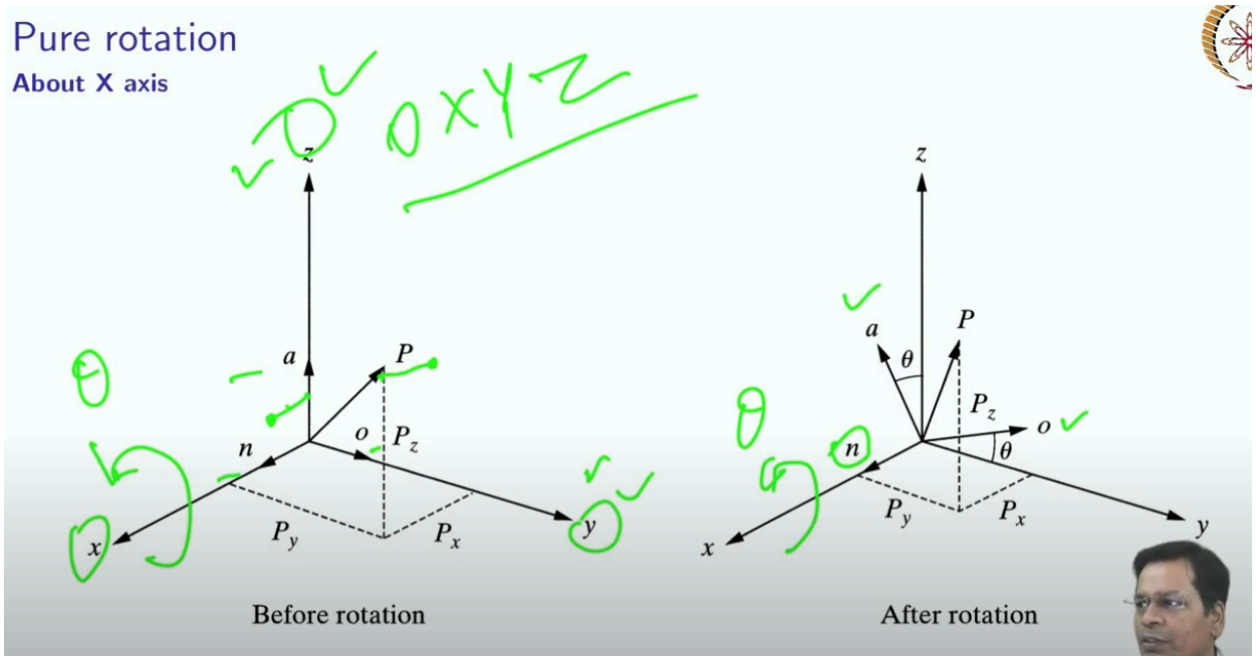
NPTEL Online Certification Courses
Industrial Robotics: Theories for Implementation
Dr Arun Dayal Udai
Department of Mechanical Engineering
Indian Institute of Technology (ISM) Dhanbad
Week: 04
Lecture: 17

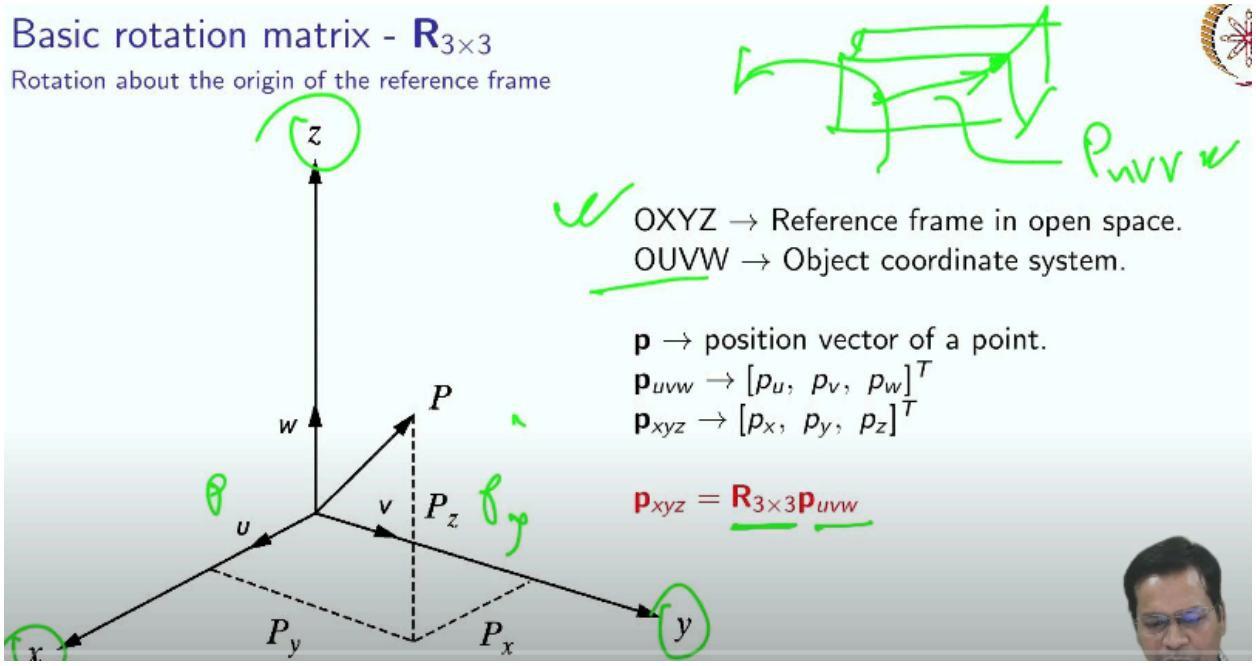
Pure Rotation, Arbitrary Axis Rotations, Euler Angles

Welcome back. So, in the last class, we discussed kinematic transformation. We saw how to use the degree of freedom for a robot. While discussing kinematic transformation, we came across a homogeneous transformation matrix, for which we have already discussed what is translation matrix is. So, today, we will move ahead, we will discuss the rotation matrix, we will discuss Euler angles, we will discuss arbitrary axis rotation, and we will continue further with the rotation matrix. Let us move ahead.

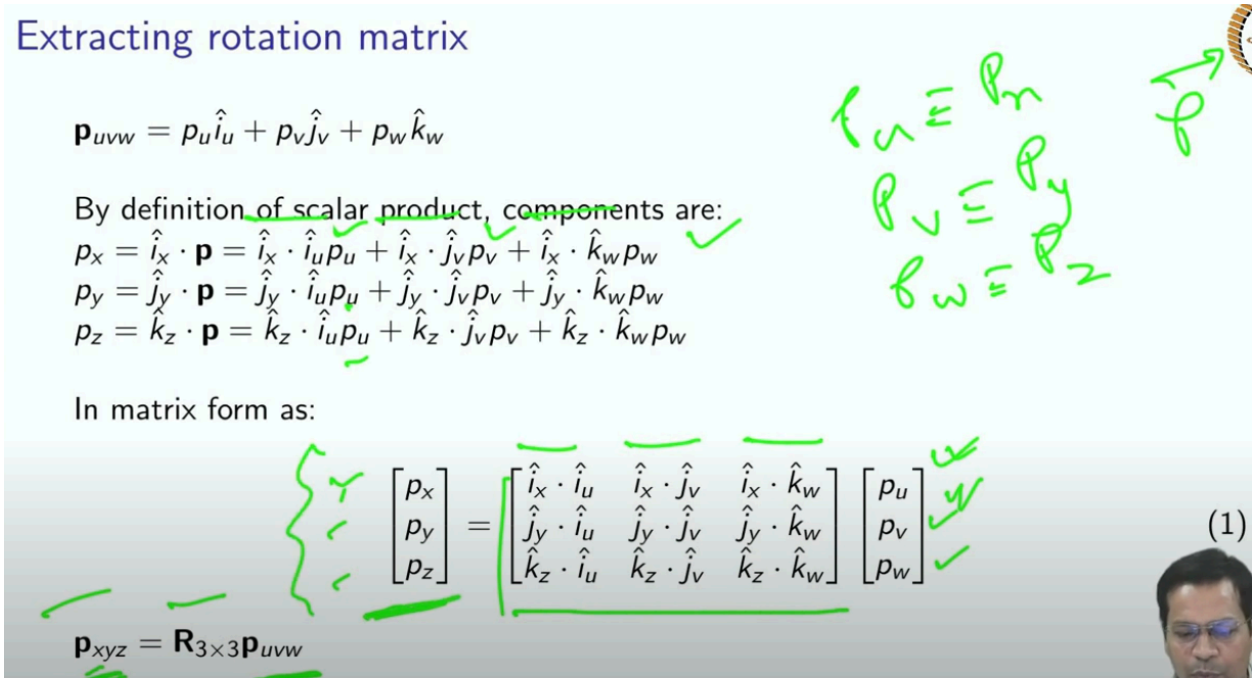


Let us just see what a pure rotation is. So, if an object is in space, which is in a global space of the frame, O, x, y, z, so that is the frame, that is this one I am talking about. So, if this object is in this space, if I rotate about the x-axis by an angle, theta to say so, let us see how to represent them, how to find out a transformation matrix which actually can achieve this, which actually can achieve this. So, if n o, an is the three unit vectors that were associated with the frame which was attached to that object. So, n remains where it is because, that is, it was aligned along with the x-axis, about which we were rotating, about which we are rotating by an angle theta. So,

what is going to change is O and A, which will be rotated by an angle theta. So, that is what, so let us see.



