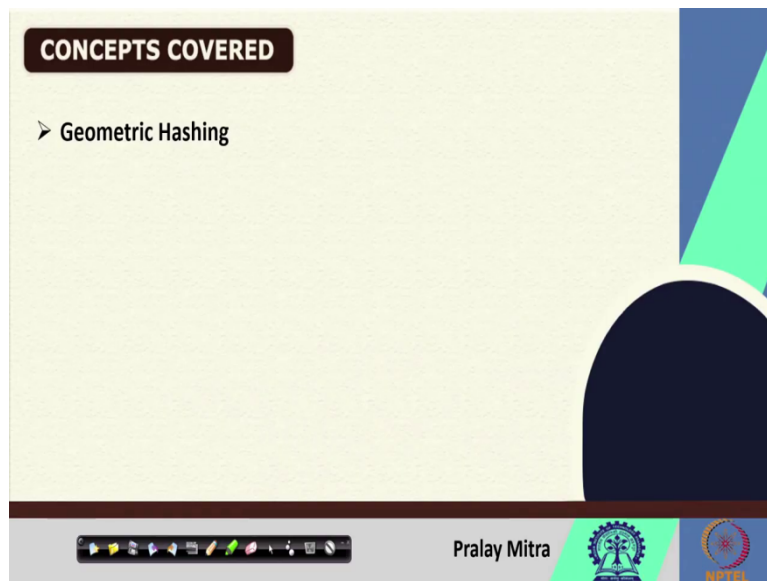


**Algorithms for Protein Modeling and Engineering**  
**Professor. Pralay Mitra**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**  
**Lecture No. 13**  
**Geometric Hashing (contd.)**

Welcome back. We are almost ready to introduce to you the exact algorithm for geometric hashing, which will be utilized for recognizing one particular orientation out of all the different possibilities from the protein molecules.

(Refer to Slide Time: 0:34)

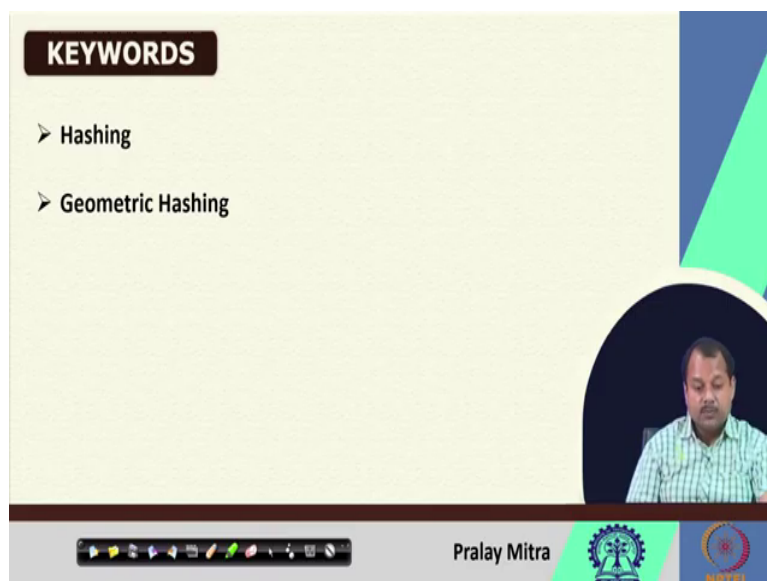


**CONCEPTS COVERED**

- Geometric Hashing

Pralay Mitra

The slide features a light green background with a dark blue and light green geometric design on the right side. A dark blue semi-circle is positioned at the bottom right. The footer includes a navigation bar with icons, the name 'Pralay Mitra', and logos for IIT Kharagpur and NPTEL.



**KEYWORDS**

- Hashing
- Geometric Hashing

Pralay Mitra


The slide features a light green background with a dark blue and light green geometric design on the right side. A dark blue semi-circle is positioned at the bottom right, containing a video feed of Professor Pralay Mitra. The footer includes a navigation bar with icons, the name 'Pralay Mitra', and logos for IIT Kharagpur and NPTEL.

## Geometric Hashing


**Algorithm 4:**  
**Input:** Two protein Molecules  
**Output:** A list of protein complex decoys (several orientations) formed out of the given protein molecules.

