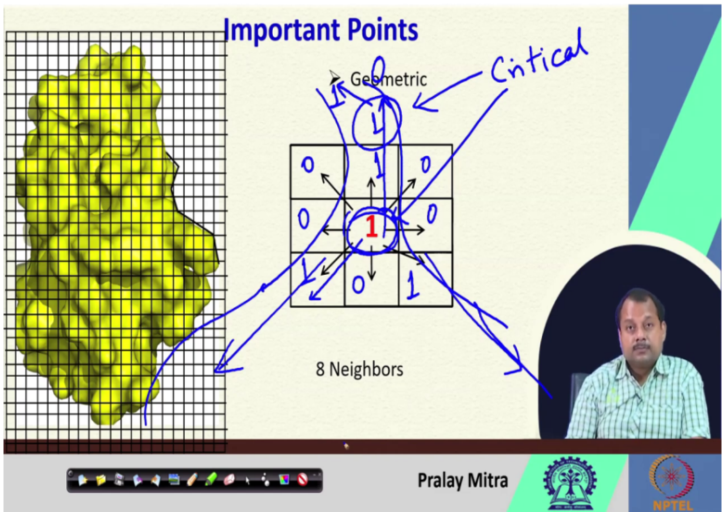
Important Points

- Geometric

2D
↓
3D

4 Neighbors

Pralay Mitra



Important Points

- Geometric

Critical

8 Neighbors

Pralay Mitra

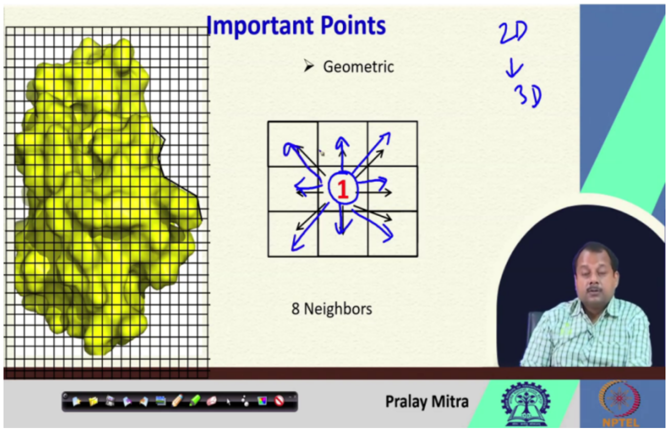
These are the lines that I have drawn. You can see that the lines are the black one. After this is the black one next. It is the black one then the black line, and that way it is going. Now my job will be whether to use this one or not and if I wish to use this one, then how can I use this one that is also my concern. Now, if I wish to use this one, then what I will do? I will put that in a grid. Again the old strategy is adopted.

After putting in a grid, you probably noted that that grid is of a very small step size. If I assume that this step size is 0.5, then in one cell at most one atom can be accommodated. Now, you can figure it out, or let me color it so that you can able to figure it out. This is my starting point. The line is following this way. And this is another point where the inclination changes.

Next, this is my other line and here inclination changes another line, inclination changes, another line inclination changes like that way it is going. In this situation, to know whether that particular atom or the point is contributing to a critical one in the context of declaring or defining the shape of the protein molecule. So, what we can do, that we can look for its neighbor.

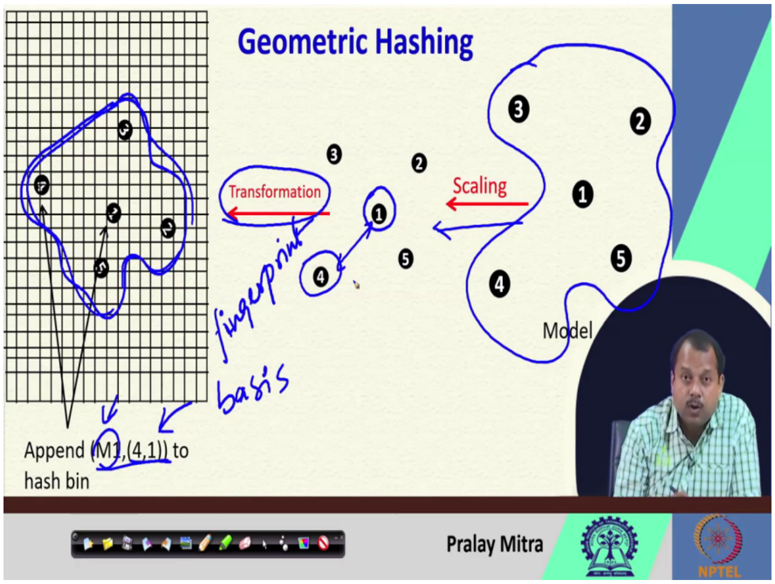
All those atoms are declared on the surface. Now, through that way, I am passing this straight line. I am demonstrating in 2-D. But you can extend that one to 3-D. Let us assume this is my atom position in one grid cell. If this is my atom position, then following this what may happen that after this one I will look for its neighbor. Now, again so if I look for its neighbor, then it may be a 4-neighbor an 8-neighbor. In the 8-neighbor we have a few more. So, this one, this one, this one, this one.