So, what we intend to find out here is a rotation matrix 3 x 3, that is, the rotation transformation matrix, has the capability to rotate the point p u v w, which is in object space, that is, in object space. So, if this is an object with a position vector, this is from this corner, so this is p_{uvw} in this frame. So, if this object rotates about one of the axes, what will be the consequence of rotation on uvw when projected to, when seen from the global coordinate system, that is, OXYZ? So, OXYZ is the reference frame in open space about which rotations are happening, and OUVW is a frame which is attached to the object coordinate system. So, that is there. Then, I want to find out the new point location in the frame OXYZ after rotation, which has happened to the object. So, this is what is our intention.



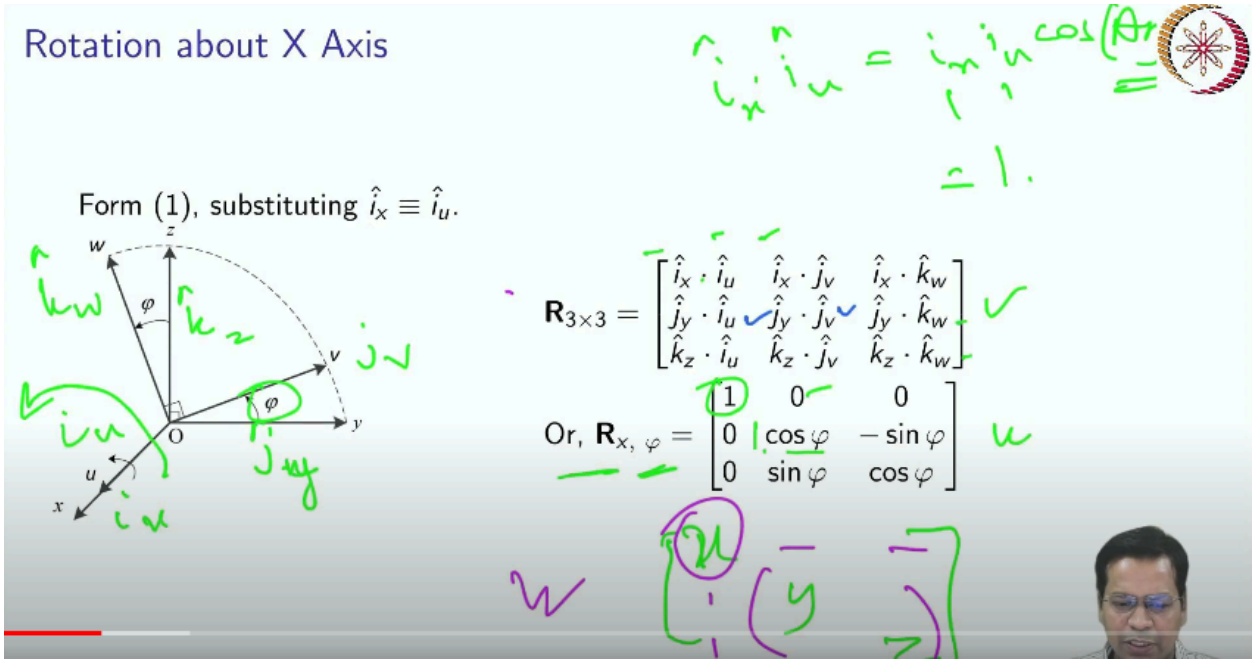
Let us see how this point has initially represented this point in the object space. Initially, the object axis is well aligned with the x, y and z axis of the OXYZ frame. OXYZ frame, u, v, w were aligned with this. So, pu was equivalent to px, which is the coordinate of. So, pu was equivalent to px. px is the projection of this point along the x-axis. Similarly, py was a projection of p along the y-axis. So, this was equal to p v, an equivalent initially. And similarly pw was equivalent to pz initially, without rotation.

So, let us come back to this. So, this was the point. Now, I have to take a projection of this point in order to find out. Let us say this has already rotated. Now, I want to find out the projection of this vector along the x, y, and z axes, So that effectively gives me what I want. So, by scalar product definition, we know if at all we have to project a vector along any of the axes. So, I can directly take a unit vector along that axis and take the dot product with the given vector. So, if I take a dot product that has the capacity to project that vector about which you are taking the dot product, about its direction.

Okay, so this is what? So, px? Now I want to find out what px is. What is that? This is the projection of p along the x-axis. So, what was that projection of p along the x-axis? I simply took a dot product, and this is what it is. So, this is nothing but a dot product. It has a scalar value of px. So, now I will get px. So, that is the projection of position vector p along the x-axis.

Similarly, if I take dot product jy and p, so I will get py. This is now projected, this vector p along the y-axis, similarly p z. So, all three can be obtained like this. What were iv, jv, and iu? So, what was iu, jv and kw? So, these were unit vectors along the u, v, and w axis in the object thing. So, these were unit vectors.

So, now let us write this in compact notation like this. So, if I can write this in vector form, so you see, this is pu, pu, pu, so all pu will come here. So, this was here, this is here, and then pv, p v and pw, pw. So, ix iu into pu, i x jv into pv, ix kw into pw, and that is how I rearrange that in a metric form. What, effectively, is this saying? So, this is your initial point, which was initially pu was equivalent to p? V was equivalent to p y, p? W was equivalent to p? z. Now that it has rotated. So, what are my new coordinates? Along the x, y and z axis. So, this is the required transformation matrix which is able to do that. So, what I am getting is the projection of position vector p, which was initially there, along the x, y and z axis. So, this, effectively, is your new point. So, what is it? This is p x, y, z. initially, it was p? U, v, w. that was the initial point. So, after rotation, the new projections are this: So these are the new coordinates. So, this is the new position vector. Say: So, this is what is your rotation matrix? 3. 3. So, that is what was required.

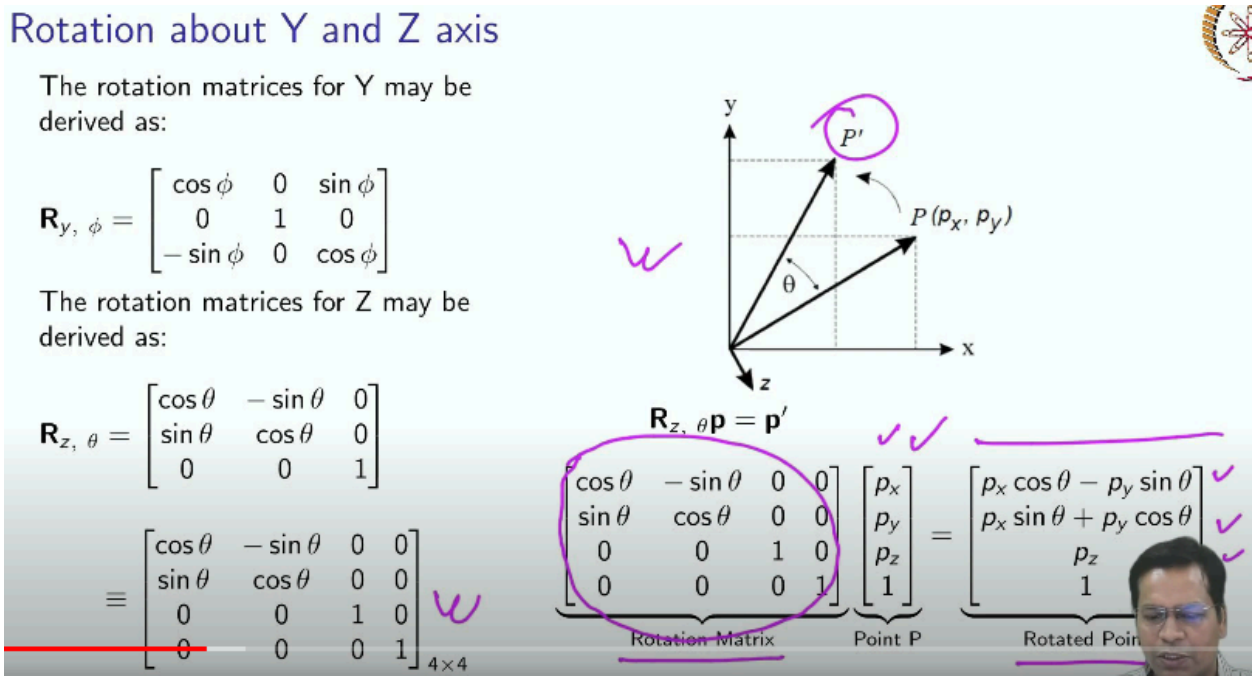


So, now let us say, can I derive pure rotation about the x-axis? So, in order to do that, let us start with what we have obtained in this one here, and then can I substitute now? So, iu is not changing. iu is aligned with ix, whereas jv has changed. jv has come to a new location.