**Steps:**

1. Identify the important points from the input molecules to form the models.
2. Preprocess the larger model to populate hash bins.
3. Recognize the smaller model based on the voting at the hash bins.



Pralay Mitra




## Geometric Hashing


**Preprocessing phase**

For each model M do the following:

1. Extract the model's point features. Assume that n such features are found.
2. For each ordered pair (of three, non-collinear, points  $(p_1, p_2, p_3)$ ), or basis, of point features do the following:



Pralay Mitra



We shall be continuing the same concept and the keywords are also the same. The algorithm takes two protein molecules as input and outputs a list of complex decoys with several orientations formed out of the given protein molecules. Do you remember that the last week, it was a brute force method with time complexity  $O(n^6)$  to generate docking decoys?

We translate along the X-axis then we translate along the Y-axis, then we translate along the Z-axis, then we rotate about the X-axis, then we rotate about the Y-axis then we rotate about the Z-axis, and then one orientation we got. In this case, we will utilize that one to generate one particular orientation.

So, you can understand that not all the surface atoms will be considered, We will consider only a subset of the points which we call the critical points. Those critical points or the

important points are sufficient to represent the shape or the physicochemical property of a protein molecule on its surface and that is enough or sufficient for the protein molecule.

I mentioned that input is two protein molecules. Grossly the step will be to identify the important points from the input molecules (also called models), next pre-processing the larger model to populate hash bins. Recognized the smaller model based on the voting at the hash bins. So, those are the three different steps of this geometric hashing.

Now, we will go into detail about steps 2 and 3 because regarding step 1 we discussed it in detail including its implementational details. In the pre-processing phase, what we will do for each model  $M$  - is extract the model's point features. So, in this case, our emphasis is not only on the coordinate but also if other features attached to the atoms specifically the atom part of an amino acid or residue.

If that atom is having some important or specific feature as an atom or as a part of an amino acid or part of that protein molecule, you extract that information also. In this case, the programmer or the developer has the freedom to add his features. You can add that one so that we will go for the matching and then vote. Next, whether it is matched or not that decision we can take based upon these feature values.

Assume that  $n$  such features are found. Accordingly, the length of the fingerprint or the information which will be stored as the model information will also vary on how many features are there. Now, each ordered pair of three non-collinear points that you remember ( $P_1, P_2, P_3$ ) was considered arbitrarily. This can be the basis.

(Refer to Slide Time: 4:46)

**Geometric Hashing** *hash table*

Preprocessing phase (contd.)

(2a) Compute the coordinates  $(u, v)$  of the remaining features in the coordinate frame defined by the model basis  $(p_1, p_2, p_3)$ .

(2b) After proper quantization, use the tuple  $(u_q, v_q)$  as an index into a 2D hash table data structure and insert in the corresponding hash table bin the information  $(M, (p_1, p_2, p_3))$ , namely the model number and the basis tuple used to determine  $(u_q, v_q)$ .

Hash Function:  $h(Q(u), Q(v)) \rightarrow$

The slide includes a 4x4 grid representing a hash table and a small video inset of the presenter, Pralay Mitra, at the bottom right. Logos for IIT Bombay and NPTEL are also visible at the bottom.

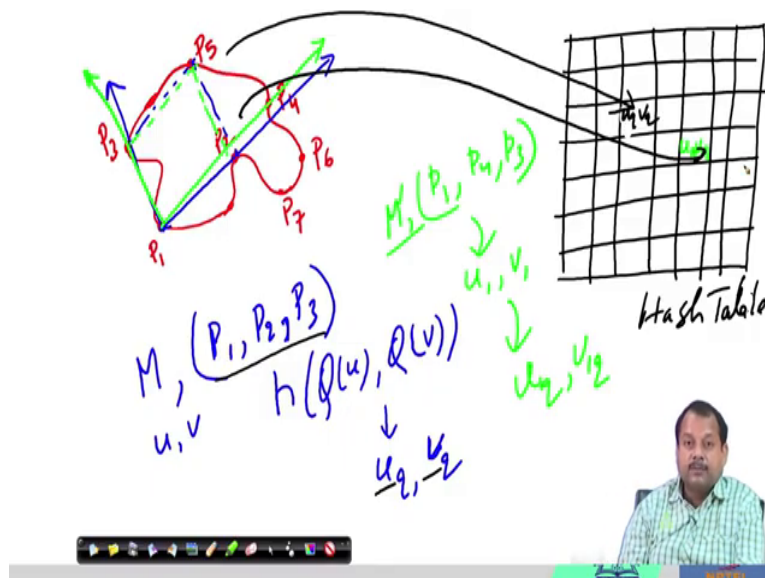
Compute the coordinate  $(u, v)$  of the remaining features in the coordinate frame defined by the model basis P1, P2, and P3. If you remember that diagram - compute the coordinate  $u, v$  of the remaining feature in the coordinate frame defined by the model basis P1, P2, and P3.

