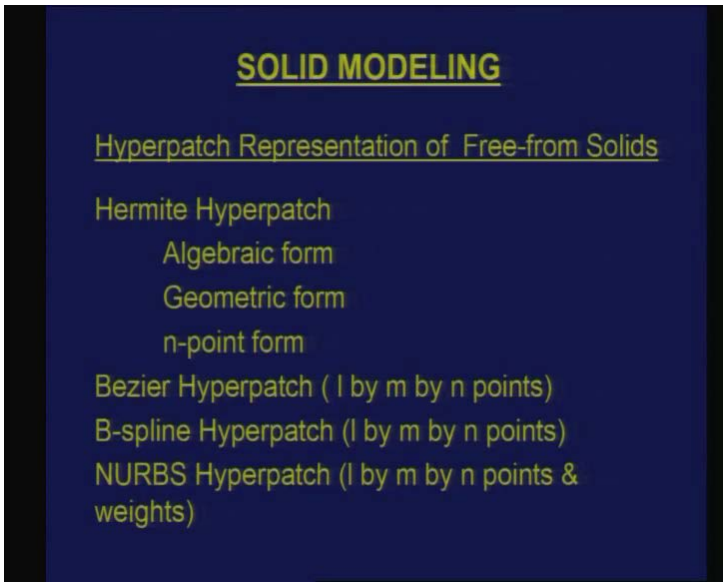
CAD / CAM
Dr. P. V. Madhusudhan Rao
Department of Mechanical Engineering
Indian Institute of Technology, Delhi
Lecture No. # 10
Solid Modeling

Starting from today, we will spend a few classes discussing about the subject of solid modeling. So far we have been discussing about parametric curves and surfaces. So now we will take up the solid modeling as a subject. There are many ways of modeling solids and many of these methods are very different from what we studied in curves and surfaces. So let's look at them briefly in today's class then we will take up more details about these models and methods in our subsequent classes. Now one of the logical extensions of modeling solids is basically extension of parametric curves and surfaces which we discussed to solids also.

(Refer Slide Time: 00:02:26 min)



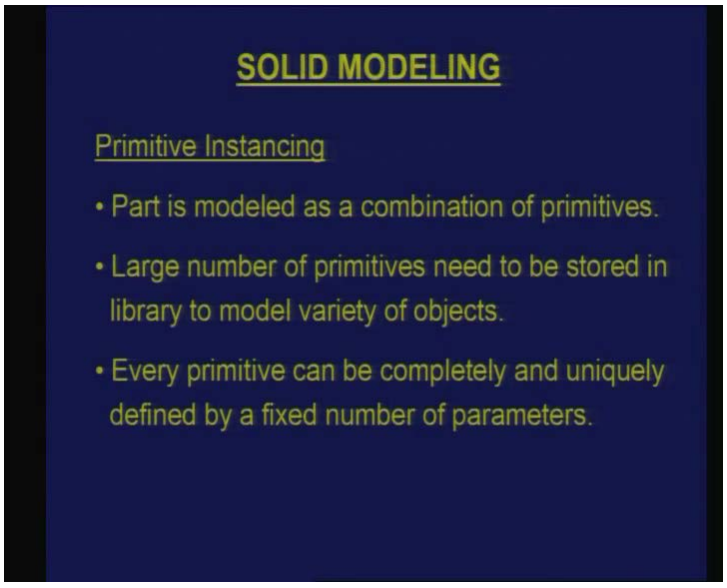
Just like we had a Hermite curve and Hermite patch, I may have a Hermite hyperpatch which is nothing but a piece of solid which is bounded by 6 surfaces. And you can have three different forms just like we had for curves and surfaces. I can have algebraic form in which if any coordinate let's say x y or z is a function of uvw and you will have all the terms like up to u cube, v cube and w cube as a part of this. So I can write down the algebraic equation and that may be one of the ways to do that or I can have geometric form where I can define let's say the input will be in the form of 8 corner points of a hyperpatch. Then first, second and other derivatives at those 8 corner points, if I provide this as a input I can also construct a geometric form of a hyperpatch. Then one can also define an end point form just like we have a parametric cubic curve which is defined by 4 points, interpolating 4 points or I have a bicubic patch, bicubic Hermite patch which is defined by interpolating 16 points in a space.

So I can have tricubic Hermite hyperpatch which is defined by 64 points that is 4 into 4 into 4 a matrix of points. So these are like one way of looking at solids. Then the definitions can be extended to Bezier hyperpatch, B-spline hyperpatch and also Nurbs hyperpatch is this thing. This is not a very convenient way to model solids because most of the solids which we come across particularly in engineering applications are not free formed solids. And so it's not a very convenient way to model the solids. Suppose if I want to model a simple cube this thing, a NURBS hyperpatch representation would be a very difficult way of looking at a very simple solid. But it has its own applications particularly in computer graphics and others, hyperpatch representation are used like one of the applications is in computer animation.

Suppose I have a solid and this solid undergoes deformation because of application of let's say forces or collisions. So the deformations can be which basically have a complex shapes can be represented using hyperpatch kind of representation. So it has applications but not widely used method of modeling solids. When it comes to solids, particularly for example most of the solids which we come across particularly in engineering applications are relatively simple. They are like either extruded shapes or shapes which can be very easily generated by parametric directions which are like line and circle. So which we usually call as a prismatic shapes, those are the more common forms. So we look at other ways of representing solids moving from a hyperpatch representation.

One of the first methods of modeling solids which came, which was introduced is called as a primitive instancing.

(Refer Slide Time: 00:06:14 min)

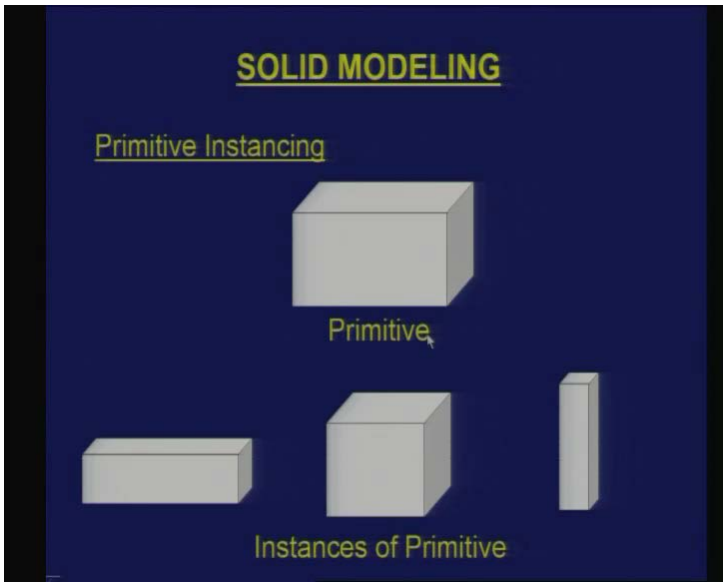


In fact when solid modeling was in the beginning particularly in 60's and 70's when this subject came up. Primitive instancing was one of the first proposed models to basically construct a solid. So in primitive instancing, what is done is a part is modeled as a combination of primitives like if I want to construct an object, it should be treated as a

combination. That means many primitives put together side by side or connected kind of a thing. And the kind of objects which can be modeled will be defined by how many primitives you have. It's like I can have gear as a primitive or I can have a spring as a primitive or suppose if I am constructing a mechanical object, I will directly take these primitives and define the parameters to specify primitive uniquely and place them in a space and build more complex objects.