Similarly, kw has changed, and it has come to a new location, and jx, kz or wherever it was. So, it is in the same position. Now, substitute ix and iu here, and the angle between them is 0 degrees. It has not changed. So, ix dot iu is equal to ix iu cosine of the angle between them, correct? So, because the angle is 0 degrees, it is 1, and these were already unit vectors. Effectively, this is giving you 1. So, that came here. Now, let us see what is happening to the other parts of the matrix. So, this is ix into jv. ix into j v. Both are always orthogonal to each other. So, if they are orthogonal, the angle between them is 90 degrees. So, the dot product becomes 0 degrees.

Same with this, same with this, same with this. So, all these elements will be shifted. Now let us see what are the other parts. So, now, what is the angle between jy and jv? So, this was jy and jv.

So, the angle between them is this. So, what happens? And it is rotated like this, so that is. As for the right-hand thumb rule, it is in a positive direction only, so it will be cos phi. So, jy, and jv both were unit vectors that turned out to be one and cosine of 90 degrees. So, that came here.
Similarly, when you work out for other parts of this matrix, you will see it is like this. So, this is a rotation matrix for the rotation about the x-axis by an angle z. This is it. So, it gives you. So, there is a very good note over here. Just if I can tell you, this represents x, this represents y, this represents z. So, in order to rotate about the x-axis, you have to make this one and the corresponding row and column 0, and this is the element which will have some changes that are to be taken care of. Otherwise, it looks like this. This makes so easy to recall your rotation matrices. So, this is rotation about the x-axis by an angle z. This is it.



Similarly, if I can obtain a rotation matrix for the rotation about the y-axis by an angle phi that can be derived using the same startup point, that is this one. If you start with, you will obtain this and rotation about z axis may be derived as this. So, this is rotation about z axis, as I told you. So, this was equivalent to. This is equivalent to 1. This should change the corresponding row, and the column will go to 0. So, this is what? So, it is cos minus sin and cos theta.
Similarly, in the case of y, it is cos sin minus sin, cos only. This has changed its sign. Otherwise, it is the same. So, this is what is your rotation matrix. If I write down in 4 cross 4 homogeneous transformation matrix form, it will look like this: So this is your rotation matrix subpart. So, this is for this only, but it is in 4 cross 4. This is a pure rotation matrix, whereas this is a homogeneous transformation matrix with a rotation matrix sub-matrix inside it. It says: There is no displacement. So, there is no translation. Which is happening? This was 0, only. This was one as it is. So, this will cure. This will present your rotation matrix in 4 cross-4 homogeneous transformation matrices. So, let us move on. So, what does it actually look like? So, this p has

now come to p dash. Projections have changed. If it is represented in its own frame, the coordinates will remain as it is, So pu, pv and pw will remain as they are, whereas the projection of u, v and w along the x-axis will change. So, new projections are px, y and, similarly, PC. So, that is what we did. Actually, we found out. So, p dash is the new point that is obtained by revolving by an angle theta about z axis. This is your z-axis. The positive direction of rotation is done, and you have a new point that is this. So, let us do that. So, this is your initial point: PX, by, pz1. Okay, point in the frame, the way we will be representing from nano. This is your 4 cross 4 rotation matrices, exactly this one. So, if you apply this trans, Formation, this transformation, to this point, you reach a new location. So, this is the new coordinate of your point. So, that is p dash. Coordinates of p dash you can quickly derive using geometrical technique also. You will arrive at this rotated point. So, this is how you have to do rotation.

Composite rotation matrix

Sequence of finite rotations are important!

e.g.: $\mathbf{R} = \mathbf{R}_{y,\,\phi}\,\mathbf{R}_{z,\,\theta}\,\mathbf{R}_{x,\,\alpha}$

$$= \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\phi\cos\theta & \sin\phi\sin\alpha - \cos\phi\sin\theta\cos\alpha & \cos\phi\sin\theta\sin\alpha + \sin\phi\cos\alpha & 0 \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & 0 \\ -\sin\phi\cos\theta & \sin\phi\sin\theta\cos\alpha + \cos\phi\sin\alpha & \cos\phi\cos\alpha - \sin\phi\sin\theta\sin\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Try changing the order now!

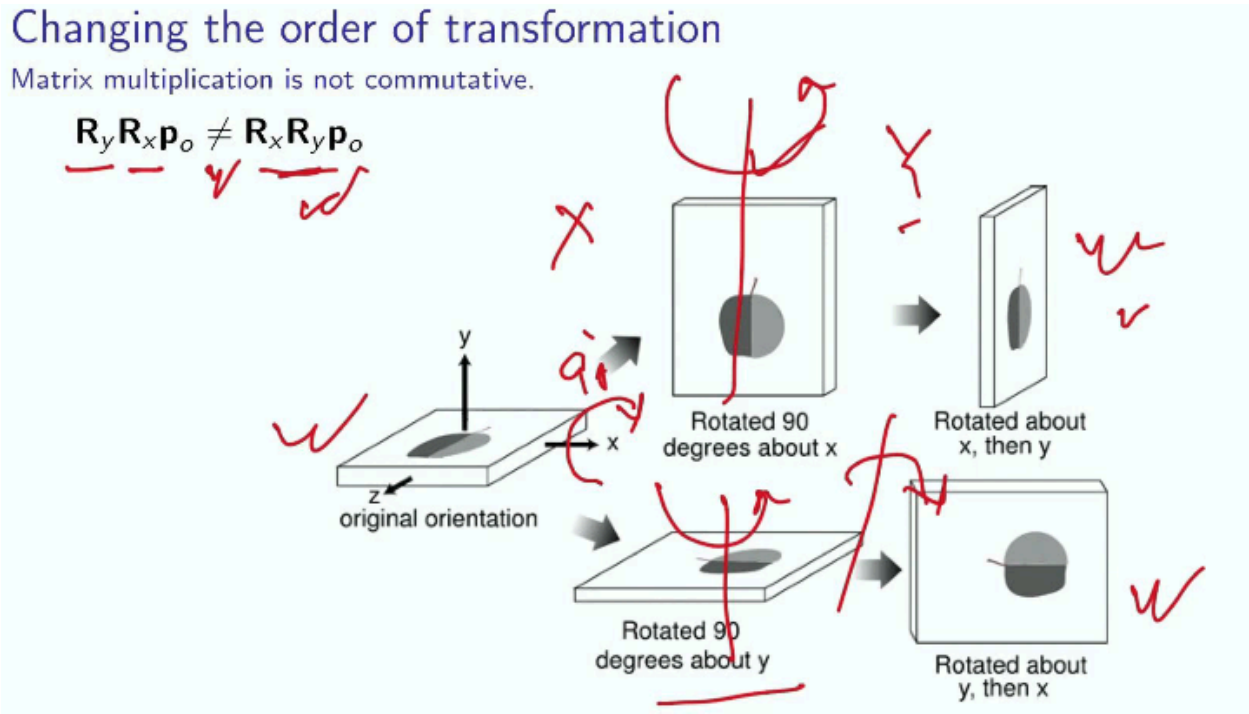
TODO: Use any symbolic computation tool to perform the above task. e.g.: MuPAD/MATLAB (Windows), Wolfram Mathematica (Windows/Raspberry PI/Android) wxMaxima (ubuntu).

Now, let us look at composite rotation. Okay, So how to go for it? Let us say you have a point p, which is to be applied with rotation about the x-axis by an angle alpha, the z-axis by an angle theta and the y-axis by an angle 5. So, how will you go about it? So, here goes your point. So, this is your initial point, given as Px, Py, Pz, 1. So, on this point, you have applied transformation first transformation, that is, rotation about the x-axis by an angle alpha. What have you done? Rotation about the x-axis by an angle alpha. So, this is your required rotation matrix. When applied to this point, you will reach a new point. Okay, You will reach a new point. Then you apply the resultant of this. Resultant of this will. Give you a new point. Now, to apply a second transformation.

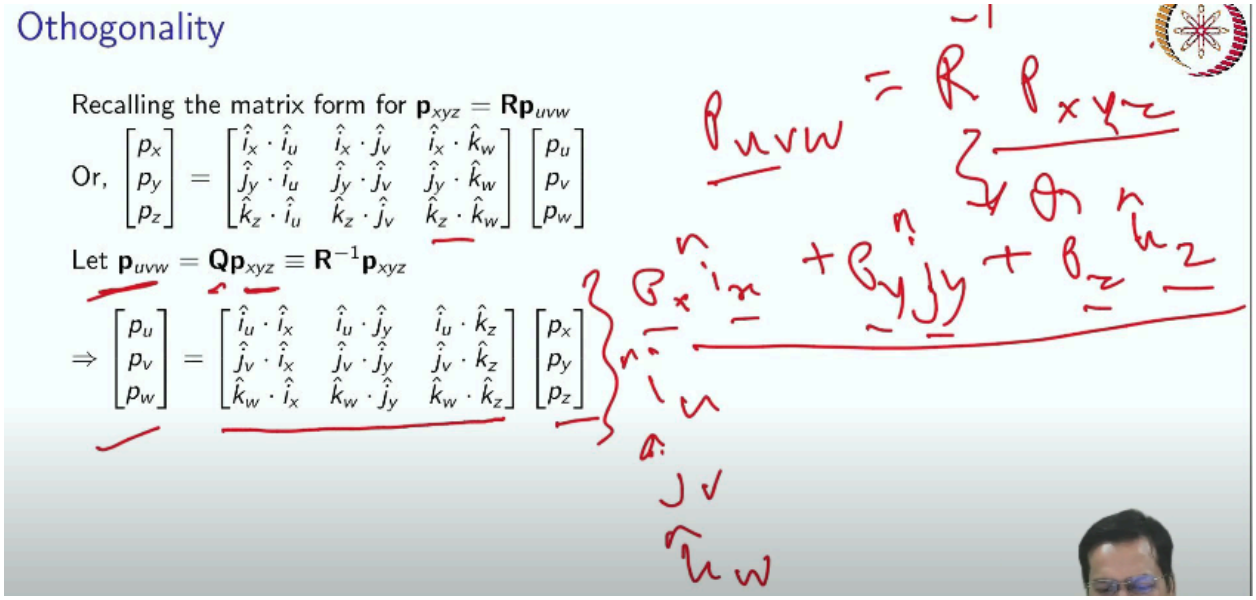
The second transformation is there, which says rotation about the z-axis by an angle theta. Rotation about the z-axis by an angle theta. So, at this point, the p dash is now at. So, this was initially P. Sorry, as you do the same colour, this is initially P. Then you came to p dash. Now, the P dash is acted upon by this rotation matrix. So, you will reach a new point, that is, P double