After proper quantization, use the tuple,  $(u_q, v_q)$  as an index into a 2-D hash table data structure and insert in the corresponding hash table information  $(M, P1, P2, P3)$ . P1, P2, and P3 are the basis coordinates based upon which you got this information, the model M will incorporate the information-like features that I have mentioned.

Based upon the number of features you are considering the size of the M as an array will change and also the information that you wish to store will change or will vary. Namely, the model number and the basis tuple used to determine  $u_q$  and  $v_q$  that you got from these  $u, v$ , and the hash function you determined using these.

This hash function will be mapped to this hash table. Pictorially it suggests that for each ordered pair of three non-collinear points (P1, P2, P3) or basis. Compute the coordinate  $(u, v)$  of the remaining features of the coordinate frame defined by the model basis pair P1, P2, P3 and after proper quantization using this hash function you go to that 2-D hash table.

(Refer to Slide Time: 7:11)



Let us assume that there is an object. This is my P1, this is my P2, this is my P3 and say this is my P4, this is my P5, this is my P6. Several other points are also there. I am not detailing that one. These are only the schematic diagram. Now, you declare one, or even before that one you select one basis coordinate or non-collinear triplet that I am assuming  $(P1, P2, P3)$ . If it is  $(P1, P2, P3)$ , through this I am passing one coordinate system.

There is a small change. Let me do that. The numbering will be this is P2 and this is P7. So, I got P1, P2, and P3. P1, P2, and P3 are my basis. For this basis, I can consider this point say P5. So, projection of P5 on this I will get. For this P5 model information, I will get from this projection, I will also get  $(u, v)$ .

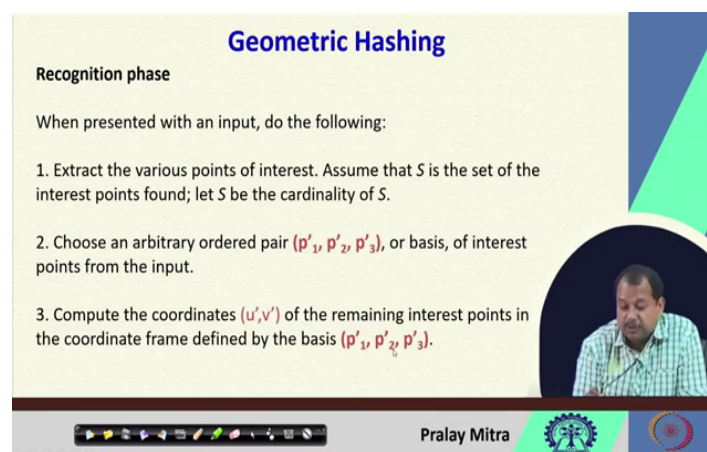
Using  $h(Q(u), Q(v))$ , I will have one index position say  $(u_q, v_q)$ . I am assuming the existence of one hash table. This is my hash table. This  $u_q$  and  $v_q$  will be in some position say here, say this is my  $u_q, v_q$  which means this one will go and will be mapped here. Instead of (P1, P2, P3), if I consider another basis point say (P1, P4, P3), I will have this feature say  $M'$  since the basis point has changed, so, the feature may also change at least the geometrical features which will be there. Under this situation, similar to  $(u, v)$ , I will get  $(u_1, v_1)$ . I can get this hash function. If I assume this is my  $(u_{1q}, v_{1q})$ , then this green will go and map here that way.

After selecting the number of important points, then you choose a list of triplets that are non-collinear, and for that, all the points I considered only P5. But when I am considering P1, P2, and P3 then except you want P1, P2, and P3 other points I can calculate and map here.

Now, this mapping will be done initially for one protein molecule and that is my pre-processing stage in this pre-processing stage it says that it is offline. So, during the say actual orientation calculation, I did not have to do that one. This is my pre-processing stage.