(Refer to Slide Time: 8:49)



Important Points

- Geometric



Geometric Hashing

Append (M1,(4,1)) to hash bin

The simplest thing you can do - checks whether you are deciding the critical point based on the 4-neighbor concept or the 8-neighbor concept. After deciding on that in one direction it is moving after that one what will happen is that this is 1 but there is 0. So, there is a change of direction. If there is a change of direction you mark this point and declare it as a critical point.

Now, in this way, see you are going but suddenly you are moving this way. After this one, this is one but this is 00 and say 0000, then you declare this point as a critical point. Now, if it is a situation like say, this is not 0, but this is 1. If it is the situation, then you see in which way the surface is forming. So, whether this way surfaces or this way the surface is forming or say surfaces forming in both ways.

If both the ways, then this is one critical point after that one following this you are going to get one critical point following this you get another critical point. Following this, you are going to get one critical point, and following this, you are going to get another critical point. So, once you will have this critical point information, then combined with the physicochemical information you are having a subset of the points on the surface of the protein molecule.

Now, up to now, or until now what I mentioned is considering the atom as the point, and for the speed of your calculation you can digitize that point or you can put that protein molecule inside a grid cell, and from that grid cell, you identify in a grid, and from that grid, where that particular point belongs and from there you decide to start working.

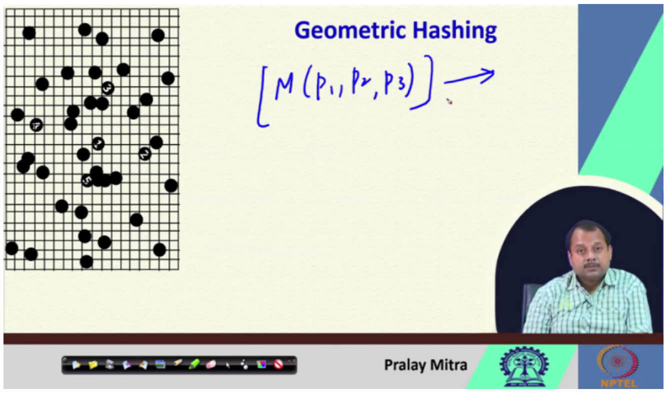
Until now, I am working with that assumption. I will change that one this week itself. Now, I got a few points and clearly, those points are not all the surface points on the surface, but a few points are a subset of the points. If I assume that subset of the points on the right-hand side, then, these are the points on my model. So, 1, 2,3,4,5. Now, I can do some scaling.

I am doing some scaling although. I mentioned that scaling we will not allow for our protein molecules. If I allow scaling, then the dimension will change and in that way, I have to scale not only the one molecule but all the protein molecules on which I am applying my algorithm. So, better to avoid that one. But if I do that one, then I will get some sort of scaling like this. But what is this transformation?

On this transformation, if I apply this transformation, then you can see that there is some translation and rotation visible that rotation you see that the numbers or the inside of the circle black circle are also kind of tilted. Now, this is one situation, and in this situation, if I assume that 1 and 4 are my basis points, then for this model. I will make an entry to my table which is nothing but the hash table with the M1, 4, 1. What is M1? M1 is this particular orientation and 4, 1 says that is my basis.

I mentioned the basis coordinate in the affine transformation. So, with this respect to these bases, this is my model in M1. What can be there, in M1 other physicochemical information or say molecule-related information or say surface represented, etcetera, whatever you wish to store. You store here a fingerprint used for that information. That is only one transformation, and based upon that one, I got one model and that transmission is using this 4 and 1 as a basis.

