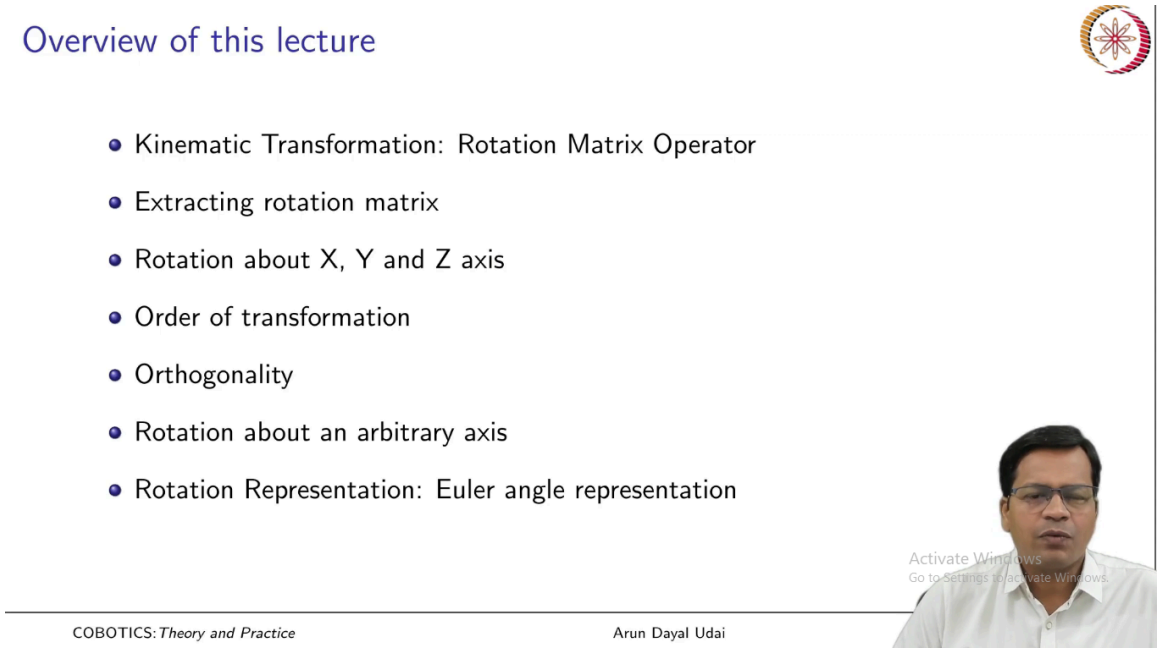
NPTEL Online Certification Courses
COLLABORATIVE ROBOTS (COBOTS): THEORY AND PRACTICE
Dr Arun Dayal Udai
Department of Mechanical Engineering
Indian Institute of Technology (ISM) Dhanbad
Week: 03
Lecture: 12

Rotation Matrix, Arbitrary Axis Rotation, and Euler Angles

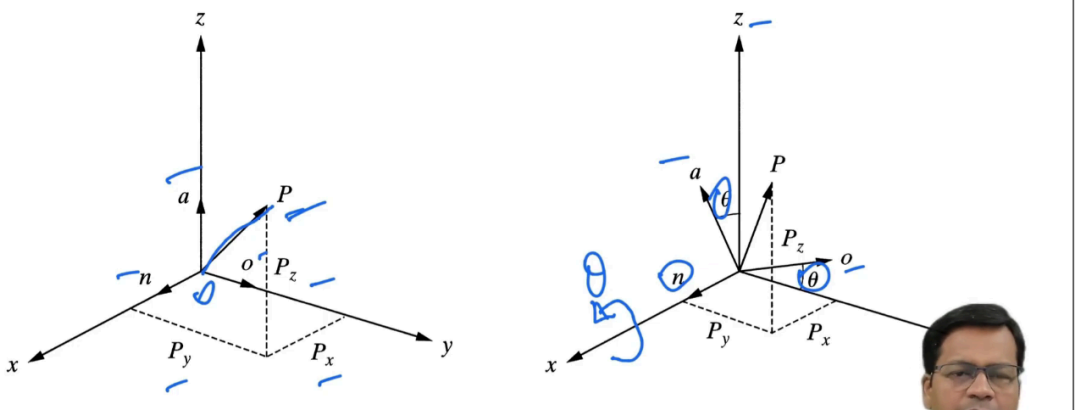
So, welcome back to the course Cobots: Theory and Practice. So, we were in week 3, Transformation Matrices and Robot Kinematics.