What is next? By this pre-processing stage, I populated these hash tables and in that hash table at each position or in each hash bin there is model information and the basis information. Basis information tells how I am getting that orientation and model information tells about the features which you will be matching during the recognition phase. So, it will be done during the pre-processing stage.

(Refer to Slide Time: 13:42)



**Geometric Hashing**

**Recognition phase**

When presented with an input, do the following:

1. Extract the various points of interest. Assume that  $S$  is the set of the interest points found; let  $|S|$  be the cardinality of  $S$ .
2. Choose an arbitrary ordered pair  $(p'_1, p'_2, p'_3)$ , or basis, of interest points from the input.
3. Compute the coordinates  $(u', v')$  of the remaining interest points in the coordinate frame defined by the basis  $(p'_1, p'_2, p'_3)$ .

Pralay Mitra

Now, in the recognition phase when presented with an input do the following - extract the various points of interest similar to the first point of the pre-processing stage again, I have to extract the important points. Then assume that  $S$  is the set of the interest points found and let  $|S|$  be of the cardinality of  $S$ .

Choose an arbitrarily ordered pair  $P1', P2', P3'$  or basis of interest points from you. Compute the coordinate  $u', v'$  similar to the previous one of the remaining interest points in the coordinate frame defined by the basis  $P1', P2', P3'$  same as the previous one.

(Refer to Slide Time: 14:25)

**Geometric Hashing**

Recognition phase (contd.)

4. Appropriately quantize each such coordinate and access the appropriate hash table bin for every entry (model,  $(p_1, p_2, p_3)$ ) found there, cast a vote for the model and the basis.

5. Histogram all hash table entries that received one or more votes during step 4. Proceed to determine those entries that received more than a certain number, or threshold, of votes: Each such entry corresponds to a potential match (hypothesis generation).

Hash Table

Threshold

Pralay Mitra

Next, appropriately quantize each such coordinate, and appropriately hash to the bin (same as the previous one) for every entry model  $(P1, P2, P3)$  found there. Cast a vote for the model and the basis. What is the difference? Up to this point, it was the same that the recognition phase and pre-processing phase is the same. But what is new in the recognition phase is that once you will get one hash bin location where it is supposed to be stored and is stored for the pre-processing stage for the recognition stage you cast a vote for the model and the basis means, it is matching with the initial protein molecule.

You are doing a pre-processing with one protein molecule and say recognition with another protein molecule. Now, for these protein molecules, one particular orientation is going to one hash bin, and one particular orientation along with the model and basis point is going to that same location which means they are matching. And when they are matching, I am casting one vote.

Plot a histogram of all table entries that received one or more votes. Proceed to determine those entries that received more than a certain number - threshold. You can apply some threshold based on your idea or votes each such entry corresponds to a potential match that is the hypothesis of the generation.

Each such entry corresponds to a potential match, potential match means that there is a probability that there is orientation. So, what is zero in the vote? Zero vote means this is one way to solve, this is another way to solve, look there is no matching. That is why there is no vote, but whenever there is some surface overlap matching in the feature and the orientation matches then there is a vote.

(Refer to Slide Time: 17:39)

**Geometric Hashing**

Recognition phase (contd.)

6. For each potential match discovered in step 5, recover the transformation  $T$  that results in the best least-squares match between all corresponding feature pairs.

7. Transform the features of the model according to the recovered transformation  $T$  and verify them against the input image features (verification step).

8. If the verification fails, go back to step 2 and repeat the procedure using a different image basis pair.

protein model

Pralay Mitra

For each potential match discovered in step 5, the previous step recovers the transformation  $T$  that results in the best least-square match between all corresponding feature pairs. Next, transform the features of the model according to the recovered transformation  $T$  and verify them against the input feature verification step. In this case, this image - my model remains because initially this geometry hashing was designed for recognizing the object in computer vision. As of now, we are discussing two-dimensional, but I believe that you can easily extend that to a three-dimensional protein model. Now, if the verification fails, go back to step two and repeat the procedure using a different image or basis pair.