dash. That is your new point. After applying this, this matrix, okay, And now, finally, the third transformation which is there, that is rotation about the y-axis by angle phi. Okay, So about the y-axis by an angle phi. Okay, you have applied that rotation matrix to the resultant point, P double dash. So, you reach a new point, which is the P triple dash. That is your new location. So, this is how you have to apply. And then, in a composite way, if you can multiply all these together, you can multiply all these together- you reach a new matrix, which is known as a composite matrix. That can be in a single operation. This has a capacity to rotate about the x-axis by an angle alpha, the z-axis by an angle theta, the z-axis by an angle theta and the y-axis by an angle phi. So, all three rotations are now in a four cross-four matrix form. So, this is a four cross four matrices, and it can directly be applied on the point, and you get to p. triple dash is straight up, okay. So, yes, this is very important to remember that rotation matrices are not commutative. That means the sequence is very, very important, not just a rotation matrix. Mathematically, matrix multiplication is not commutative in nature. A into b is not equal to b into a, not it. So, the same applies here also, even for the rotation matrix. Results are different. So, that will take you to a different final composite transformation matrix, in which you change the order. So, if you can try changing the order in, you can see when you reach okay, you will see this matrix itself is different, and so if your product of this matrix is to the initial point, and finally, you will reach a new point. Okay, You can try doing that using various tools that I have listed here. This can be done using any symbolic computation tool to perform the above task.

You can use a MuPAD in Matlab if your university has it or Mathematica if it is there. That is free, with Raspbian for initial use, to come, even on Androids. wxMaximal is freeware is there. It can run on ubuntu. It can run on Windows, and it can run on Mac too. So, this is free. So, you can use any of these tools to do symbolic computation. You do not need to have an angle for it. You can directly use alpha, theta, psi, or whatever symbols that you want to go at.



Let us just see by an example, as we have seen just now. So, r, y, and x are in a different order as compared to this. Both are not equivalent. Let us see by demonstration. Initially, let us say you were here. Now you have rotated about, rotated about the x-axis, about the x-axis by an angle of 90 degrees, and you came here. Now you have rotated about the y-axis by an angle of 90 degrees, and you have finally reached here. So, the first rotation was about the x-axis, and then it was about the y-axis. Now, see the change in order. Now, I want to rotate first about this part of it, so first about the y-axis by 90 degrees, and then I have rotated about the x-axis by 90 degrees, about the x-axis by 90 degrees. I reached here. So, you see, the orientation of both objects is different. Object which was rotated, x and y, then later y and x, in a different sequence. It reached a different place. So, you see, this is by example, you can just see, Okay.



So, now let us see some properties of a rotation matrix, and they are very, very important while we deal with rotation matrix in order to simplify our expressions quite a lot of time. So, yes, this was the initial thing that we solved earlier. So, Puvw, acted upon by a rotation matrix which looked like this that gave you a rotated point in the frame OXYZ if I want to do the opposite. Okay, if I want to do the opposite, what should be done? So, I can write simply like this: so Puvw equal rotation matrix inverse, if I can take the inverse of this and Pxyz.

$$P_{uvw} = R^{-1} P_{xyz}$$

So, this should be the case, right? Let us say we represent this using a new matrix called Q. So, Q applied to Pxyz should give me Puvw. Now, let us derive this from the first principle. Now, what do I actually want to do? Initially, the vector was px of x, then py and pz, so initially, the vector was px.$\hat{i}$x+ py. $\hat{j}$y + $\hat{k}$z. kz. Okay, so this was your initial vector. Now, I have to take a projection of this on Puvw. So, I have to take the top product with $\hat{i}$u, $\hat{j}$v and $\hat{k}$w, so three dot products will give me a projection of this vector along Puvw. So, rearranging that this time will take you to this, and this is what you see. This is your new rotation matrix to transform Pxyz in the P_{uvw} frame. Okay, express px, py, pz in Puvw. Okay, so we can. Simply, we have just taken projections of that.

Othogonality

Recalling the matrix form for $\mathbf{p}_{xyz} = \mathbf{R}\mathbf{p}_{uvw}$

$$\text{Or, } \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} \hat{i}_x \cdot \hat{i}_u & \hat{i}_x \cdot \hat{j}_v & \hat{i}_x \cdot \hat{k}_w \\ \hat{j}_y \cdot \hat{i}_u & \hat{j}_y \cdot \hat{j}_v & \hat{j}_y \cdot \hat{k}_w \\ \hat{k}_z \cdot \hat{i}_u & \hat{k}_z \cdot \hat{j}_v & \hat{k}_z \cdot \hat{k}_w \end{bmatrix} \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix}$$

Let $\mathbf{p}_{uvw} = \mathbf{Q}\mathbf{p}_{xyz} \equiv \mathbf{R}^{-1}\mathbf{p}_{xyz}$

$$\Rightarrow \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix} = \begin{bmatrix} \hat{i}_u \cdot \hat{i}_x & \hat{i}_u \cdot \hat{j}_y & \hat{i}_u \cdot \hat{k}_z \\ \hat{j}_v \cdot \hat{i}_x & \hat{j}_v \cdot \hat{j}_y & \hat{j}_v \cdot \hat{k}_z \\ \hat{k}_w \cdot \hat{i}_x & \hat{k}_w \cdot \hat{j}_y & \hat{k}_w \cdot \hat{k}_z \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \equiv \begin{bmatrix} \hat{i}_x \cdot \hat{i}_u & \hat{i}_y \cdot \hat{j}_u & \hat{i}_z \cdot \hat{k}_u \\ \hat{j}_x \cdot \hat{i}_v & \hat{j}_y \cdot \hat{j}_v & \hat{j}_z \cdot \hat{k}_v \\ \hat{k}_x \cdot \hat{i}_w & \hat{k}_y \cdot \hat{j}_w & \hat{k}_z \cdot \hat{k}_w \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \mathbf{R}^T\mathbf{p}_{xyz}$$

(As dot product is commutative)

$$\Rightarrow \mathbf{Q} = \mathbf{R}^{-1} = \mathbf{R}^T$$

Now, $\mathbf{Q}\mathbf{R} = \mathbf{R}^T\mathbf{R} = \mathbf{R}^{-1}\mathbf{R} = \mathbf{I}_3$, a 3×3 identity matrix.

Hence, $\mathbf{R}$ is an Orthogonal matrix.

So yes, rearranging that, because dot product, you know, is commutative: i_x into i_u, is, i_u into i_x. Both are the same, so I have just flipped them, and what do I get as the dot product is commutative? So, it is. It is exactly like this, okay, and what you see, this column is now appearing here as an R transpose. It is R transpose. So, what you find here is something that can do inverse transformation. R inverse is equal to R transpose. So, what does it say? Q now is equal to R inverse. It is also equal to R transpose.

$$Q = R^{-1} = R^T$$

So, what it is basically now is q, R is equal to R transpose, R is equal to R inverse, and R is equal to the identity matrix.

$$QR = R^TR = R^{-1}R = I_3$$

So, what is it? This is basically an orthogonal matrix. So, rotation matrices are orthogonal. Okay, they are orthogonal, okay, and you will also observe rotation matrices, each of them as, because they are a projection of unit vector along x, along y, along z. Each one of them is are unit vector. So, this is a projection of u along x. uv along y this is k along z. okay. So, this is what. Okay, so these are three vectors, these are three vectors, and overall, again, the whole of this is you take the determinant of this rotation matrix, it will give you one, okay, so you can try doing that. But yes.