So, one has to make basically a mental picture of given a particular object what primitives are needed and how to put them together in order to get more complex objects. And whenever like there is a object, one has to model a object where the given primitives are not sufficient. So, one has to introduce new primitives also as a part of the library for modeling the object or for future applications. And in order to define a primitive also like one has to give a certain minimum number of parameters by which a given primitive can be uniquely defined that was another thing. So, I will explain this using let's say an example a little later.

(Refer Slide Time: 00:08:09 min)

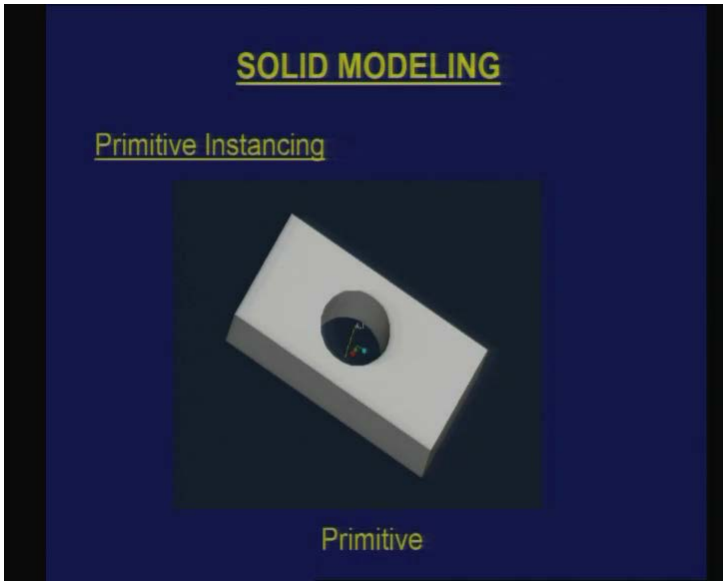


If take an example like I may have in my library a primitive which looks more like a cuboid. Now this is a primitive as far as size is concerned, you have a variations are allowed. As far as shape is concerned, it has a fixed shape. So this is primitive which is stored in library, using this is a primitive I can construct any of these things. They are all instances of the same primitive. Once I have a primitive like this, I need not have other primitives which are just to variation in terms of sizes like take an example of primitive like this which is a rectangular parallelepiped or cuboid, one can very easily define this primitive using three parameters.

Somebody can call it as length, breadth and depth or length, breadth and height or whatever it may be. So three variables are enough, by varying these three variables I get many instances of this particular primitive. So if any object which has this kind of

primitive which is need, so I will just vary these values and get any of the size variations as far as a primitive is concerned. But suppose if I want to construct let's say another object which instead of having let's say rectangular cross section, let's say I have a hexagonal cross section then that becomes a different primitive, I cannot model. So you need to have another primitive in your library in order to do something, so that is like a very you can say nontrivial or you can say or rather trivial way of modeling the object.

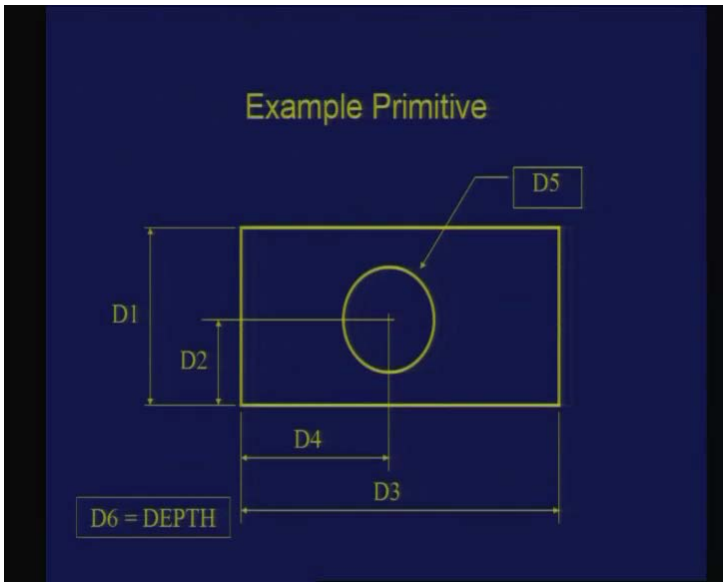
(Refer Slide Time: 00:10:01 min)



Like, here is just an example. I may also have a primitive like this. This is again sort of rectangular parallelepiped but you have a cylindrical through hole in it which is like one type of primitive. Now in order to define an object like this, I choose certain minimum number of parameters so that I should be able to get the instances of this primitive. So how many values are needed in order to basically store this kind of primitive uniquely, like for a parallelepiped we had three. Here those three values are anyhow they are needed. Apart from this you also have a cylindrical through hole, so you need at least three more values.

That is in order to define let's say center of this particular hole which may be in the center, it may be anywhere on the solid. And also what is a diameter of this particular hole. So 6 values, 6 parameters will completely define this particular object and by giving different values for these 6 parameters, I get many instances of this kind of primitive.

(Refer Slide Time: 00:11:20 min)



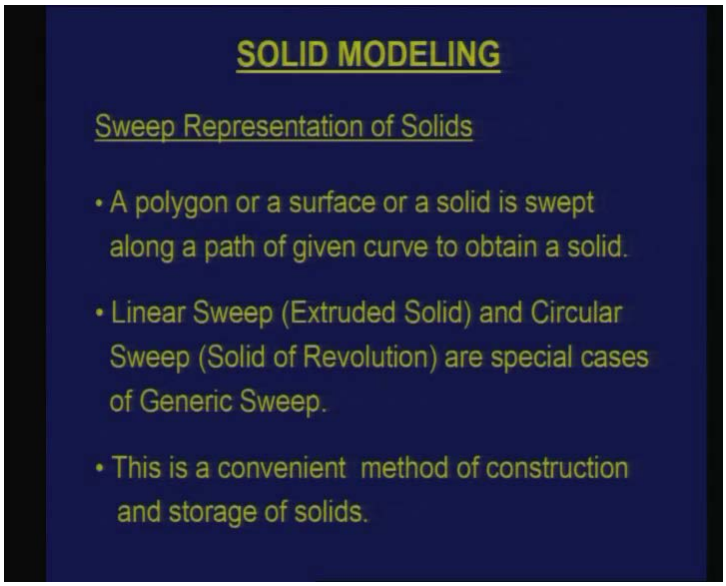
So this is shown here like this is an example primitive which we are looking at. So I have one of the dimensions let me call it as D1. Then D2 is a dimension where center is measured with reference to one of the surfaces then D3 is the other dimension, D4 is the center distance from another surface. Then D5 is a diameter of this particular hole and D6 is the depth of this particular thing, depth of hole as well as the object because it's a through hole. Suppose if I want to represent let's say a blind hole, I may need another parameter which becomes like what is the hole of, what is the depth of the cylindrical hole which I have.

So, since I am calling an object with a through hole as a primitive, this cannot be used to represent a blind hole so you need another primitive to do that. That may be that you call as another primitive which is stored in library. If I want an object like this through hole, I will pick this is a instance that is primitive otherwise I will choose the other one. And by giving these values I can this thing, one can also impose certain constraints in terms of parameters like you say that there are D1 to D6 are the values which one has to supply in order to build an object. but in order to supply input you also have certain constraints is that either I can have mathematical constraints like you can say that D2 should be less than D1 or D2 should be less than D_1 and related to D5.