To summarize, the algorithm says that first you have to identify the critical points for both the protein molecules, then you decide for which you are going to pre-process and for which you are going to recognize. Now for which you are going to pre-process, then corresponding to

each set of non-collinear triplets you compute the  $(u, v)$ , then using that  $u, v$  you generate the hash index/bin. You store the non-collinear triplets which we are calling as a basis or the ordered pair or the basis coordinate and the model feature information in that hash bin location, that is a pre-processing stage.

In the recognition stage, everything you have to do whatever you have done for the pre-processing stage. But instead of storing in the hash bin hash table, you go to the hash table using the same way you generated the hash table index, you go to that location and whenever you find that somebody is present there you compute your features vector and their feature vector you perform some computation and if you see that there is a match, then you give one vote one increment and of course, you need to store this information also so that in future if required you can generate that particular orientation.

That way once you will fill up one orientation corresponding to one triplet (non-collinear basis point) during the recognition phase, your voting is over. Then you pick the best vote or the vote or the cases whose votes are beyond the threshold value (better than zero). You pick those cases and go for the verification.

In the verification stage, you go for the reverse transformation or you recover the transformation and you generate and then you check whether it verifies or not. If yes, then find that is one of the possible cases. If not then you discard and move on to another one.

(Refer to Slide Time: 21:19)

**Comments on Geometric Hashing**

Complexity:

Preprocessing Step:  $O(Mm^3)$

Recognition Step:  $O(Mm^3)$

worst case:  $O(i^4Mm^3)$

(M: #models, m: #model points, i: #scene points)

The slide includes a video inset of the presenter, Pralay Mitra, and logos for IIT Bombay and NPTEL.



If we look at the complexity, then it is comparatively very fast. In the pre-processing stage,  $O(Mm^4)$  whereas, for the recognition stage the worst case is  $O(i^4Mm^4)$ , where  $M$  is the number of models,  $m$  is the model point.

The model point means the critical points that you have identified. Definitely what we have discussed based upon that one critical point is not going to be the same as the number of surface points. It is much less than that one. That way it is advantageous. Although at the complexity level, you are looking at  $Mm^4$ , it is not much since the number of model points is less very, very less compared to the total number of the surface points.

That total amount of computation time that will be required for the geometric hashing will be very less compared to the brute force technique or if I go for the fast Fourier transform-based technique, then also I use the same point.

(Refer to Slide Time: 22:42)

**Comments on Geometric Hashing**

- For the algorithm to be successful, it suffices to select an image basis triplet which belongs to some model.
- The goal of the voting scheme is to reduce the number of hypotheses that must be verified (filtering).
- In the case where model points are missing from the image (i.e., due to occlusions), recognition is still possible as long as there is a sufficient number of points hashing into the correct hash table bins.

Pralay Mitra

The slide features a blue and green geometric design on the right side. At the bottom, there is a video inset of Pralay Mitra, a navigation bar with various icons, and logos for IIT Bombay and NPTEL.

There are some disadvantages too. For the algorithm to be successful, it suffices to select an image basis triplet which belongs to some model otherwise it will be a problem. The goal of the voting scheme is to reduce the number of hypotheses that must be verified during the filtering stage. In the case where model points are missing from my image, it is still possible as long as there is a sufficient number of points in the correct hash table bins you can generate, that is good news. Remember this!

(Refer to Slide Time: 23:30)

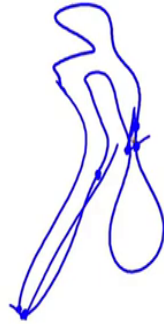
**Geometric Hashing – points to note**

- Looking for 4 point correspondences between the 3-D model and the 2-D image (3D hash table).
- Four non-coplanar points define a 3-D affine basis; the coordinates of any 3-D point can be computed in this coordinate frame.
- During recognition, we vote for all the bins lying on a given line in the 3D hash-table.
- Avoid unstable basis triplets
  - “Skinny” triangles lead to instabilities in the computation of the affine transform parameters.
  - Avoid “skinny” triangles using an “area” criterion.
- “Learn” good geometric hash functions

Pralay Mitra

Looking for 4-point correspondence between the three models and that 2-D image 3-D hash table. Four non-coplanar points define a 3-D affine basis - the coordinate of any 3-D point can be computed in this coordinate frame. During recognition, we vote for all the bins lying on a given line in the 3-D hash table. And you have to avoid unsuitable basis triplets. There may be some basis triplet for which a skinny triangle lead to instability. For example, if the situation is say some protein molecule, if you consider this one and say you are considering this as one say this is point this is one. You see that there are two frequent points for which the direction changes. But, if the situation is something like say here is one, here is one and say here is one. So, these three points, they are also called non-collinear. If this is the case, it forms a skinny triangle, then that may create some problems, avoid skinny triangle using an area criterion.