(Refer to Slide Time: 14:21)



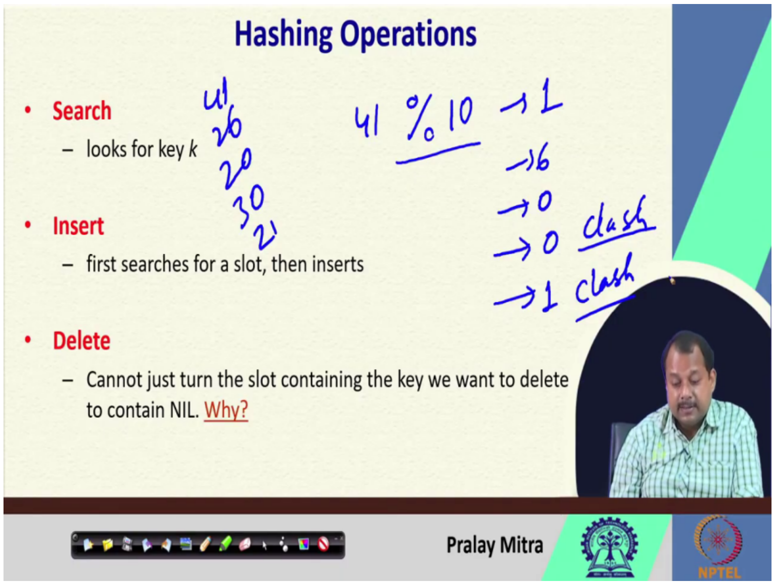
Geometric Hashing

$[M(p_1, p_2, p_3)] \rightarrow$

Pralay Mitra

Now, if I perform several such transformations, then. I may populate the grid like this. Corresponding to each such transformation, I shall have one entry in my hash table where say the model is followed by so one point say P1 another point P2 and if say I am using triplet, then it will be P1, P2, and P3. So, this will be the entry to the hash step.

(Refer to Slide Time: 15:00)



Hashing Operations

- Search
  - looks for key *k*
- Insert
  - first searches for a slot, then inserts
- Delete
  - Cannot just turn the slot containing the key we want to delete to contain NIL. Why?

41
26
20
30
21

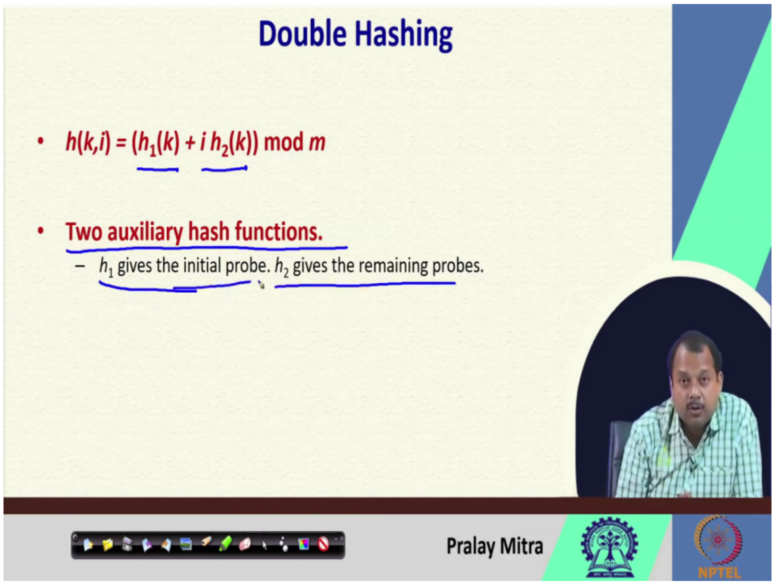
41 % 10 → 1
→ 6
→ 0
→ 0 clash
→ 1 clash

Pralay Mitra

Now, going back to that concept of hashing operation just to refresh. In hashing operation, what we did do? We did the searching, we did insertion and we performed deletion and also I mentioned that deletion is not possible. The only you can do is the superposition.

Now, you also remember that we discussed several issues like collision, say you have one that has function and that has function say my hash function picking the same one. So, that you can remember, mod 10. If I take that mod10 is my hash function, then if my numbers are say 41, 21, 20, 30 or say this is my 26, then 21 here. If I go this way, then for 41. I shall get 1. For say 26, I shall get 6. For 20 I will get 0. For 30 there will be a clash for this one, and there will be another clash. So, these clashes are there. There are two ways to remove those clashes.