So, in this lecture, we will discuss kinematic transformation, especially the rotation matrix operator. We will extract the rotation matrix. We will discuss rotation about the X, Y, and Z axes. We will move ahead with the order of transformation, and what would be the effect of changing the order of transformation. We will discuss what orthogonality is. We will discuss rotation about an arbitrary axis. And finally, we will end with Euler angle representation, which is a kind of rotation representation. So, everything is about rotation in this lecture.

Kinematic Transformation: Rotation Matrix Operator

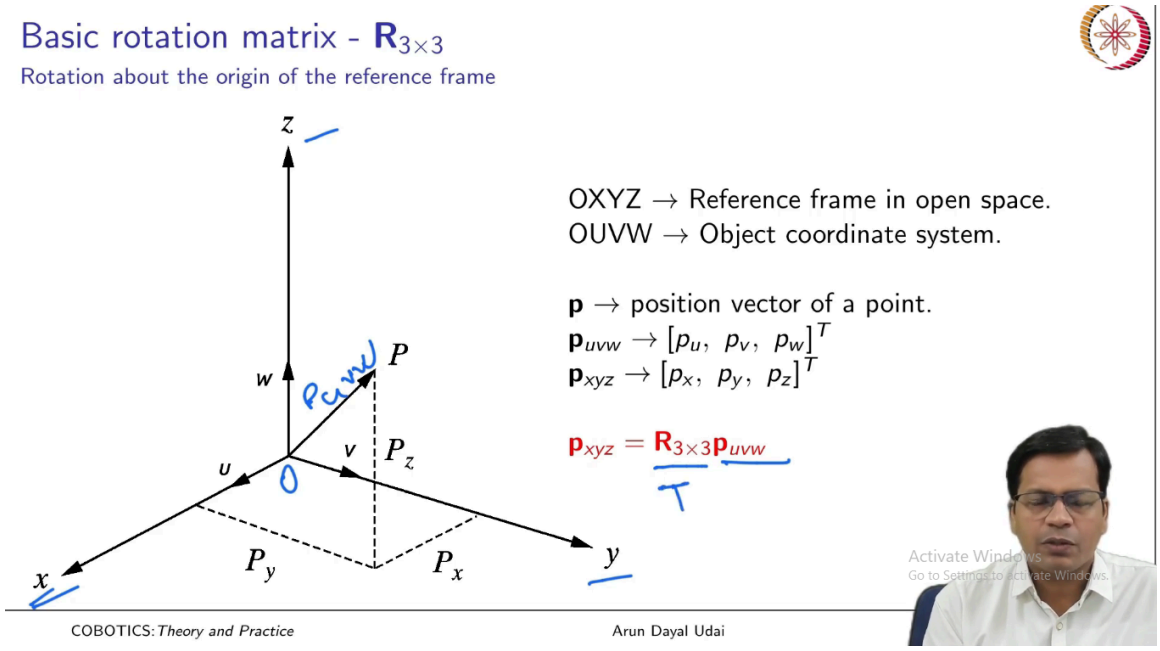
About X axis



Activate Windows
Go to Settings to activate Windows.

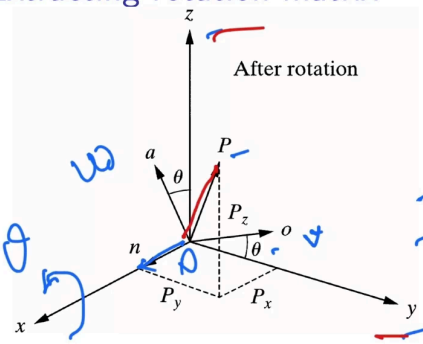
So, let us begin with kinematic transformation, especially the rotation matrix operator. In the last lecture, we discussed the kinematic Translation matrix operator. So, kinematic transformation: the rotation matrix operator.