(Refer to Slide Time: 24:05)



Area criterion is one, and angle criteria are another. Three non-collinear points will form one triangle, then the internal angle will sum up to  $180^\circ$ . Now, you have to be careful that the distribution of the internal angles among the three angles actually will be more or less the same. It should not be that one angle is less than  $2^\circ$  or  $5^\circ$  and the rest is going to be say  $170$  or so - that may lead to some skinny cases.

You have learned good geometric hash functions. That is always a criterion specifically when you are interested to work with the hash function. If you are working with the hash function then selecting the correct hash function or good hash function must be your priority, otherwise a lot of problems may arise among a lot of problems the first problem is that collision. If a collision occurs, you have to resolve it by rehashing or double hashing, or chaining. So, those are extra complications on top of what we are going to do with this geometric hashing. This extra computation because of a poor hash function is not good and will create some additional computational overhead.

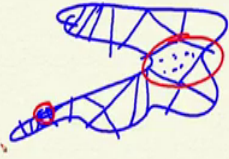

(Refer to Slide Time: 27:04)

## Geometric Hashing – points to note

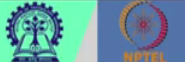
- “Learn” good geometric hash functions
  - Make the size of the bins proportional to the density of the data.
  - Learning is based on the “Kohonen” neural network.
  - Think of the grid as an “elastic” net that deforms based on the density of the data.

Data Distribution

Grid size

Pralay Mitra




Learn good geometric hash functions. Making the size of the bins proportional to the density of the data is a good thing, okay learning is based on the Kohonen Neural Network. That is another suggestion. Think of the grid as an elastic net that deforms based on the density of the data. Also, you think about the data distribution and the grid size. If you do not think about those cases, then some situations may appear where because of the distortion the distribution is not good.

And if the distribution is not good, then what will happen is that maybe here it will be sparse it will be very much congested in this case. In this case, it is very congested; here this is very, very sparse. So, those kinds of problems will appear. Better to avoid those kinds of problems.

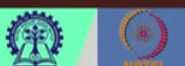
(Refer to Slide Time: 28:42)

## Geometric Hashing

- Pro:
  - Faster
- Con:
  - Storage requirement is very high and increases with the increase in object points.
  - Proper identification of object points are crucial for the success.



Pralay Mitra



**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.

The slide features a video inset of a man in a checkered shirt speaking. At the bottom, there is a toolbar with various icons and logos for IIT Bombay and NPTEL.

Also, the advantage of geometric hashing is that it is very fast as demonstrated by its computational complexity. If you look at the computational time, then consider the fact that the number of critical points is very very less compared to the number of surface points. The surface point by surface point as of now, we understand that the grid cell or maybe the atoms, but in the very next lecture, when I will discuss the molecular surface, then we will see that the surface atoms. The surface points are more compared to that one. On the other hand, what we are considering is very very less. The only disadvantage of this geometric hashing is the storage requirement. It is very high and increases with the increase in the object points.

I can give you one real-life example and I estimate that, for one protein molecule where there are about 330 amino acids and it's a homodimer which means two protein chains are of equal size, so 330 in one subunit or one protein molecule and 330 another protein molecule, one geometric hashing may take about say 5 to 6 GB of data.

If you do not have any expertise on how to use that GB of data, partly keeping in the say secondary storage and partly in the RAM, and to speed up the computation, if you wish to put in or pull everything from the secondary storage to the RAM during your recognition phase, then you need a very good Workstation or very large support for the memory.

Currently, with the advent of computer technologies, memory is not that much costly. And some of the desktops even have 4 GB or 6 GB, or 16 GB data. It is then not a problem. But otherwise, it's a problem. Proper identification of object points is crucial for success. If I miss some of the anchor points, then definitely, I am going to miss the correct orientation, because orientation is directly proportional to the computation which is based on the identification of

the critical points. That's it. Again I mentioned that mostly in this geometric hashing, I took the help of these three papers, mostly the work of Wolfson. Thank you very much.