(Refer to Slide Time: 16:39)



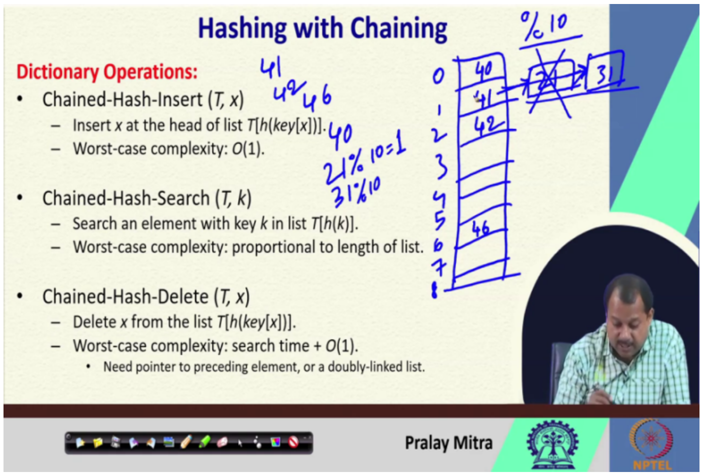
Double Hashing

- $h(k,i) = (h_1(k) + i\, h_2(k)) \bmod m$
- Two auxiliary hash functions.
  - h_1 gives the initial probe. h_2 gives the remaining probes.

Here to remove the clash, two different most widely used techniques exist. One is double hashing. When there is a clash using say one hash function you can use another hash function the simplest one is that h, k, and i. h(k) is my primary hash, then i h 2k this is my auxiliary hash. If there is a clash because of this h(k).

I mentioned that probably you can take the say percent mod of 10 and there is if there is a clash then you can go by div of 10. So, those are the different techniques. The two auxiliary hash functions we talked about also h1 and h2 give are the remainder and the initial probe. Now, if you use these double hash functions, then to some extent you can remove the collision.

(Refer to Slide Time: 17:40)

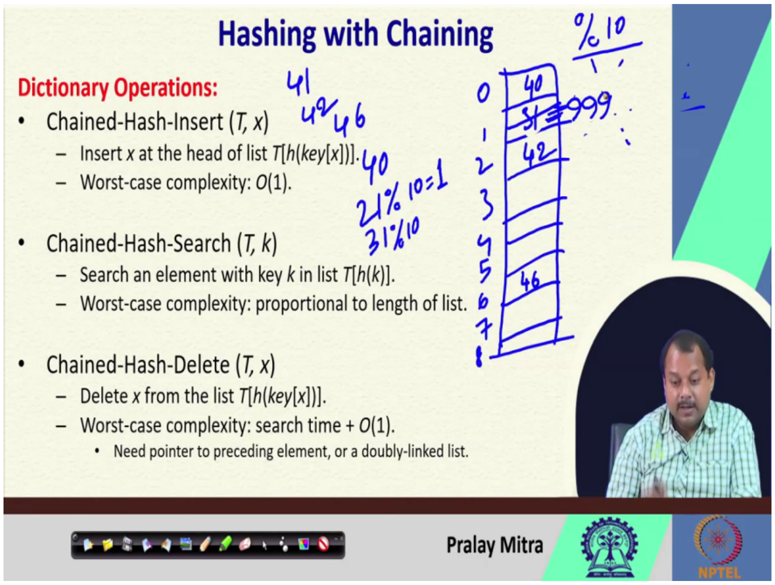


Hashing with Chaining

Dictionary Operations:

- Chained-Hash-Insert (*T*, *x*)
  - Insert *x* at the head of list *T*[*h*(*key*[*x*])].
  - Worst-case complexity: *O*(1).
- Chained-Hash-Search (*T*, *k*)
  - Search an element with key *k* in list *T*[*h*(*k*)].
  - Worst-case complexity: proportional to length of list.
- Chained-Hash-Delete (*T*, *x*)
  - Delete *x* from the list *T*[*h*(*key*[*x*])].
  - Worst-case complexity: search time + *O*(1).
    - Need pointer to preceding element, or a doubly-linked list.

Pralay Mitra



Hashing with Chaining

Dictionary Operations:

- Chained-Hash-Insert (*T*, *x*)
  - Insert *x* at the head of list *T*[*h*(*key*[*x*])].
  - Worst-case complexity: *O*(1).
- Chained-Hash-Search (*T*, *k*)
  - Search an element with key *k* in list *T*[*h*(*k*)].
  - Worst-case complexity: proportional to length of list.
- Chained-Hash-Delete (*T*, *x*)
  - Delete *x* from the list *T*[*h*(*key*[*x*])].
  - Worst-case complexity: search time + *O*(1).
    - Need pointer to preceding element, or a doubly-linked list.