So I can have a mathematical equations or what you call as a relations which will define a relation among these parameters. So, whenever you are giving an input, you can give the 6 inputs but the input should also satisfy those constraints which are imposed. If you, if it doesn't satisfy the constraints then instance or a primitive cannot be successfully modeled or it cannot be used or displayed as of part of this. So when you have a case like this, it is also called as a constraint based modeling. So it's a primitive instancing but you are apart from the variables which uniquely define, you are also imposing a constraint in order to do this kind of a thing.

So this is like you can say one of the simplest way to model a given solid is that have library of all possible geometries which you are likely to encounter in future and build more complex object as a combination of these things. We discussed about sweep, when we were discussing about parametric surfaces like sweeping is a concept which can be applied to curves, surfaces and solids. A point which is swept along a curve is let's say along a path gives me a curve. If I take a curve or let's say straight line and sweep it along a certain curve path or straight path then I get a surface. So the same thing can be extended to solids too. Let's say if I take a plane or let's say a surface patch and sweep it along certain trajectory then the volume which is swept by the surface is a solid. So that is one of the ways to represent a solid.

(Refer Slide Time: 00:15:08 min)



And an interesting thing is like the, you can say the object which you are sweeping need not be a surface it can be another solid too. I can take one solid and this solid is swept along certain path, so volume swept by the solid which you are sweeping is also another solid. Now this kind of concept is used extensively in many engineering applications, particularly in virtual simulations etc. For example let's say I want to depict motion of a tool or the volume swept by a tool, tool itself is a solid so tool is moving along certain path and straight line, circle, circular arc or some other trajectory. So the volume swept by the tool is also a solid or there are two objects which are moving in space and I would like to find out whether they collide or not.

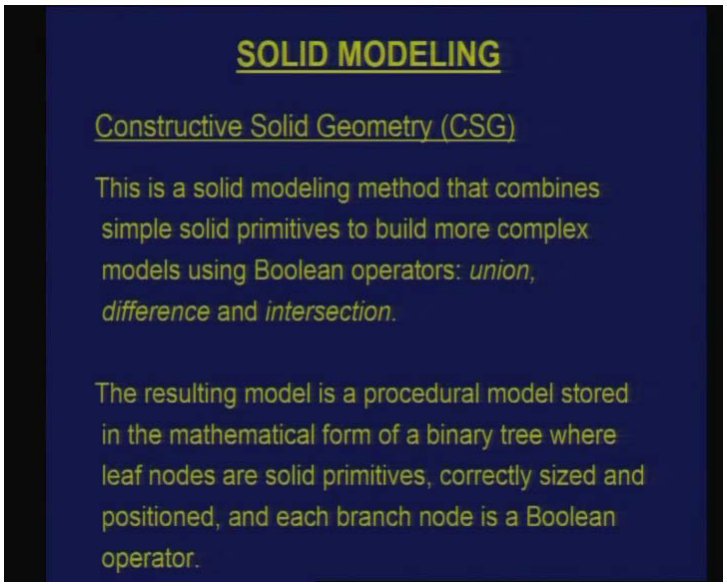
So what you do is you model the objects which are moving, you also model the path along which they are moving. So, since a solid which is moving along a certain path gives me another solid, I can construct the volume swept by the two solids and see whether the volume swept by the two solids intersect or not. So, that can be used to check whether in a situation like a motion modeling whether there are any collisions which are happening or not.

Of course, of among the swept representation of solids just like curves and surfaces, there are two of special interest one which we call as a linear sweep. If I take let's say polygon or a surface and sweep it along a straight line, the object which I get is an extruded object or I can have a circular sweep and where the object is swept along a path of a circle or a circular arc. So, I can have a solid of revolution which may be complete 360 degrees of revolution or it can be a partial too or special cases of a generic sweep. And this is also a convenient method for construction and storage of solids like when it comes to solid modeling, we will see that there are different types of representation.

Some representations are very convenient as far as constructing the object is concerned. Some may be convenient as far as storage of the object inside a computer is concerned. Third may be like you are also looking at how easy to calculate certain volumetric properties from this particular object. That means if I am doing some computational geometry related calculations, so there may be other methods which may be convenient. So it's also, one can say we will conclude later that there is no universal of a single representation which will give all conveniences. So, there are multiple ways of modeling solids and one may choose to use more than one representation for a given application depending on what you are trying to do with the solid model.

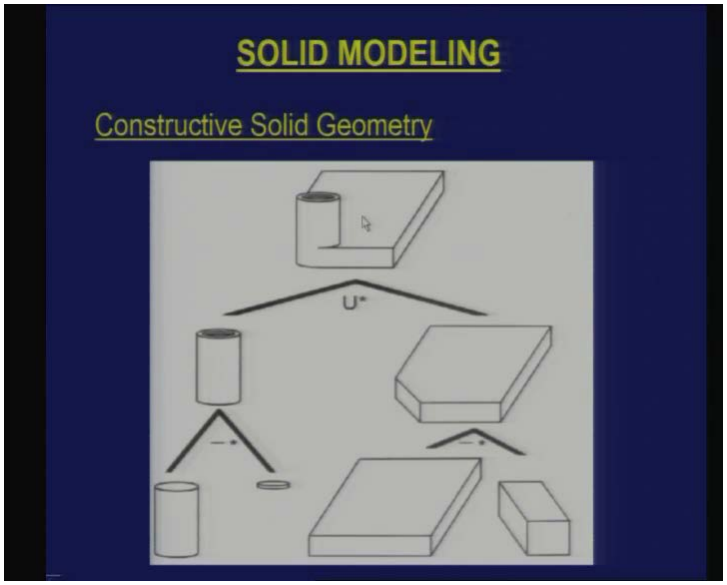
So, as far as sweeping is concerned it's a convenient method, I think you may have like used a packages like CAD CAM packages where you use extrusion and revolve kind of terms which is nothing but you are representing an object as a, using a construction method as a sweeping in order to build more complex objects from simple geometries. And it's also easier to store, that's because I have to store basically what I am trying to sweep a polygon or a surface or a solid and also the path along which it is swept. If I am able to store these two, I have stored the entire solid representation. From these two inputs, I can generate other representations or I can display the object, so I can do many things as far as sweeping is concerned. So sweeping is a logical extension what we discussed in for curves and surfaces.

(Refer Slide Time: 00:19:41 min)



One of the methods which was introduced particularly in 60's and 70's is called as constructive solid geometry. Now in constructive solid geometry, what is done is one uses certain known simple solids which are called as primitives like there are certain solid forms which we can say that known form of solids with which we are familiar like cylinder is a solid or a cone is a solid or cuboid is a solid, torus is a solid and sphere is a solid. So I can use a combination of these and combine them in terms of Boolean operations. The three Boolean operations which are used are union, difference and intersection Boolean.

(Refer Slide Time: 00:20:44 min)



These Boolean operations are actually mathematical operations or you can say geometric operations which perform, which are performed in order to build more complex objects. I will come to that using an example this thing. And the object which is let me first take the example then I will come back to this like here is an example of building a solid object using constructive solid geometry. If I look at this object, this is an object and it has a cylindrical protrusion and within this cylindrical protrusion, you also have a certain amount of material which is removed. It is to a certain depth which is also it's like a blind hole but a very shallow blind hole kind of a thing. Now in order to, this object may look little complex in order to build but this can be easily visualized as combination of may known primitives which are put together by Boolean operations. So like if I take these two in fact the entire object can be modeled using just two primitives, a cylinder and a cuboid or what you call as a rectangular parallelepiped.

