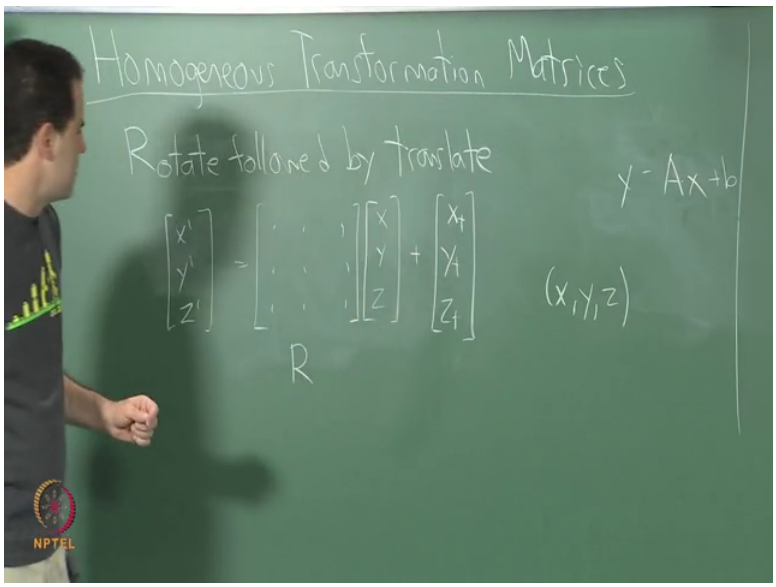
Virtual Reality
Prof. Steve Lavalle
Department of Applied Mathematics
Indian Institute of Technology, madras

Lecture – 5-1
Geometry of Virtual Worlds (homogenous transformation)

(Refer Slide Time: 00:15)



All right, let us do homogeneous transformation matrices, one of the reasons why I want to jump to this now is I want to combine rotation and translation, because I need to use both of those operations to be able to place a rigid body anywhere I like in the world because remember these are the movable parts of our model the moveable models. So, one way to achieve that is to do rotate rotation followed by translates.

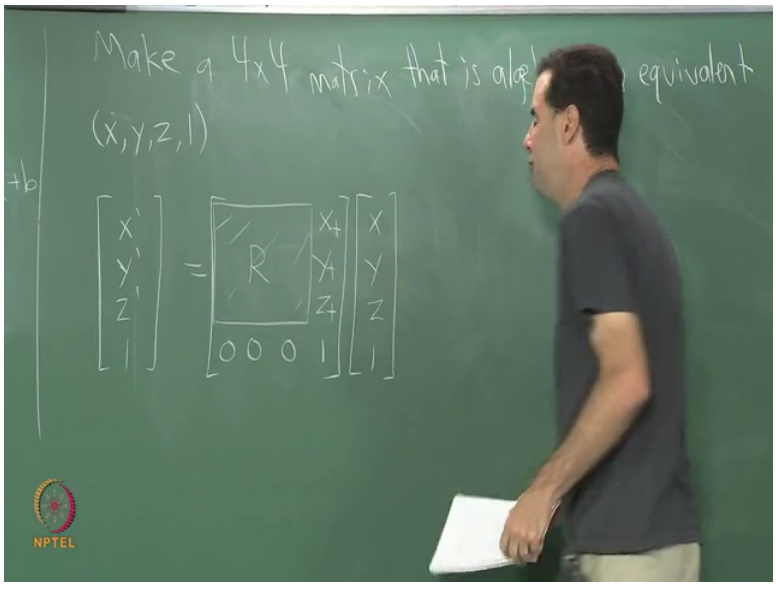
So, in other words, I get my transform point x prime, y prime, z prime by multiplying out this rotation matrix R which I will just again not write all the entries involved just put points here. So, that is a valid rotation matrix, and then xyz is my 3 D point, where I would like to rotate we agree that that is just simple rotation that we have been doing using the matrix, and now I add to it some translation I do not want to do, I was using this notation last time he was x t, y t and z t for translation. So, if I go ahead and do this, I should be able to generate any position and orientation for the rigid body, how many degrees of freedom do I have here.

Student: 6.

6, correct. So, I have 3 different parameters here I can apply for translation, and then I have 3 independent parameters I can apply in here after we satisfy all those constraints I gave last time for a rotation right. So, that gives a full 6 degrees of freedom, again we could do this part of the computation we are using quaternions, but I am doing all matrices today at this point because of some of the particular transformations I will beginning into. There are interesting questions and about when to convert the quaternions and why at different various times.

So, one thing to note is that the rotation part is a standard linear transformation, but this addition part comes outside of that because I am not multiplying a matrix here in the front right. So, I am not multiplying and adding matrices, I somehow I do not have a simple equation of the form y equals A x, I have added some extra term to it. So, one simple trick that people like to do is just extend the matrix by one dimension, add another row add another column and then you can get this expression into a form that looks like this or there is just a single matrix. So, it is just a kind of let us say mathematical or algebraic hack, and we end up getting something that is algebraic equivalent.