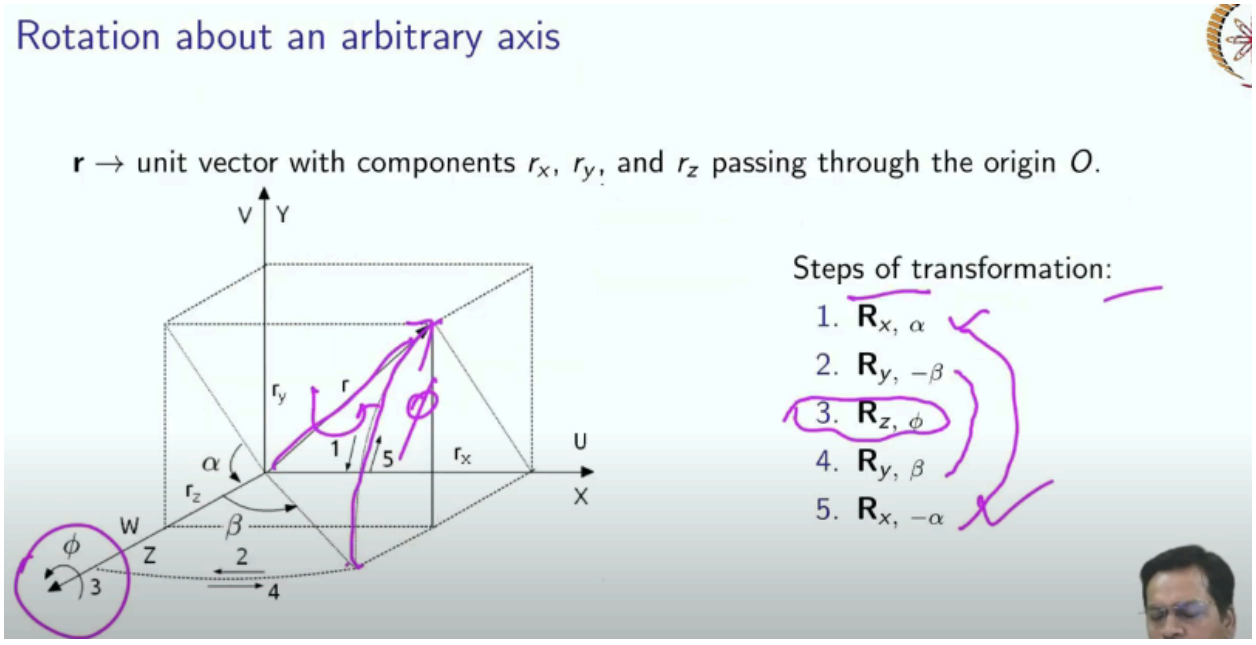
Notes:

1. If the rotating coordinate system *OUVW* rotates about one of the principal axes of *OXYZ* frame, then *pre*-multiply the previous (resultant) rotation matrix with an appropriate basic rotation matrix.
2. If the rotating coordinate system *OUVW* rotates about its own principal axes, then *post*-multiply the previous (resultant) rotation matrix with appropriate basic rotation matrix.
3. All analogous in case of homogeneous matrix as well.
4. Inverse of rotation matrix is equivalent to its transpose.

Let us move out and just see a few notes, which are very, very important. If the rotation coordinates system OUVW rotates about one of the principal axes OXYZ, then you saw you were doing a pre-multiplication of the previous resultant, okay, with the appropriate basic rotation matrix. What do I mean? So, first, let us start with the point: you apply the rotation matrix. These two are new points. Now, for the second transformation, you have to apply the p dash with the next transformation. Okay, not just rotation can be any transformation, okay. So, you did this. This is equivalent to r1 into p, and you have a second transformation that is R1, R2, p. okay. So, because the p dash is R1 is p, I have just substituted it here. You see, you just pre-multiplied that. So, this is what. So, you both were in projected on p top, okay, so this is what is the meaning of this.

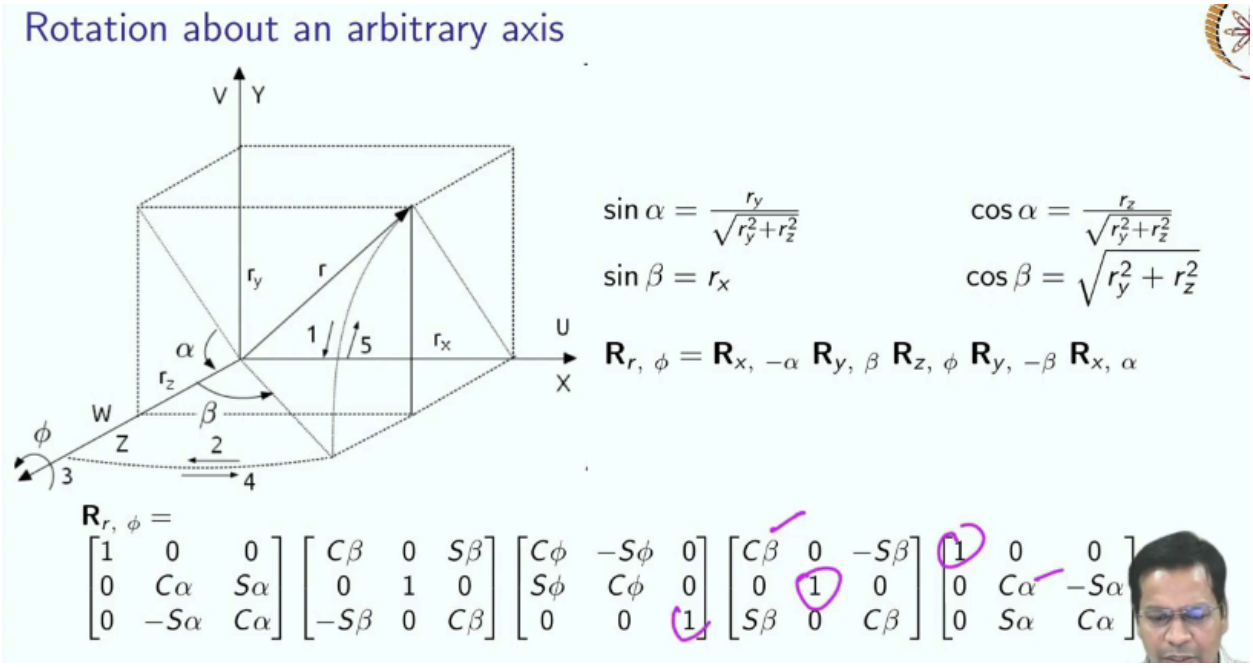
Next, this time, if the rotating coordinate system rotates about its principal axis, that means once you have reached there, the object rotates in a new coordinate system where it has reached. Okay, if this object has come to a new place. Let me show you this. So, this is your space. Okay, so initially it rotated about its x-axis, about its y-axis. That is fine, and it can do this. It can do this. Let us say it has rotated here, and your x-axis is now along, and the y-axis is also inclined. So, in this case, what has happened, you see? So, this time, if I want to rotate about the object's x-axis, it should rotate about the axis, which is like this: okay, so this time, it will rotate like this. If it has to rotate with the original x-axis, this will go something like this: got it? So, that means the object's own axis. If it rotates, you have to post-multiply the rotation transformation. Now, what do I mean in this case? Instead of having R1 R2 over here?

The second transformation is about the object's coordinate system only. That means in the new location, once it has reached, the object has again rotated, but not with respect to the principal axis, but with respect to its own axis this time. And r1, and then you have to do r2 so that the transformation matrices will go in this sort. Okay, this is what Okay, even in the case of homogeneous transformation matrix, you have to do the same thing, not just for the rotation matrix, even for a homogeneous transformation matrix. So, you have to go in similar these two orders for respective jobs. Okay, so the inverse of a rotation matrix is equal. To transpose that. You saw, it is an orthogonal matrix. So, this is true anyway.



Now, let us do rotation about an arbitrary axis. why? So, as far as you know, you are ready to rotate about the x-axis. You can rotate about the y-axis, and you can rotate about the z-axis. You always have in hand what is known as a principal transformation matrix for rotation. So now, can I rotate about an arbitrary axis? That is very much required. Let us say a robot has reached a place. Now, it doesn't want to rotate about the global frame; rather, it wants to rotate about its own frame, where it is okay. So, can we do that? So, in order to do that, do that. It is a standard transformation technique, not just um specific to robotics. I tell you so. It is there in computer graphics also. It is there in animation software, anything. So, the same thing we'll be doing here. We'll try to align. Let us say this is your object. Let me start by explaining this. So, this is your initial vector. Okay, I want to do a finite rotation about this axis, about an axis which is long in space. It is not aligned with x, not aligned with y, not aligned with z. Can I do that rotation? Rotation by an angle, let's say theta above vector, which is r. r is a unit, and the projection of r along x is rx, the projection of r along y ry, by along z is rz. So, you can create a geometrical cube like this of dimension, rx, ry, and rz for the three orthogonal directions. So now, I'll start aligning this. Now, I'll start aligning this to one of the axes. Let us say I have chosen to align this r about the z axis. But how will I go about it? The first rotation will be, let us say, rotation about the x-axis by an angle alpha. So, what I have done is your x-axis. You have rotated it by an angle alpha. And what happens if this position vector, this r vector, now comes down like this, and you come to this location? Okay, and this is your r, the distance, that position vector length is not going to change. so it has come to a new location. That is r. So, this is it. So, this was the initial vector. So, it was. This was ry, so that remains as it is. So, now, this is it. So, this is your first transformation rotation about the x-axis by an angle, alpha you were raised here. Now, the second time, you see you have to follow the right-handed coordinate system in order to measure any angle. So, it was positive. In the earlier case, it was a positive one. Okay, alpha was positive.