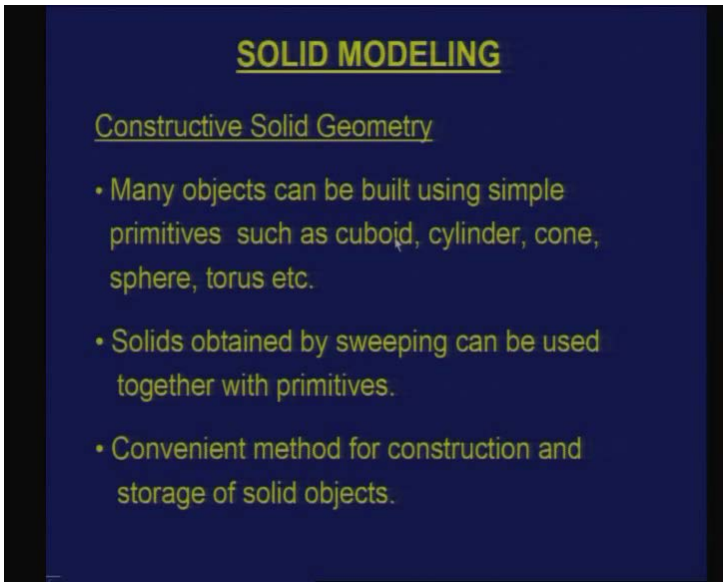
Sometimes this primitive is also called as a box, there are different names which are used. So I can use box and a cylinder to build more complex object. So first thing is here is a basically object where you have some kind of wedge shaped this thing which has been provided. So I can take two primitives, take a primitive which corresponds to a dimension of this, then another primitive and place the second primitive with reference to first, that is important. These are when you say a difference operation or a union operation, these are not scalar operations.

So this is basically an operation which you are on a set. Set consists of the points which are which a particular object occupies in a space. So I will take a primitive, I will take another primitive, construct these two primitives and then place this primitive with reference to this and do a difference operation. When I carry out a difference operation, you are trying to look at material which is there in this but not in this which gives me a shape like this. Once I have an object, I can model this separately the cylindrical. So I have two cylinders, again I have difference operation. This is a larger cylinder, this is a smaller cylinder which is in the form a cut out or a material which has been cut. So I will carry out these two difference operations. I have intermediate objects, now these are combined by a union operation. union is nothing but basically you are looking at the material which is there in both the primitives, this and this including the common volume which is there in both so I can carry out build an object like this starting from a simple primitives.

So if I look back, the resulting model in a CSG is procedural model where the representation is in the form of a tree because if we look at this is a tree like structure where when it comes to all the like branch nodes which are Boolean operators, whereas primitives are basically the leaf nodes like all the leaf nodes which you see are the primitives whereas the branch nodes are all the Boolean operation which are happening. So using these two combinations, I can build more complex object.

So in a method like constructive solid geometry, again user has to make a mental picture of given a object what kind of primitives and Boolean operations and transformations can give me the desired object and one has to put together and keep building more complex object starting from simple objects.

(Refer Slide Time: 00:24:43 min)



Now when it comes to a representation like this, in fact one the surveys which was conducted in a many years back says that many of the objects particularly which are used in mechanical engineering can be very easily build by just 5 primitives. What are those? A cuboid or a box shaped primitive cylinder, cone, sphere and torus. A combination of these 5 can give variety of objects. That is also has a close relation with in fact manufacturing processes. If one tries to analyze let's say many of the manufacturing machines which are used, you have a combination of motions which are linear and circular motions. And these combinations of motions usually yield the objects which are also like can be very easily built using these primitives.

In all these primitives also you have the parametric directions which are either line or a circle or a combination of these in building these objects. And one can also combine a like Boolean operations can be, need not be for only these primitives. I can also combine with the objects which are produced by sweeping like if I take this thing, instead of having one of the known primitives like one of those standard 5 or 6 primitives, I can take an object which is obtained by another method of solid construction like a sweeping, bring that and carry out the Boolean operation. So this is also, then it you are actually enhancing the capabilities of the solid modeling. That is true like when you take care of for example a package like CAD CAM package, so it's lot of sweeping operations together with Boolean operations, one tries to build more complex objects.

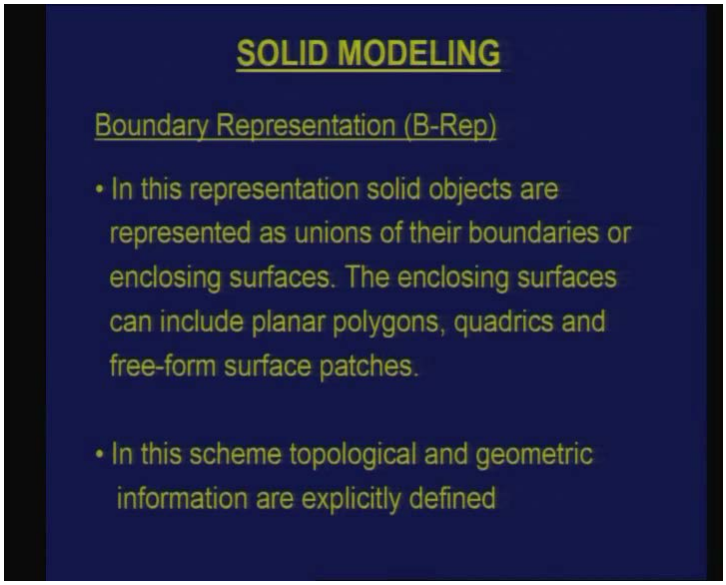
Then constructive solid geometry is also a convenient method for construction purpose like I can very easily a build an object. And so what is stored in a CSG is that in order to represent an object, you are actually storing this tree, all the leaf nodes and all the branch odes and the operations, so I will be storing a primitive, the parameters of this primitive and all the operations which are done on this. So this entire tree structure is what is stored. So representation becomes very compact even the complex objects can be very easily compactly stored. And it's a very convenient method for constructing an object

using a Boolean operation. But it may not be convenient for certain application. Suppose if I want to know what is a volume of this object, if I want to know that probably just from this tree like data structure, it may be difficult to evaluate.

So I have to do certain additional calculations and to do that or I want to know whether this object has any two parallel faces. this tree may not give me a direct information like this whether there are two this thing because ultimately one has to look at the final object and try to let's say evaluate from the final object, we will come back to that. You may have to convert into probably other representations and then I can do, I can know whether information like whether the object has two parallel faces or kind of thing.

So there are some conveniences in terms of construction and storage but there are also inconveniences as far as computational geometry calculations are concerned like suppose if I want to know, if I want to display this particular object I would like to remove the hidden lines. So it's not a very convenient method to do that because I need to apply special algorithms in order to do that, than more simpler algorithms which are available.

(Refer Slide Time: 00:29:11 min)



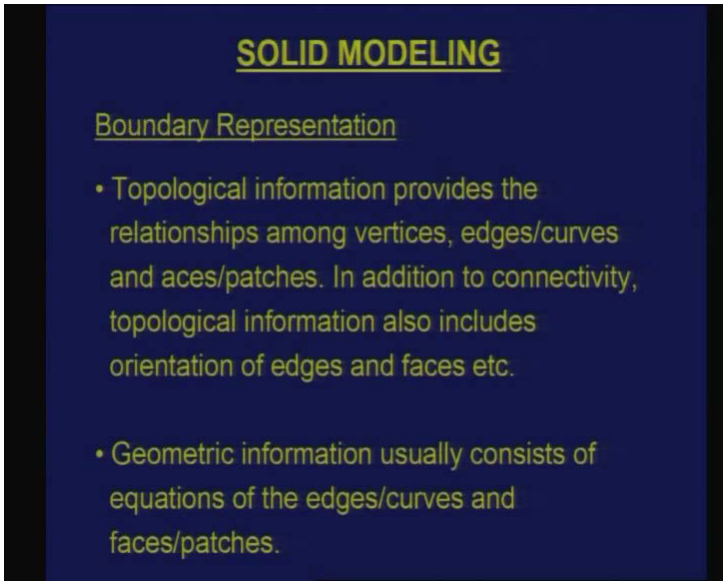
Now another very popular representation for modeling solids is what you call as a boundary representation. It's more commonly known as B-Rep. Now what is done in a representation like B-Rep is the object is basically represented by the boundaries or the surfaces which are enclosing the boundary of a solid object. So, we are not representing the complete interior points explicitly. You say that here are the boundaries. Boundaries has two sides, there is one on the material side and other is a nonmaterial side. So a volume which is enclosed by all the bounding surfaces is nothing but a solid. So if I do that, that is this type of representation is usually called as a boundary representation.