Let us say there is a point P with position vector OP. O is the origin, and it has three projections along orthogonal axes, that is, Px, Py, and Pz, or you can call it n, o, and a. If it is rotated about the x-axis by a certain angle theta, then n remains where it was, but it comes out of the y-axis and now it is o, and similarly, a comes out of the z-axis by an angle theta again, so it becomes a. So, this is how it will look once it is rotated about the x-axis.



So, we are looking for a transformation matrix that can actually do that. So, let us say there is a vector Puvw that is here; this is Puvw, this is the vector, and I apply a rotation transformation matrix on it, and it gets rotated about the X, Y, or Z. So, rotation about the origin of the fixed reference frame. So, over here, X, Y, and Z are the fixed reference frame, whereas the rotating reference frame is U, V, and W. The point is now represented in the XYZ frame. Now, after rotation, we want the new location of the point to be again represented in the XYZ frame. So, Puvw is the initial vector that needs to be rotated using a transformation operator over here, and it should now be represented in the XYZ frame again, and the rotated position of this should be rotated and represented again in the x, y, z frame.

Extracting rotation matrix



$$\mathbf{p}_{uvw} = p_u \hat{i}_u + p_v \hat{j}_v + p_w \hat{k}_w$$

By definition of scalar product, the components are:

$$p_x = \hat{i}_x \cdot \mathbf{p}_{uvw} = \hat{i}_x \cdot \hat{i}_u p_u + \hat{i}_x \cdot \hat{j}_v p_v + \hat{i}_x \cdot \hat{k}_w p_w$$
$$p_y = \hat{j}_y \cdot \mathbf{p}_{uvw} = \hat{j}_y \cdot \hat{i}_u p_u + \hat{j}_y \cdot \hat{j}_v p_v + \hat{j}_y \cdot \hat{k}_w p_w$$
$$p_z = \hat{k}_z \cdot \mathbf{p}_{uvw} = \hat{k}_z \cdot \hat{i}_u p_u + \hat{k}_z \cdot \hat{j}_v p_v + \hat{k}_z \cdot \hat{k}_w p_w$$

So, now let us begin extracting the rotation matrix, which can actually do that. So, P_{uvw} is the vector that is shown here. It is given by p_{uiu}, p_{vjv}, p_{wkw}. What are i, j, and k? They are orthogonal vectors, unit vectors along the axis of the frame, which is the rotating frame.

Now, By definition of scalar products, the components along x, y, and z, let us say it gets rotated by an angle theta about the x-axis, so it comes to a new location. U remains where it was, and V and W actually come out of the x, y, and z axes, and a new position is attained, and this time, O and A are the vectors that are actually making some angle with the initial y and z axes.

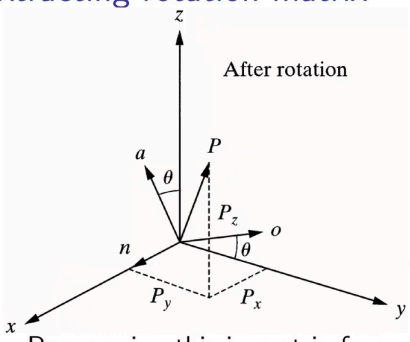
So, again, this point P, this position vector OP, will have projections along x, y, and z after rotation, and those will be given by the projection of Puvw, that is, this vector along x. So, that should be given by P_{uvw} dot ix. I am taking the projection of P_{uvw} along x. So, I am taking the dot product ix dot Puvw. So, this actually gives me the projection along the x-axis. So, this is the first thing that I do to get the Px, the projection of P_{uvw} along x. Similarly, I could get Py by taking the dot product with jy, okay? Similarly, Pz can be obtained by taking the dot product with kz. So, this time, ix, jy, and kz are the unit vectors along the x, y, and z-axis.