This time, I have to do rotation about the y-axis by an angle minus beta. Why minus beta? This is your positive direction, but you have to be sorry. This is your positive direction, but I want to rotate this vector down in order to align with the z-axis. This r should now be rotated like this to come here. So, this distance remains r from here to here. It has come, and thus the rotation is beta, angle by negative value. So, you have to rotate by minus beta. So, that is rotation about the y-axis by an angle minus beta. So, this is your y-axis. So, you have to go like this. Okay, so now you are ready to do rotation at rotation because now you are ready, you already have a rotation matrix for rotation about the axis; that is what we wanted to do. So, this is your second transformation. So, the third transformation will be the actual one that I wanted to do. So, what I wanted is rotation about r by an angle phi. So, this is now rotation about the z-axis. So, it is this, and it is in the clockwise and counterclockwise direction. It is in a counterclockwise direction, and it is positive. So, it is Rz, ϕ. So, now that I have already rotated this object, okay, this object is already rotated. Now, I have to bring back everything to its place, so I do the inverse of this. So, this will go here, this will go here now. Okay, you have to come back now. Now that you have already rotated, nothing has to come back like this. This is here, and it also has to come back like this. again, it comes back to its original location. As you see, the first is alpha, next is minus alpha. This comes here: beta minus beta and beta. It is here. This is the actual rotation that you wanted to do, that is, rotation about r by an angle ϕ. This is what we wanted to do, and that was done here. So, what you effectively did is, instead of because you did add a rotation matrix- to rotate about an arbitrary axis. you have taken this arbitrary axis to one of the standard axes, rotated it there and brought it back to the original location. so this is your steps to take this transformation.



Now, let us just write all of them and find out the angles first because you know each one of them will constitute the elements, will be constituted of elements cos alpha, sine alpha, cos beta,

sine beta, cos phi, sine phi. So, can we calculate all those angles using this? So that can be easily found out? So, wherever you see, it is just r_x. that means you have in denominator, that is, r_x square plus r_y square plus r_z square, because r was initially a unit vector. So, this is actually a unit vector. That is one. So, that is why you see only r_x. So, let us just find out what sine beta is to say: so, this is beta. So, sine beta will be simply this distance or this distance. What is sine beta? Okay, so it is. This was initially r, so this is r. The magnitude of this is r, and this is your sine beta. Both are the same, so r sine beta is this distance, and how much is this? This is rx, okay, rx by r. So, this is your sine beta. Now, a similar one. You can pick up any one of these, so in order to do cosine beta.

You have to take this distance, actually. What it is. It is r_y square and r_z square. When you do cos beta with r_y and r_z, okay. So, because by is now aligned to this, this is exactly r_y and r_z. So, now, similarly, you can find other angles to substitute them when you do actual rotation in your rotation matrices. So, this is your order. First, you did rotation about the x-axis by an angle: alpha, y-axis, about minus beta. This is your actual rotation. Then you brought it back here. This was brought back here. Okay, so you have to do the reverse transformation. So, this is here. So, if you substitute all the rotation matrices, it is something like this station about the x-axis by an angle alpha, y-axis by minus beta, you see, and z-axis by 5. Okay, again, y-axis, beta, x-axis minus alpha.

Rotation about an arbitrary axis

$$\mathbf{R}_{r,\,\phi} = \begin{bmatrix} r_x^2 V\phi + C\phi & r_x r_y V\phi - r_z S\phi & r_x r_z V\phi + r_y S\phi \\ r_x r_y V\phi + r_z S\phi & r_y^2 V\phi + C\phi & r_y r_z V\phi - r_x S\phi \\ r_x r_z V\phi - r_y S\phi & r_y r_z V\phi + r_x S\phi & r_z^2 V\phi + C\phi \end{bmatrix}$$

where, $V\phi = 1 - \cos\phi$

Note: Other methods are using Quaternions, Rodrigues matrix, Direction cosines, etc.

So, all the matrices are here. Multiplying them together will give you something like this: where v, phi is equal to 1 minus cos is known as what sign. So, this is a composite rotation matrix. Okay, in order to do it in the shortest possible time, maybe this is one of the very good one which is normally good in computer graphics, in order to do the object transformations are done. That uses this. This is also used in robotics.

Euler Angle Representations

It reduces requirement of nine elements of rotation matrix for specifying rigid body orientation to three angles ϕ, θ, and ψ.

Table: Eulerian angle sequence of rotations

	System I Gyroscopic	System II Z, V, W	System III Roll, Pitch, Yaw
1	ϕ about OZ	ϕ about OZ	ψ about OX
2	θ about OU	θ about OV	θ about OY
3	ψ about OW	ψ about OW	ϕ about OZ

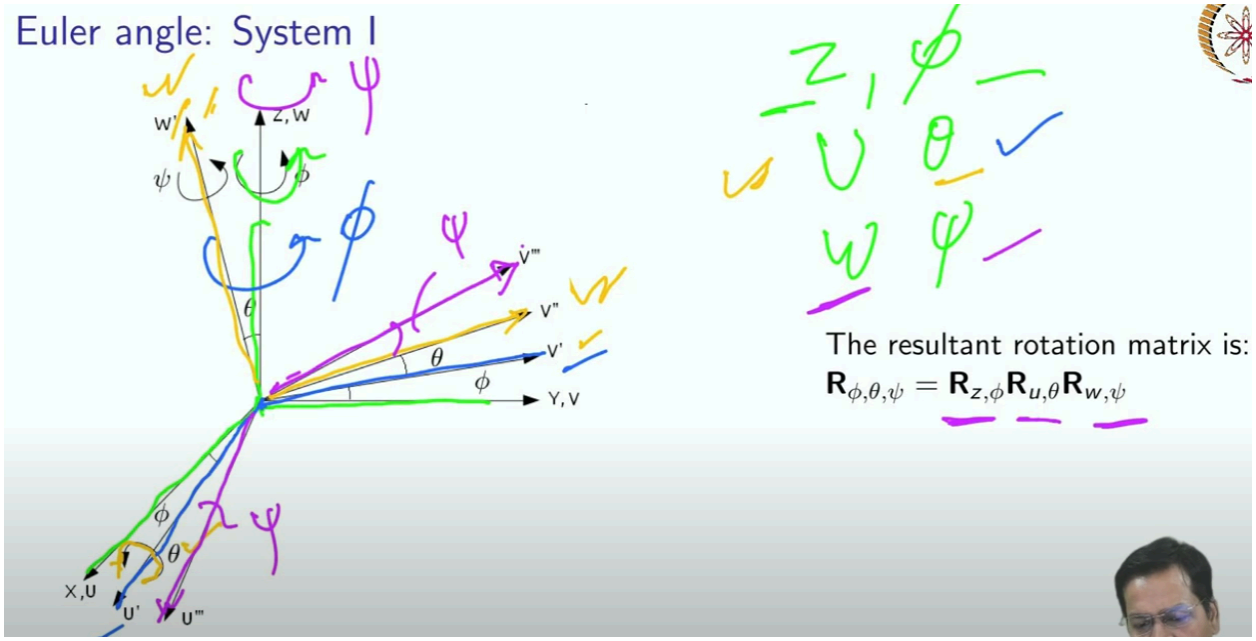
Note: These are three out of 24 different combination possible.

Then now you know, need to represent orientation the way, because orientation, you know, can be achieved using multiple ways. There are three rotations which can be done along the principal axis. So, you know, now an object can rotate about its own axis as well. Okay, so you now have another three-axis, which is in the object plane, so there can be multiple combinations to do the rotation in three-dimensional space. So, the object may be rotated about the principal axis, then rotated about its own axis, and then the principal axis. You can have multiple combinations of those. A total of 24 combinations are possible that way. So, that is how you can orient any way one in three-dimensional space by defining these three rotations. So, it reduces the requirement of. Basically, what you saw is the rotation matrix now has one, two, three, four, five, six, seven, eight, nine elements. You have to know all of them in order to find out this, but if you can find out all these elements, you know they are not linear because each one of them is a unit vector. If this changes, these two will also change any of these wills. This will change you, and you will see a change in this. So, a total of three things can change, not more than that, because there are three orthogonal vectors, you know. So, that means the reason to be in it. So, uh, by specifying in mule around it, there are three rotations which are one, that is three degrees of freedom system. Any object, if it is only a rotation which is to be done, has three degrees of freedom. You know that.

So, yes, there are three rotations, which are represented as phi, theta, and psi. that can be done in the global coordinate system or in an object coordinate. So, that is what. So, there are a few important ones which are there. One of them is a gyroscopic one, a gyroscopic one that says rotation about phi by z. This is a rotation about a universal frame. That is the global thing. That is the principal axis. And then we do it in the object frame, u direction by an angle, theta. Then you

do the object frame, again by an angle, phi. So, the three rotations are psi, theta, and phi. Okay. So, first was in principal axis, remain to wherein object.