So the enclosing surfaces can be polygons, if there are only polygons all the surfaces which enclose a solid or planar polygons then it's called as a polyhedral, the object can

be called as a polyhedron or it can be a curved surfaces like I can use quadric surfaces like it can be surface like spherical surface or it could be cylindrical surface, one of the surfaces or I can also use free form surface patches like I can say that here is a solid which is enclosed by 6 Hermite patches.

So there are 6 Hermite patches which completely enclose an object which becomes a boundary of the object. So that's also if i represent a solid using this, this is also one of the ways and it's called as a, it becomes a part of the B-Ref. but basic difference if you look from other methods which are there particularly constructive solid geometry is that every solid information has two things. One which you call as a topology and another is geometry. Here these are these two are separately defined. These two what we call as a topology of an object and the geometry of the objects are defined separately. So what does one mean by this?

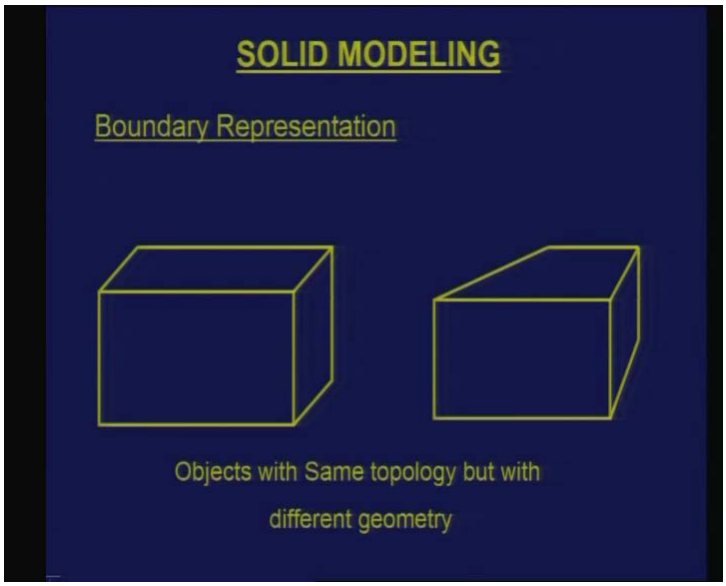
(Refer Slide Time: 00:31:33 min)



Probably this requires little more explanation like basically the topology of a solid consists of basic, the relationship which are there in terms of vertices, edges, curves, faces and patches. So how a particular, one particular face or patch is related with another patch whether it shares a common edge or common curve is a part of this. And this complete connectivity information is separately stored, apart from the geometric information. Suppose I say that there is a curve one, curve one and curve let's say I have a surface one and surface two are having a common curve which is called as a curve one. So that's my topological information. But what exactly is curve one, what is its equation or what are the data which is needed to generate this curve is becomes a part of geometry.

So these two are separated and stored separately in a boundary representation like when you say in a CSG if I say that I have a primitive, primitive has both topology and geometry which are mixed up. They are not separated. I will give this using let's say an example.

(Refer Slide Time: 00:32:41 min)



Take for example I have two objects, here are the two objects solid objects and both the objects have 6 faces and they have 8 vertices and they have 12 edges. Now if I number them let's say number all the 8 vertices, 12 edges and 6 faces and also try to represent let's say how is the connectivity information. Both the objects will have the same topology. As far as connectivity is concerned like I can say that this surface, these two faces share a common edge so same is true with these two faces which share a common edge. So when I store a topology for this, it is same there is no difference. But what makes these two objects different is geometry. So what is geometry in this case or the xyz coordinates of the 8 vertices. Basically the xyz coordinates of the 8 vertices which I can solve.

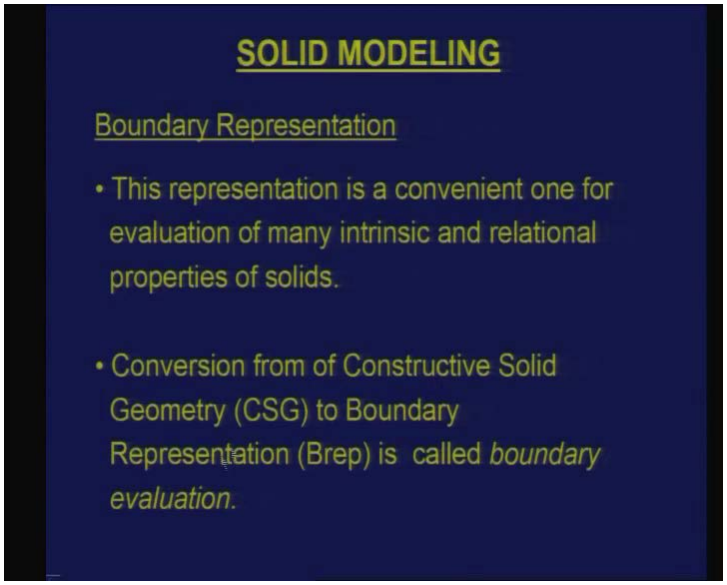
So by changing the xyz in certain fashion, I get different shape and size objects. Shape also is changing, size is also changing because earlier it was a cuboid now it has become some other shape. But what is common between the two is the topology, how the vertices etc is related. Now in order to store an object what I will do is I will suppose if I want to store this information in a file, I will say that this object has 8 vertices which I call as V_1 V_2 V_3 V_4 V_8. Then this has 12 edges which I call as E_1 E_2 E12 then this has 6 faces I call it as F_1 F_2 all the way to F_6. Then I take let's say a face one. Face one is made up of 4 edges let's say E_1 E_2 E_3 E_4. Each edge, one of the edges made up of two vertices let me call as V_1 and V_2. So I will store this complete information in one place.

Somewhere in the end or somewhere at other place, I will store the coordinates of this, the 8 coordinates of the points which give me an object like this. So this type of representation gives you a separation of topology and geometry is one of the probably most widely used representation for solids. If you say what is probably the most we can say widely used or most useful representation, every representation is useful at some stage or the other but when it comes to large number of applications, boundary representation is used.

But it is not a convenient method to build an object. Imagine if you have to build an object like a box like object like cuboid, I don't want to give the complete information that it has 8 vertices, it has 12 edges, it has 6 faces and this is the relation among this thing and lastly give a coordinates. So that may be, that kind of input in order to build an object like a cuboid is not a very convenient. But it's a convenient way to store it like I may use a primitive called as a cube from a library but I may convert into a boundary representation and store it as a part of this for certain applications like we were looking at some examples suppose I want to check whether given object, a complex object has two parallel faces or not.