(Refer Slide Time: 03:17)



So, let me just you know there is no real magic to it other than just recognize that it algebraically computes the same thing. There are some deeper interpretations having to do with projective geometry, but I do not need to go into that for this part. So, we make a

4 by 4 matrix that is algebraically equivalent when applied to a point. So, what I do is I take my xyz point over here, when I just applied the rotation and translation in the standard way my point is just x y z, over here I am going to extend the vector by one dimension and just put a one at the end. So, I am going to have the point look like this I will put a one at the end, but I will just know that this one is something extra that I am using to perform an algebraic trick.

So, I am not really increasing the dimension of the point itself, and then I am going to calculate some x prime y prime z prime and the results going to be one for that, and now have a 4 by 4 matrix, and the upper the upper left 3 by 3 component is just going to be the rotation matrix. So, this will be the rotation matrix R just copy it exactly from here and place it in, and then I am going to take my translation parts and put them down here x t, y t, z t and the bottom row is going to look like the a piece of the identity matrix.

So, it would be 0 0 0 1 and then I will apply this matrix to my new 4 dimensional vector x y z 1. So, I go ahead and perform this multiplication now, do we believe that we get the same result as we would from performing this multiplication right. It should not take too much to see the algebra of that for example, if I just look at the calculation of x prime, I get I get the upper row of the rotation matrix the first row of the rotation matrix applied to this column xyz here right. So, that gives me exactly the rotation part, and then because I am carrying this one along here right, I get because I am carrying this one here this one will multiplied with this x t and just add it on right when I take this inner product between this 4 dimensional row, in this 4 dimensional column I add on this xt that is exactly what I get over here right I am just adding on this x t for the translation.

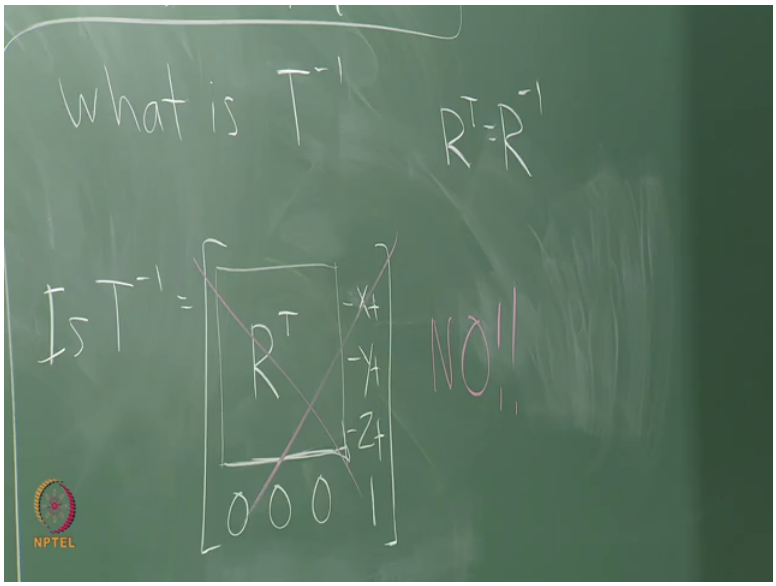
So, it is just a very simple hack that converts it into what is called homogeneous form. So, this is a homogeneous linear transformation and if by the way you want to add some extra part to it where there is an offset vector, then this is called an affine transformation. It has very different properties and sometimes it is difficult to work with and you know just algebraically a bit difference. So, if you want to start chaining matrices together, and have there be translations and rotations happening all the time then you do not have to switch operations, you do not have to sometimes add on a vector sometimes multiplied by a matrix. If you use homogeneous transformation matrices then you are always just multiplying matrices to achieve both rotations and translations.

So, that makes it very nice people who do work in computer graphics who want to code algorithms very efficiently and say GPUs, would like to use the same piece of code or the same circuitry let us say over and over again. So, that is another motivation for putting everything into one standard form and just performing the same algebra every time. I have an interesting question for you. Now, what is the inverse of this right. So, this is a homogeneous transformation matrix this 4 by 4, that performs rotation followed by translation. Well before I asked about the inverse completely though before you answer that we should also understand what is the axis of rotation here right so.

So, if I translate first and then rotate is that the same as rotating and then translating, right. So, if I translate first then have effectively moved the axis of rotation. So, you could do that you could consistently go in to find things that way; we are going to do it and be easier to interpret way which is first apply the rotation so, that we when we think about the axis angle representation, we are rotating exactly about that axis in the axis angle representation.