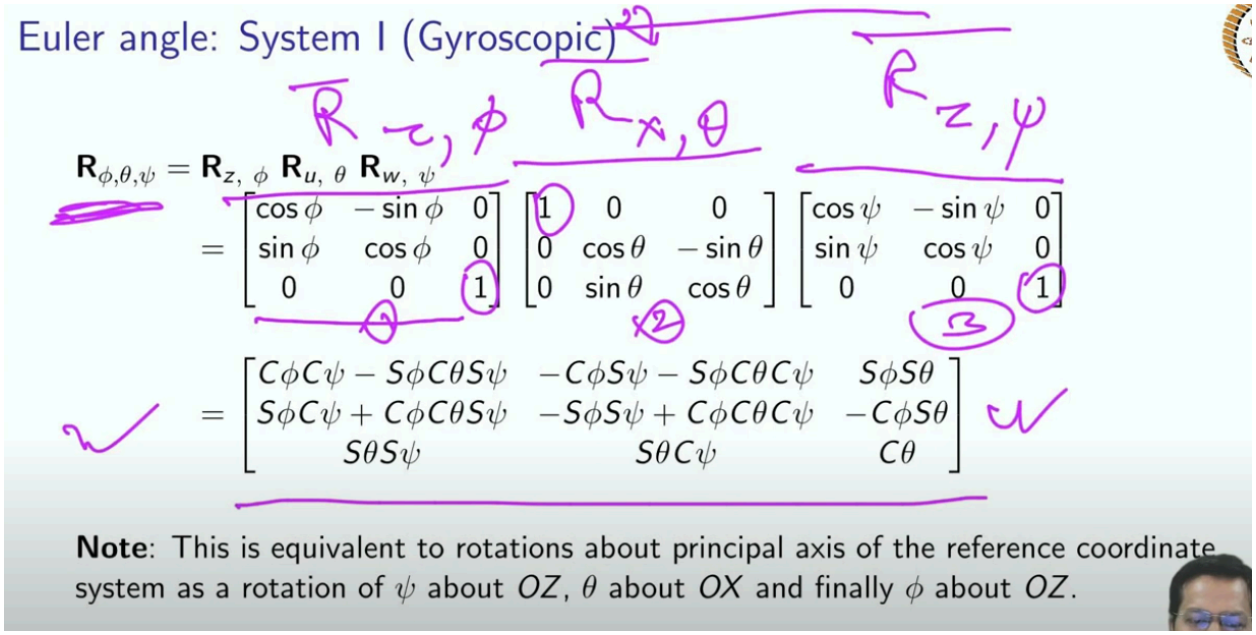
Now, similarly, system two. This is also a gyroscopic system, but instead of u, now you see, it is v, rest, everything is the same. So, position phi for the principal axis. Then, Object axis v, object axis w, phi, theta, psi. Now, this is a very, very common one which is easy to understand, that is, and your axis. We will discuss all of them in very much detail, you know.



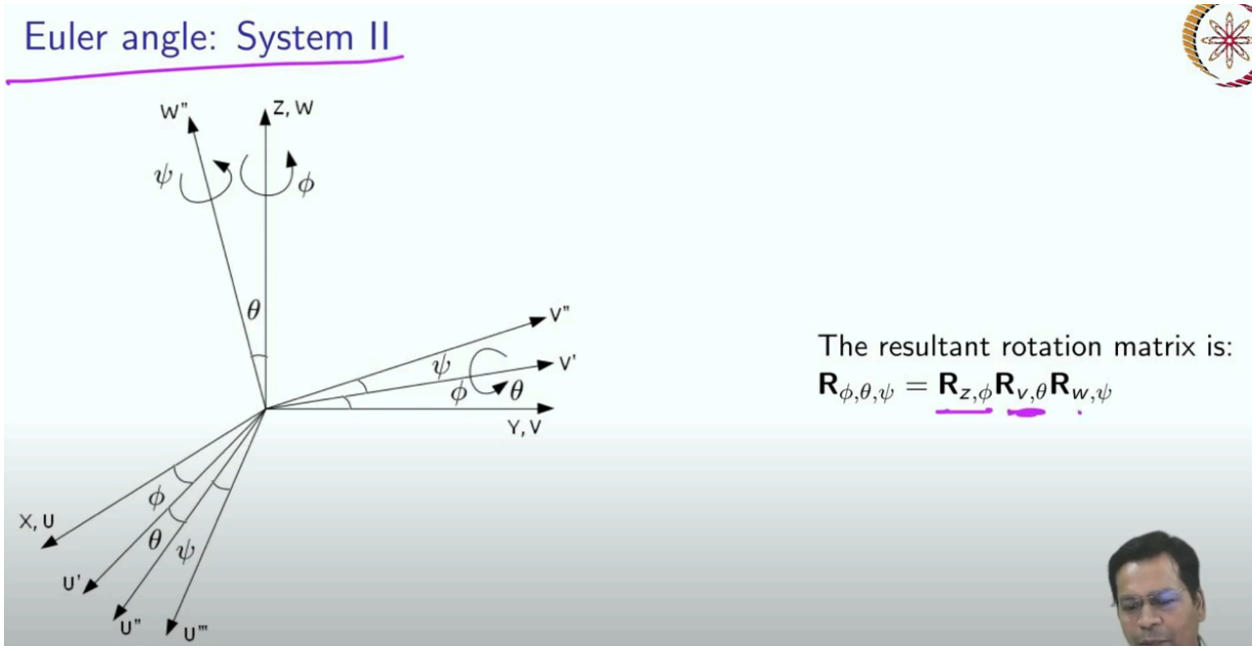
Let us start with system one, how to go for it. So, let us say you initially started somewhere over here. You initially had your vector, which was u aligned like this, v aligned like this and w was aligned here. So, the first rotation is, if you talk about system one, it is rotation about the z-axis by some angle, okay. And then rotation about local frame by an angle, then again local frame by an angle, okay, three rotations are there. So, see how it is happening. First, you have rotated about the z-axis by ϕ angle, so the z-axis by ϕ angle. So, what has happened? The new location of this is: it has come here. Now, v has come to v dash. So, u came to u dash, v came to v dash. What if this is after the first rotation like this? This is fine, okay. This is after the first rotation. Now, see what happens once you go to the second operation. The second operation will do something else. So, it has to rotate about u by an angle theta. So, let us see where it is u. So, here is u.

You have to rotate an angle theta. This time, v goes to v dash and goes to v double dash. w goes to the new location. That is w dot not. It can be written as w double dash also. So, yes, these two will change. So, you are, you have these here. You have these here. That is your new location once you do this, not the third transformation, third rotation. So, that is actually going, this, so it is rotation about w lookup, print by an angle, so w by a sign. This will take it to a new location. Here is your angle: this is your new location. So, this is where we finally reached, so this is all it

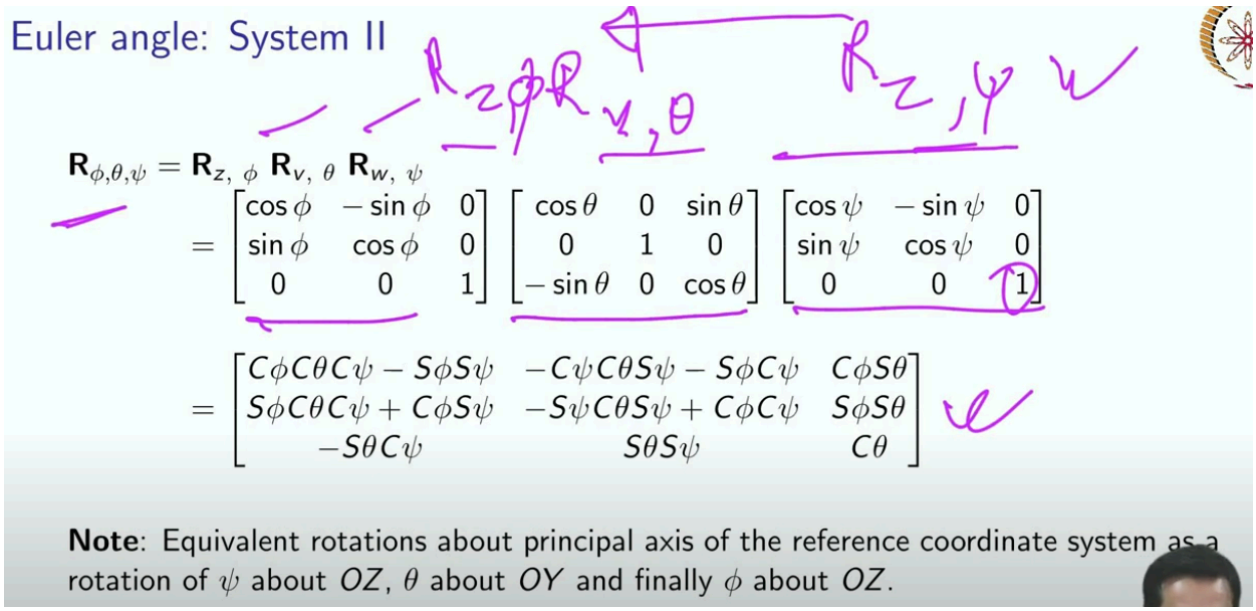
is done. So, rotation transformation is the first transformation, which is here because it is local rotations about the object frame. It has to be post-multiplied. So, it comes here. The next one comes here, your resultant transformation matrix, if you write them all.