I can easily find out this, so it basically requires looking at the topology and geometric information. since you have also the faces which are explicitly stored, I can compare the faces and I can say here is an object like let's say if I take an object like this yeah, there are three cases of parallel faces here is an object where there are one case of two parallel faces. So, one can easily evaluate from the geometry and topology information. Same thing from other representations may be more difficult to do.

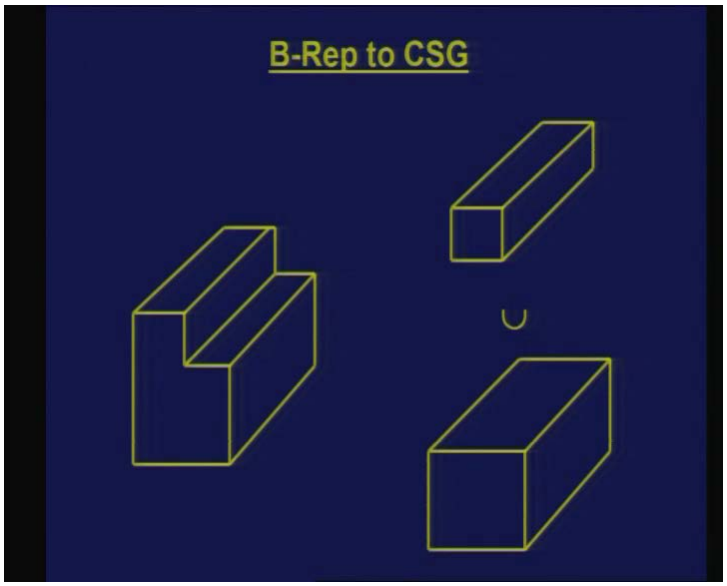
(Refer Slide Time: 00:37:28 min)



And as I said, suppose if I want to build an object constructive solid geometry may be a convenient method but for certain applications, boundary representation may be better so I should also have a method of converting from one to another. So I build using a constructive solid geometry and convert into a boundary representation. So, this method is called as a boundary evaluation. So boundary evaluation is a very common feature in most of the CAD CAM applications. most of the packages give you an option where I build using simple sweep representations, constructive solid geometry etc and then when I want to do certain operations, I also have a boundary representation of that which is, which can be obtained using a boundary evaluation. So this is again a mathematical geometric operation of converting one representation to another which is called as a boundary evaluation.

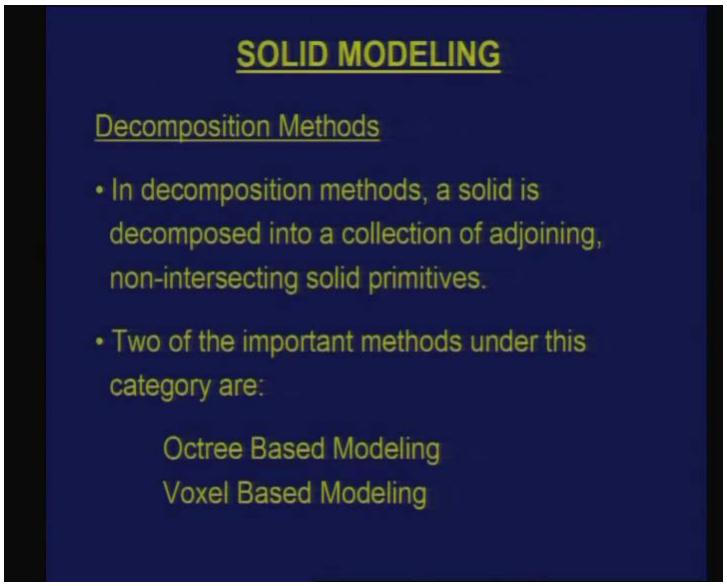
Usually the most common thing is to convert from let's say CSG to a boundary representation. Reverse is not a very common thing like given a boundary representation converting into a constructive solid geometry is not needed in many cases and neither it is a unique operation. This cannot be done like CSG to boundary representation is a unique whereas boundary representation to constructive solid geometry is not unique.

(Refer Slide Time: 00:39:03 min)



If I just take an example, let's say I have an object like this which is shown on the left. Now what would be a constructive solid geometry representation of this? Somebody may say this is union of two cuboids, one which is corresponds to this and another and i put them together to get an object like this. Somebody may have a different way of looking. They say no this is a difference operation; I take long a larger cuboid and then subtract a material which is corresponding to this and this. So the boundary representation of this object is same, it has a same number of faces, vertices, edges, geometry is also same. Coordinates have this thing but I have two different CSG representations for this. So since B-Ref to CSG is neither needed nor it is unique, it is not a usually people don't go from B-Ref to CSG whereas CSG to B-Ref is very common thing because that is required in many of the CAD CAM applications which is usually done.

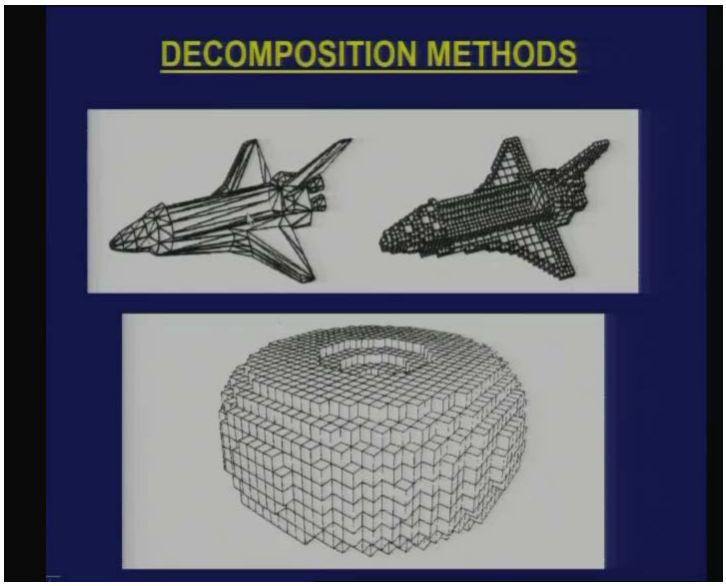
(Refer Slide Time: 00:40:14 min)



Then there are methods which are called as decomposition methods like in a typical CSG type of operation, what is done is you combine primitives but using a Boolean operations, like when I put together two primitives for joining or let's say union or difference or intersection, they may be having certain common volume. Suppose if I am carrying out let's say a union operation, this common volume comes only once because since the volume is common to both the objects I don't count it twice. But I can also build an object where I can put together the primitives where they are nonintersecting, they are not in the form an intersecting method like this kind of methods are usually called as a decomposition methods.

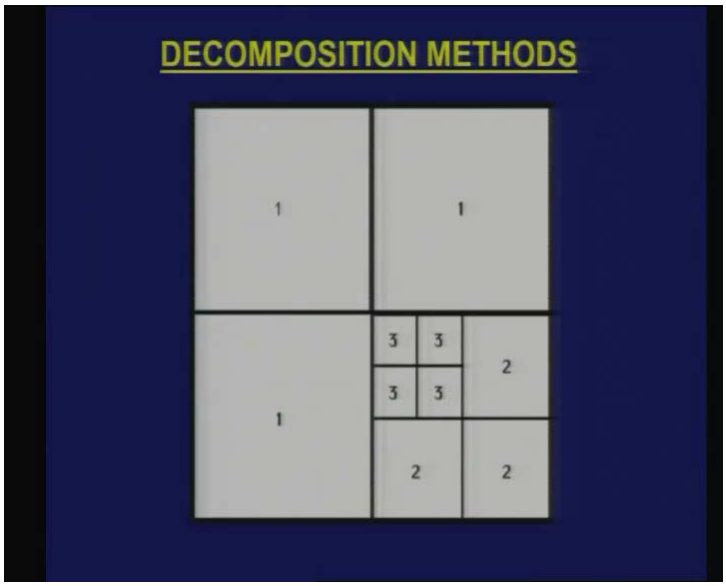
So what is done in decomposition method is basically you have a nonintersecting, solid primitives are put together in order to do that. There are many variations of decomposition methods but the two most popular representations are for solids is octree based modeling and voxel based modeling. Let's look at what are these representations because there are certain operations for which decomposition methods have an advantage over all the other methods which we have discussed so far.