Pralay Mitra

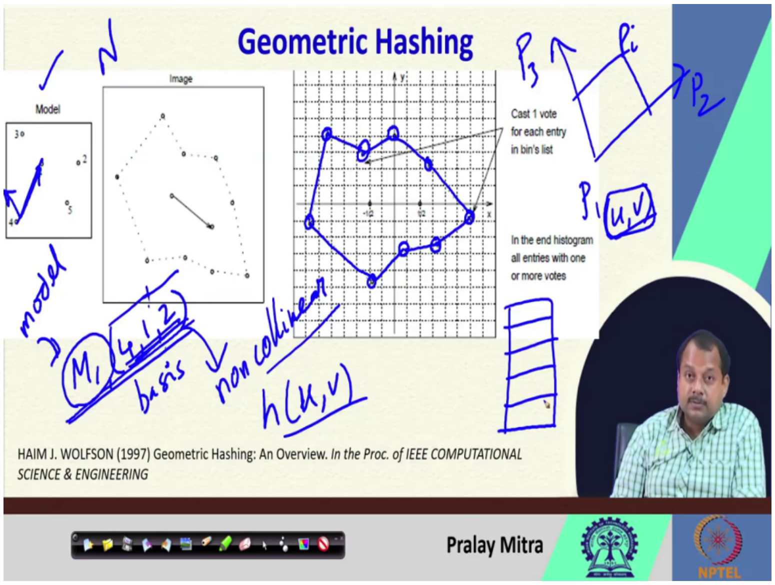
Another technique that will be utilized in geometric hashing is chain hashing. In the chain hashing there is a hash table like this 0, 1, 2, 3, 4, 5, 6, 7, and say 8. I am assuming that my hash function is say mod of 10 simplest one and my entry is say 41, 42, 46, only 40, 21, and 31. When it is 41 it will be entered here say 41, 42 entered here, 46 entered here, 40 entered here, and next 21 there is a clash.

When I will go, 21 mod 10 equals 1. So, when there is a clash, what I will do? I will form a list with 21, then I will get 31 that will also be appended here. So, that way I am creating the chain or the list in the same location. Now, if that way I go then what will happen? My hash function remains one single hash function, the primary hash function, and whenever there is a clash, I shall go for the chaining or the listing. During the deletion delete x from the list. So, here you will see the actual deletion will take place.

When there is a deletion - say I wish to delete 21? What shall I do? I shall erase this one. So, this link will directly go there. So, deletion will be a valid point here. But of course, here if there is an after 21, I wish to delete 31, It is fine 31, or after 21, I used to delete 41. So, what will happen this 31 will come here and this part is deleted. Now, if I wish to delete 31, then overwriting is required physical deletion now is not possible.

I suggest probably you will replace it with -999 assuming that the input numbers all are positive, that kind of arrangement you can do. Now, for our purposes, we will see that neither of these is required because whenever there is a collision that is a good news for us. That is a good news for us in the context of the geometric hashing, then we will just keep on adding the board that there is a collision we will keep on voting on that one. That we will discuss, but these are the concepts that we will be discussing or utilizing for this hashing.

(Refer to Slide Time: 21:03)



Geometric Hashing

Cast 1 vote for each entry in bin's list

In the end histogram all entries with one or more votes

HAIM J. WOLFSON (1997) Geometric Hashing: An Overview. *In the Proc. of IEEE COMPUTATIONAL SCIENCE & ENGINEERING*

Pralay Mitra

Now, this geometric hashing. Now, in the case of geometric hashing. What we are doing? So, please note. So, this concept I have taken from one publication of Wolfson, also the image. I have borrowed from there. In this grid, you can see that in this grid I am placing them. There are the points, I think it is clear to you. So, still, I am going to draw this one after this one, there is one point till come down.

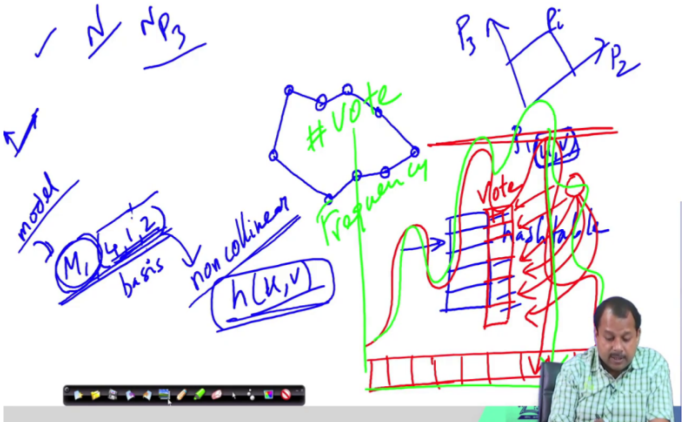
Next, there are some inside points also here. But primarily, what you can note is that these points are defining the shape. And all those points are as for the geometry is changing their direction. So, from one direction they are moving in another direction. What shall I do I shall start with this model. So, there are five points 1, 2, 3, 4, 5 and then I shall go for the transformation and while.