So, this is your first transformation, this is your second one, this is your third one. Post. Multiply, you just get this. So, this is your composite transformation matrix to do such a kind of rotation. This is a matrix which can represent three rotations. Right, this can represent the orientation of an object after three consecutive rotations in space. This is equivalent to, if you look otherwise, if you see, if it is rotated about the principal axis, what is it? This is rotation about the z-axis by an angle y. This is rotation about the x-axis by an angle theta. This is rotation about, again, the z-axis by an angle y. but if there are three rotations, if you look at. It is in the universal frame, that is, the principal coordinate axis. It can be done like this. So, the same position, same orientation, can be expressed using principal rotations also principal axis rotations; if you can do this, this and this, you arrive at the same detail, not it. So, that is what. So, this is gyroscopic system one.

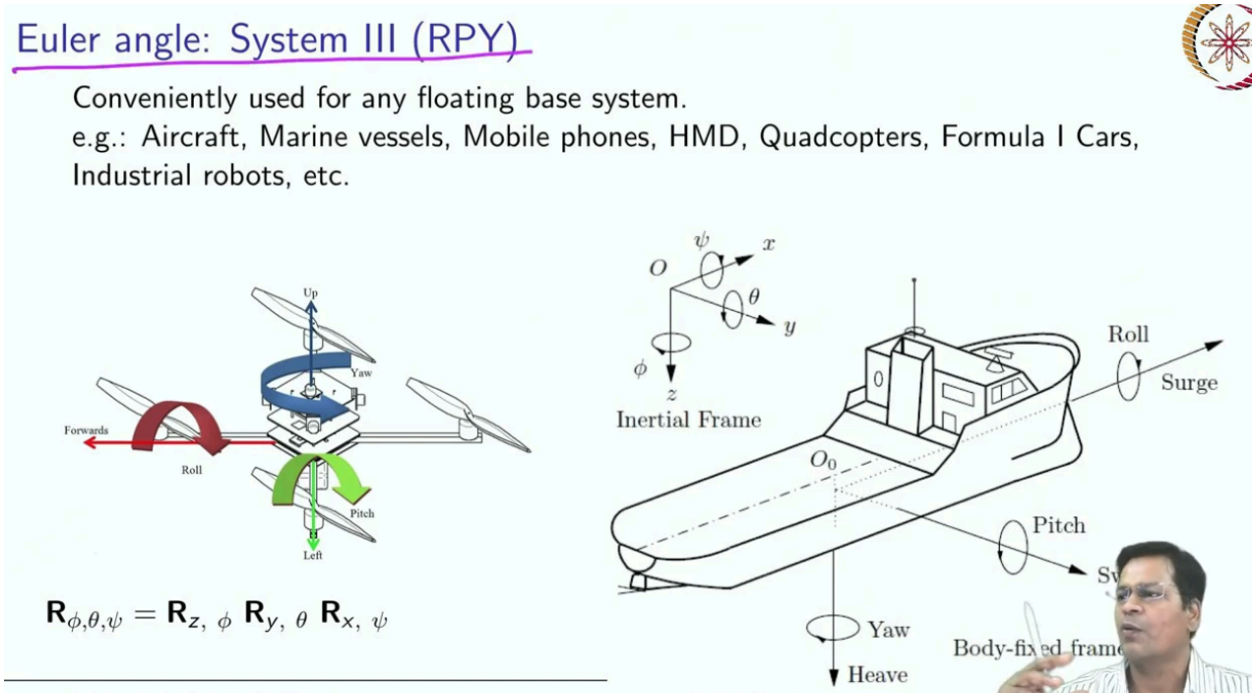


Similar is your gyroscopic system two. Again, you have three consecutive rotations. First is the rotation red axis, this time instead of u. it is v and w again, and this is your figure.



If you write them together, this is your first transformation. Post multiply the second one. Post multiply the third one, as because they are above local axis rotations. And again, if you look at a principal axis rotation, which can take you to the same orientation, it is rotation about the z-axis by an angle, psi, rotation about the y-axis by an angle theta, this is rotation about the z-axis by angle y. okay, so three consecutive rotations are taken in this order because it is principal axis rotation. So, this also takes the object to the same place, or an object with the place can be

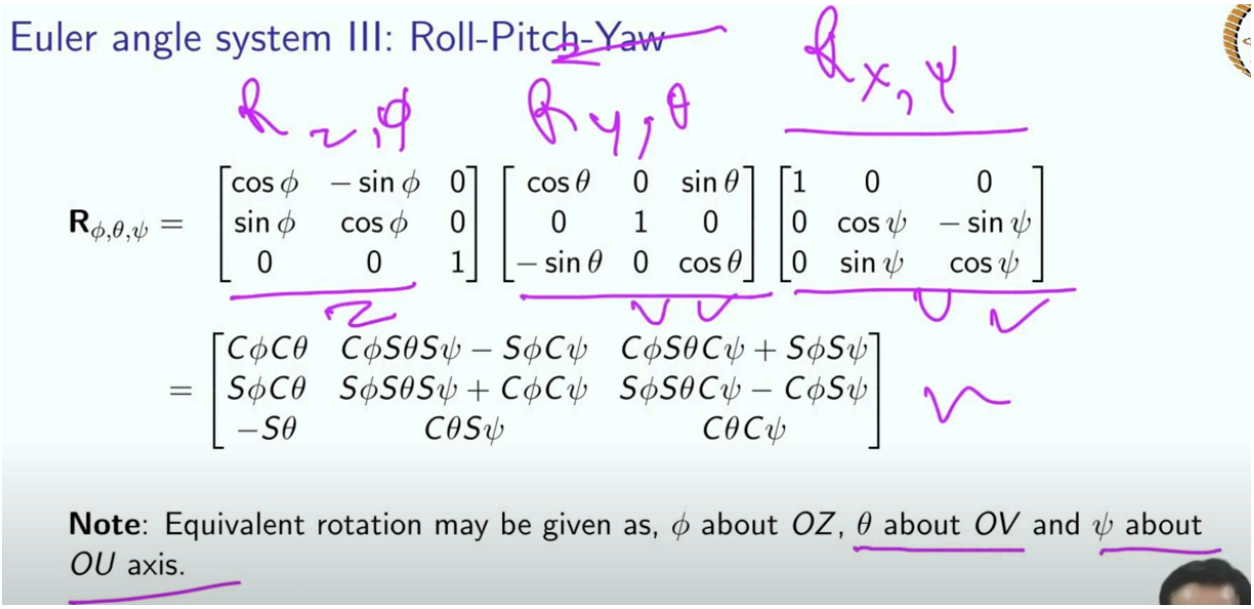
defined using this kind of ruler angle, and you reach a place. Okay, so this is equivalent. Both are equivalent.



Now let us look at the third, which is the most important one, that is most commonly used for any floating object in space. So, any floating base system, like an aircraft, orientation can only be observed clearly, which you can see in your breakdown frame. Where are you sitting? You are sitting in a local frame. So, you can see it is rolling, you can see it is rolling. It can do back-and-forth movement, like pitching, can do pinch, or it can do yaw like it, it can do yaw, same with any ship. It can do roll. That is rolling, rolling, okay, this is, you can go back and forth again pitching, and you can draw. This is also popular in marine vessels to express the orientation of a vessel. High phones, you know you already have an accelerometer to measure the angle. So, about which direction you have to measure, you have to measure about local axis, that is, with respect to earth, that is, with respect to the principal axis. Mobile phones also give you accelerometers and can give you tints. Also, the tilt angle can be calculated based on accelerometer sensors.

So, that can give you angular positions based on this ray, roll, pitch, yaw, and head mount device of a virtual reality system that also can tilt the environment which you are seeing. Quad quarters, that is, drones like this, Formula One cars, industrial robots, so industrial robots. Basically, they use this because it is easy to understand and visualise. That is the reason industrial robots, although the frames are relatively placed, it is easier to express them in a roll pitch yaw system because apprentices who are programming the robot can easily visualise how my robot is oriented if I just say this: three angles, okay, and that is all it is programmed. Internally, the robot may be converting the ray that a programmer is specifying to maybe UV or miller angle system, one or two or three, whatever. So, it internally may use something else, but externally, in order to program, in order to understand this, the new thing. So, this has just three rotations, which are

very, very simple: rotation about the x-axis by an angle, psi, okay, or y theta about z, but pi, theta set, three rotations are there, which is enough to express this. So, rotation about the x-axis, this time you will do the y-axis and z axis, three rotations are there.



So, simply, this can be expressed like this. So, this was the first operation, rotation about x, rotation about y, rotation about z. So, these three rotations we multiplied, and you arrive at this now again. Can it be inverted back? A relative frame of reference? Yes, it can be. So, this is your first rotation. This is rotation about the z-axis. Okay, not this one. So, this is yours. So, this is your first rotation. This is rotation about the z-axis. Then you do post multiplication by this so that it can be written as rotation about theta by v, about v axis, and rotation about psi above UX. Okay, these are the three. The first one is your principal axis, next to our local. So, this is how it can also be converted. What this is? New larynx system of rotation that can represent the orientation of an object in space.

So, yes, that's all for this lecture. So, next class, we'll be starting with robot kinematics DH parameters, and we'll start doing forward kinematics. We'll talk about that in the next class. That's all thanks.