(Refer Slide Time: 00:42:04 min)



We look at those also like what is done is here is an object. Now what you saying on the left is basically a boundary representation. In boundary representation what you are, what you do is that here an object which is represented by triangles. Now when an object is represented only using triangles, it is still a B-Ref. only thing is it's a very special case of B-Ref. and here is an object which is modeled using what you call as an octree based modeling. We will come back to that what is an octree based modeling.

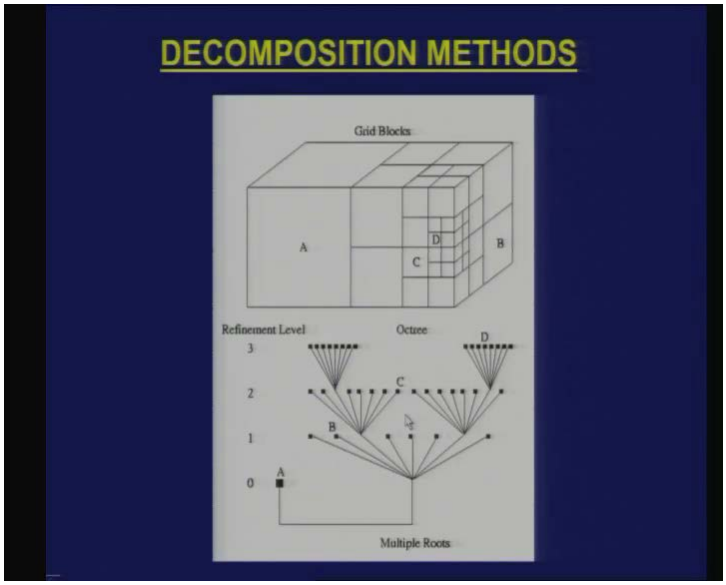
(Refer Slide Time: 00:42:40 min)



In an octree based modeling like what is shown here is basically a two dimensional version of an octree which is called as quad tree. In quad tree what is done is the object

which I am trying to represent is enclosed in a square and the square is divided into four other squares, a half the size of this. Now if the material lies then you look for the object which you are trying to represent is a part of any of these squares. Suppose if there is no material where suppose there is no material in one of the squares then I don't subdivide this like there is no material here but there is some material here, so I am going to subdivide this portion further. And I subdivide and I see that there is no material here, so I can go back and do this. So this is just a representation, this is not an object which is shown.

(Refer Slide Time: 00:43:41 min)

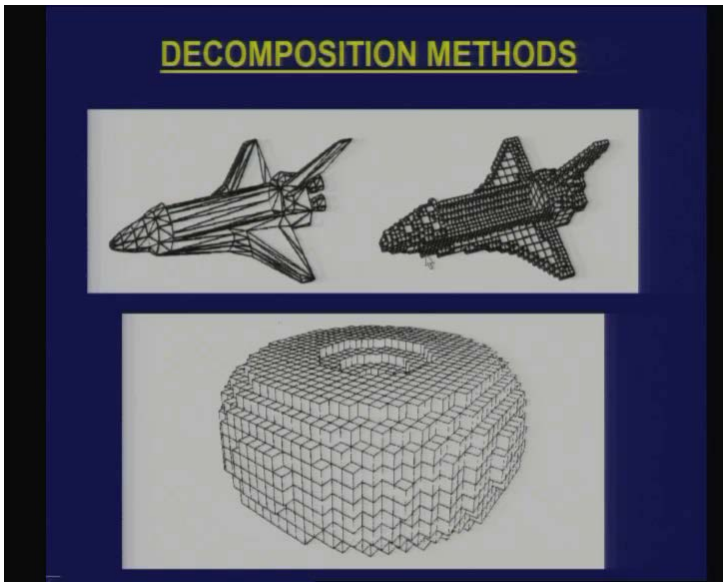


If I look let's say an octal representation here, I think this gives somewhat a structure for representing an octree like just like quad tree in which I take a square and divided into four squares. I can take a cube in an octree and divide it into 8 cubes of where the size or the dimension is half that of the original cubes. now the object which you are trying, suppose I am trying to model an object, first I enclose it in a cube and divide the cube into 8 cubes and see whether any of these eight cubes as the material which I am trying to model or not. If it has a material, suppose if there is no material I am not going to further subdivide because there is no material or not.

Suppose if it has a material, if it is completely filled with material, the entire cube is completely filled with the object which I am still, and I don't subdivide. I will say that there is a cube which is completely filled but if it is partially filled, I will further subdivide it into smaller cubes. so when do you go for a subdivision of a cube is like first given a cube I divide into 8, I will check there are three cases. A cube has completely filled with the material, no further subdivision. A cube is completely empty there is no material, no further subdivision but it is partially filled. A part of the cube is filled with the material which I am trying to represent then I will do, I go for a further subdivision of that. So you keep doing this to a certain depth which is called as an octree depth. After certain let's say, after going to a certain depth levels you get an approximate

representation of the object which you are trying to model. You are not looking at the exact representation. Suppose if I have an object which is free form surfaces, free form curves, it cannot be represented as a combination of planar surfaces so you are looking for an approximate method.

(Refer Slide Time: 00:45:53 min)



So if I look at this, so this is an octree representation of the same object. What you are seeing here are all basically a cubes of different size which are put together in order to model this object. So going from here to here like you know modeling an object from let's say a boundary representation to converting into an octree model is called as an octree encoding. This is called as an octree encoding. And as we looked at there is another version of a decomposition method which is called as a voxel-based model.

Voxel-based modeling has similarities with voxel is nothing but it's basically a three dimensional, you can say element in the form of a cube. For example you are familiar with a pixel. If I want to represent an image, I can visualize an image as a combination of pixels having different colors and different intensities which gives me a look of a two dimensional or three dimensional object. So I can also have a volumetric element which is called as a voxel just like in octrees, what is done is in voxel you have all the elements all the squares which are used, we will have the same size and shape, there is no variation or there is no subdivision. In a typical this thing what you do is like I have a square, if I want to do a pixel representation what I will do is I will take number of rows and columns.

So I have a two dimensional matrix of pixels and by saying whether a pixel is a part of the object or not, I am able to represent an object. Similarly I take a cube, divide it into smaller cubes like suppose if I have a cube of something like 10 by 10 by 10 size, if I take let's say my voxel element as 1 by 1 by 1. So I have something like a 1000 cubes, the cube can be divided into 10 into 10 into 10, a 1000 cubes. Now I will see the object

which is being represented like I take each of the cubes and see whether there is a material in this cube or not. If there is a material I can say it's a binary one, if there is no material I can say that this is a binary zero.

So simply you are going for a representation which is purely a binary representation. It has some similarities with an octree method and also there are certain advantages and disadvantages which you can clearly see. First thing is in octree you are only decomposing further in terms of squares or cubes, only if there is a material. So, that means the amount of cells which you end up in an octree would be less compared to a voxel-based.