So, you grab the book here. So, if we are if this were the origin you first apply the rotation and then translated somewhere. If you were first to translate it somewhere then the rotation would be offset by quite a bit right depending on how large this displacement was that was induced by the translation. So, it is something to pay attention to alright. So, these operations do not commute. So, if I take the inverse of this I need to pay attention to that.

(Refer Slide Time: 08:54)



So, just something to warn you about and so, what is T inverse and I have not written exactly what T is; T is this 4 by 4 matrix. So, will say T is the 4 by 4 matrix.

So, this part here that is just the matrix T; so, what is T inverse let me write a candidate for it. So, is T inverse equal to well why do not I just invert the rotation matrix what is the inverse of a rotation matrix.

Student: (Refer Time: 09:37).

Just the transpose right. So, this only happens for what are called orthogonal matrices; and orthogonal matrices are also have might you might want to call them orthonormal matrices, because the columns are have been normalized as well. So, not only are they orthogonal columns, but the lengths are normalized as well the magnitude of the column vectors. And so, if I do that I could just maybe let us say take the 3 by 3 part here and I will just put R transpose in there, because R transpose equals R inverse that does not work for general matrices, but it works here.

So, we can go ahead and do that, and now let us see why do not I just negate the translation. So, I could that is the inverse translation right. If I move forward five meters the inverse of that is moving backwards 5 meters. So, why do not I just negate that. So, I can go minus x t minus y t minus z t and then I complete the rest of this matrix 0 0 0 1 how is that is that the inverse.

Student: No.

Why not?

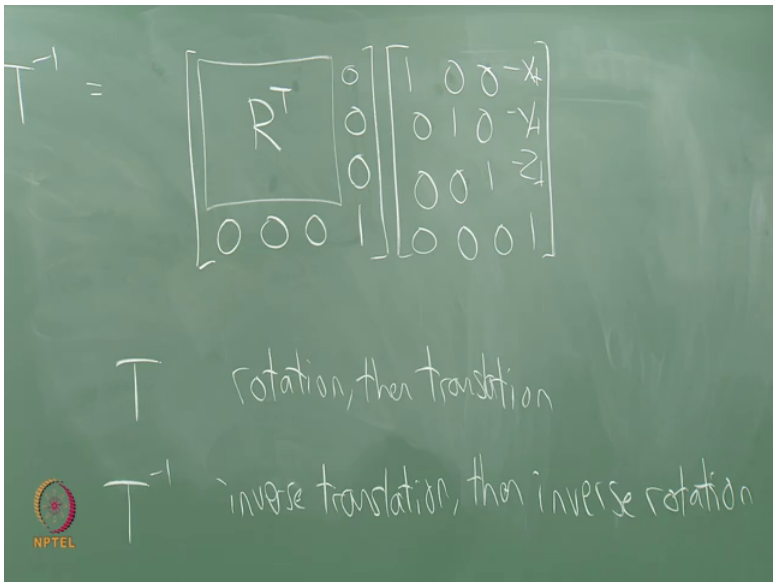
Student: In case its every (Refer Time: 10:48).

Ok, very good. So, so the operations in the wrong order here. So, if you apply this matrix it is going to first unrotate, and then untranslate if you like and then inverse translate that seems fine except for that point that I made a little bit earlier when you have non commutative algebra you have to swap the order of operations, when you take an inverse right you have to invert the order of operations. So, this is not going to work. So, the answer is no for that.

So, let us write what the inverse actually is, it is very helpful to observe these kinds of things when you are writing code, when you are doing development for these systems, I found it very helpful myself when doing development and oculus just being very quickly able to find inverses change order of operations fix different kinds of bugs that happen. Very often you can get things working most of the time and then doing something catastrophically bad like maybe the yaw and the pitch will be aligned correctly and then the roll is backwards for some reason.

So, lot of things happening like this, and these are the kinds of tricks I think that are that are helpful the kind of insights let us say, that are helpful that can save you a lot of time in trouble.

(Refer Slide Time: 12:01)



So if we go T inverse; for the first thing I would like to do is undo the translation right because that was the last thing applied. So, to undo the translation I will put it over here, I will make a matrix that undoes the translation. If I want to make a perfect matrix that undoes the translation, what should go inside of here?

The identity rotation, which also just looks like the rest of an identity matrix, so that just undoes the translation that gets applied first, but remember that I am going from right to left that is why I made a big deal out of that as well. So, we are in from right to left that gets applied first and then we come over here, and in this case we undo the rotation. So, I have my R transpose there and then I fix the rest of this matrix here, I had better not do any translations here now, this is a pure rotation. So, I just put zeros and I complete the bottom in the same way always for these I do not need to multiply that out and simplify it you can go ahead and do that there is not you know this is a fairly generic matrix here anyway. So, I like to keep it in that form. So, once again to summarize T is rotation, then translation when applied to a point and T inverse is inverse translation, then inverse rotation questions about that.