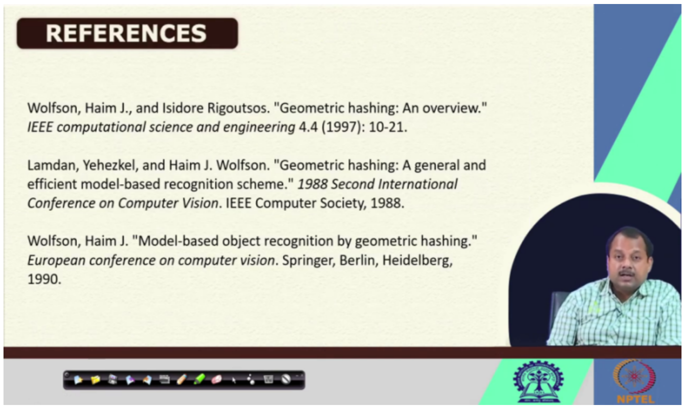
I am doing the transformation and then I shall consider one basis point to declare once an axis with that one. These basis points say I am assuming this one and based upon that one that basis point. What am I doing? I am generating this model and that model information. I am storing it in a hash table or hash function.

Now, I generate a different orientation and in the previous slide, I have shown you that you can write as M1, 4, 1. Where this 4,1 is my basis point, basis quadrant, or basis and this is my model. Similarly, corresponding to each basis if you have N number of points then I am considering not 2 but say 3 basis points. What is the limitation of these basis points? I mean what is the property which should be obeyed to form a basis these three points should be non-collinear. These three points should be non-collinear.

Now, if they are non-collinear, then by using them you are forming one basis coordinate, and in that coordinate what is the model information you are picking? Corresponding to these 4, 1, 2, or these three non-collinear points or triplets you are defining one basis. You remember our diagram where there were three points P1, P2, and P3 and there was one point Pi. So, by taking the projection u and v was affine invariant. From these 4, 1, 2 corresponding to each point such *u* and *v* that affine invariant values I can compute. I can use that for computing my hash function, and directly I can store this information in that hash table.

(Refer to Slide Time: 25:37)





REFERENCES

Wolfson, Haim J., and Isidore Rigoutsos. "Geometric hashing: An overview." *IEEE computational science and engineering* 4.4 (1997): 10-21.

Lamdan, Yehezkel, and Haim J. Wolfson. "Geometric hashing: A general and efficient model-based recognition scheme." *1988 Second International Conference on Computer Vision*. IEEE Computer Society, 1988.

Wolfson, Haim J. "Model-based object recognition by geometric hashing." *European conference on computer vision*. Springer, Berlin, Heidelberg, 1990.

This is my hash table. I have three such points. I have to check which are non-collinear and accordingly, you can generate all the models in this hash table using their hash function say $h(u,v)$. Whenever another matching object or matching molecule comes, then accordingly you can generate one hash function. If you see that already one such model exists, you did not have to store this information now, previously the information was stored, but now what you will do that you give one vote there. And when, I say that you give one vote there, then you can think of the existence of one variable attached to this hash table. That will not store the model information. But that is indexed with the hash. But this will store the voting information. Initially, they will be initialized with 0. And whenever there will be one match found, this will be incremented by one. At the end of the generation stage what you have is this histogram of this hash table.

Histogram indicates that the one-dimensional array stored the frequency of the information. So, the frequency information you have. From the frequency, which one is the max will give you the best matching model? Is this clear now? That will give you the best match. When you are getting one increment on the vote, then for that for one increment the vote, it says how many times it hits or it matches. Now, for hashing what was the collision now in this case it is the voting or the hitting? Because of the hit, it will keep on increasing and finally, you will have a plot like this, where the X-axis indicates a different hash index and the Y-axis indicates the frequency or the number of votes. That is the motivation that we are going to discuss in the next lecture and these are some of the references I am using for this geometric hashing discussion. Thank you.