Suppose if I take, I have a size of an object is let's say 100 by 100 by 100 and if I am taking let's say a voxel size which is point one, so you have 10 to the power of 9 voxels that is 1000 into 1000 into 1000 voxel information which you have to store. So the number of cells increases considerably whereas in the case of octrees you may not have that many cells because there may be many cells which you are not further decomposing in the intermediate stage. So that is an advantage but like suppose but there are some calculation conveniences for example I have converted let's say a solid, given solid into octree as well as voxel-based methods. Suppose, I want to quickly know what is the approximate volume of this object.

How will I do it in let's say an octree based method is you first see what are all the cells which are there, cells may be of different sizes. You say what are the sizes of, suppose if I take for example if I am calculating area of this thing, I can look at different cells which have different areas, count them, add them and then you get it. If suppose if I have a voxel-based model, all that which you need to see is that what are the cells, add all the cells and just multiply by the volume of a cell. So, little more convenient in terms of this. So there are situations where voxel-based decomposition methods, both octree and voxel have many conveniences in terms of other methods like one very like I gave an example of volume calculation.

So if I want to know quickly, suppose I have a very complex object which has a free form solid. But I have an octree or voxel representation of that then calculation of the volume becomes just simple multiplications and additions, there is nothing more. In fact it can be purely additions or purely you can say simple multiplications and additions. Whereas if suppose if the object is free form, if I want to do it by analytical or numerical methods, so I have to basically go for a numerical integration use a numerical method so that may be a little more cumbersome process in some cases, integral cannot be evaluated analytically. So we have to go for numerically and numerical methods also have problems in some situations, you have to look at all those things whereas it's very convenient in the case of this.

Another area where octrees have large you can applications is in terms of intersection checking. Let's say I have two very complex solids and I would like to know whether these solids are intersecting or not. Now suppose, if I have let's say a voxel-based representation of these two objects, how do we check whether they are intersecting or

not. You are just, you are interested in like knowing what is the whether they are intersecting or not, yes or no. So suppose if I have a voxel representation, when can you say that the objects are intersecting. If there is even a single voxel which is common to both the objects there is an intersection. So you just have to look at voxels in both the cases and see whether there is any common voxel for this so that is as simple as that or if it is octree based representation then you are checking basically an intersection of the cubes of one octal representation with the cubes of another one. So intersection of cubes is much more convenient like checking of intersection of let's say two cubes in a space is nothing but using only a relational operators like less than or greater than, it doesn't require any mathematical operations.

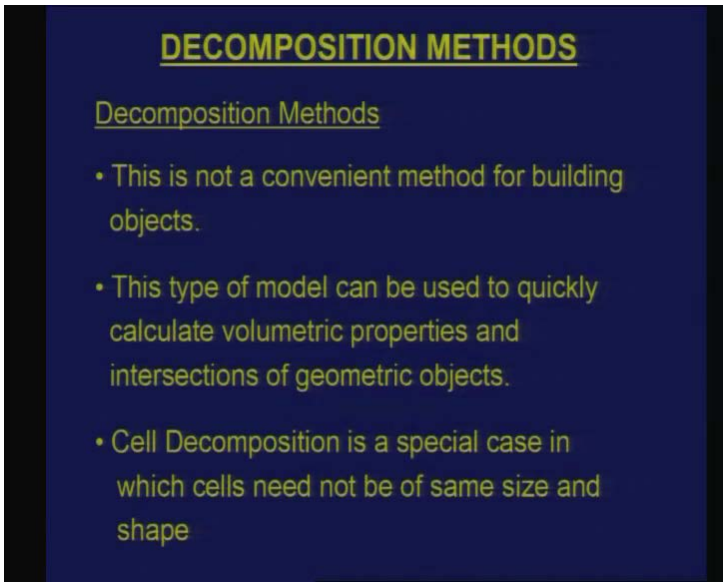
So computationally it becomes much more efficient to know whether there are any intersections between solids are also collisions. Collisions come when object is moving in a space relative to the other. So I can also, I know the sweep representation of a solid I can convert into octree or voxel and quickly check whether there is any collisions which are happening or not like in many robotics and machine tool simulations, you need to check for collisions. So one can build volumetric simulation based on octrees to quickly calculate this kind of situations.

So but imagine somebody has to build an object using a octree or voxel based method. Given an object and here is my input to build the object that is not a very convenient. It's not a convenient method of constructing an object, it may be a convenient in terms of storage or particularly if I have to do a certain computational geometry calculations which may be like if I want to know what is a centroid of an object, I can very easily suppose approximate centroid of an object again like if I use a free form representation, I have to do lot of mathematical calculations or numerical methods.

Here I can just use simple cube and other representation, other information to calculate approximate centroid of the object. So in terms of computational efficiency and they don't give you exact solution because in most of the cases, the octree or voxel based models are only an approximate representation of that. How approximate they are depends on how many cells one is going to choose or what level of subdivision which one chooses in the form of octrees or the voxel size in the case of a voxel based model.

So this is basically a voxel based, so here you have the object. Object is divided into 8, in fact what is shown here is like all the cells are shown as a black because it basically says that there is a material inside. But usually if there is no material I am going to represent it as a hollow symbol. So that I can represent it as a complete tree like structure, binary tree like structure to model a complex object.

(Refer Slide Time: 00:56:18 min)



So if we look decomposition methods, this is a convenient, this is not a convenient method for building objects but this is good for like calculation of volumetric properties and intersection of geometric objects. One can also think of like we looked at two decomposition methods octree and voxel based models. One can also think of there are methods which are called as cell decomposition methods where you are dividing an object in the form of cells like you say that all the cells are made of parallelepiped, a box like objects but they need not be of same shape and size. That is also a very common method for certain applications.

So cell decomposition is a special case of decomposition methods in which the cells which are used to model an object need not have the same shape and size as it is there in the octrees. Octrees you have the same shape, size can vary. In voxel you have same shape, same size. Here you are allowed to have both different shape and different size also as far as decomposition. So this is like if somebody has to know what solid modeling is in brief is this.

What we will do is we may have to look at some of these methods more closely and because when we take up any of these representation for subsequent applications, that would be useful like methods like boundary evaluation and etc. We will do it little more in detail in our subsequent classes. And after that then we will take up some CAD CAM applications based on the representations which we have studied for curve surfaces and solids.

So I will just stop it here, if you have any questions please clarify now itself. Student: So in the two types of decomposition methods that you have that is why octree encoding as is voxels based modeling which is the most commercially used option. So your question is which is more popular, is the octree based methods are popular or voxels based methods are popular. As such both are equally popular like for example in many of the

volume intersection methods etc people use octree based methods. Voxel based methods basically one of the problems is the storage explores, as you decrease the size of the voxel which is used. Voxel based methods are very popular for mesh generation in CFDs.

Suppose I have a computational domain and I would like to basically like in CFD, you calculate what is called as area porosities and volume porosities in order to carry out the CFD analysis. So in those situations like I can use voxel based models, octrees have also been used so that is all. Any other points? Student: in voxel based, voxel based modeling so voxel is filled partially. No, in voxel based modeling what you do is suppose you just check the center of the voxel. If center of the voxel is inside the object which you are trying to model that voxel is considered to be completely full. So this is an approximate representation.

So the resolution will tell you whether if the center of the voxel is outside of the object, you say that is an empty. So all that which you need to check in order to convert into voxel based method is first I have an object then I have imposed let's say voxel grid and the object and then take each of the voxel points and check whether it is inside the object or not. If it is inside the object it's filled, if it is outside the object it is empty, all that is binary. I think we will stop it here.