So, now this vector ($p_{uiu} + p_{vjv} + p_{wkw}$) I will substitute here, and what I get is ix dot iu into pu.

$$\hat{i}_x \cdot \hat{i}_u p_u$$

pu is the magnitude of p along u. Similarly, ix jv, pv, ix kw, pw, and the same goes along each of the axes x, y, and z.

Extracting rotation matrix



$$\mathbf{p}_{uvw} = p_u\hat{i}_u + p_v\hat{j}_v + p_w\hat{k}_w$$

By definition of scalar product, the components are:

$$p_x = \hat{i}_x \cdot \mathbf{p}_{uvw} = \hat{i}_x \cdot \hat{i}_u p_u + \hat{i}_x \cdot \hat{j}_v p_v + \hat{i}_x \cdot \hat{k}_w p_w$$
$$p_y = \hat{j}_y \cdot \mathbf{p}_{uvw} = \hat{j}_y \cdot \hat{i}_u p_u + \hat{j}_y \cdot \hat{j}_v p_v + \hat{j}_y \cdot \hat{k}_w p_w$$
$$p_z = \hat{k}_z \cdot \mathbf{p}_{uvw} = \hat{k}_z \cdot \hat{i}_u p_u + \hat{k}_z \cdot \hat{j}_v p_v + \hat{k}_z \cdot \hat{k}_w p_w$$

Rearranging this in matrix form as:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} \hat{i}_x \cdot \hat{i}_u & \hat{i}_x \cdot \hat{j}_v & \hat{i}_x \cdot \hat{k}_w \\ \hat{j}_y \cdot \hat{i}_u & \hat{j}_y \cdot \hat{j}_v & \hat{j}_y \cdot \hat{k}_w \\ \hat{k}_z \cdot \hat{i}_u & \hat{k}_z \cdot \hat{j}_v & \hat{k}_z \cdot \hat{k}_w \end{bmatrix}_{3\times3} \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix}$$

$$\mathbf{p}_{xyz} = \mathbf{R}_{3\times3}\mathbf{p}_{uvw}$$

COBOTICS: *Theory and Practice* Arun Dayal Udai

Rearranging this in the matrix form will now give me Px, Py, and Pz, which is the projection of the vector p_{uiu} along x, y, and z after rotation. These were the projections along x, y, and z that existed before rotation and that were also equal to Puvw, Pu, Pv, and Pw. So, what essentially is this? This is the desired rotation transformation matrix after rearranging this that I got is actually doing what I intended to do. This is, this becomes the operator that is applied to p_{uiu} and gives me the rotated point p about the x-axis by an angle theta, and the same can happen. It can be a rotation about any axis because this is a general transformation. I made it explicitly along the x-axis so as to explain it, but yes, This can be a combined rotation about the x, y, and z axes, anything. So, this becomes the general rotation matrix, which is an operator that is applied to Puvw to give you the new projection of the vector p okay p_{uiu}.

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} \hat{i}_x \cdot \hat{i}_u & \hat{i}_x \cdot \hat{j}_v & \hat{i}_x \cdot \hat{k}_w \\ \hat{j}_y \cdot \hat{i}_u & \hat{j}_y \cdot \hat{j}_v & \hat{j}_y \cdot \hat{k}_w \\ \hat{k}_z \cdot \hat{i}_u & \hat{k}_z \cdot \hat{j}_v & \hat{k}_z \cdot \hat{k}_w \end{bmatrix}_{3\times3} \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix}$$

This is the vector after rotation; it gave me the projections along x, y, and z, so effectively, this is the new vector after rotation.

$$\begin{bmatrix} \hat{i}_x \cdot \hat{i}_u & \hat{i}_x \cdot \hat{j}_v & \hat{i}_x \cdot \hat{k}_w \\ \hat{j}_y \cdot \hat{i}_u & \hat{j}_y \cdot \hat{j}_v & \hat{j}_y \cdot \hat{k}_w \\ \hat{k}_z \cdot \hat{i}_u & \hat{k}_z \cdot \hat{j}_v & \hat{k}_z \cdot \hat{k}_w \end{bmatrix}_{3\times 3}$$

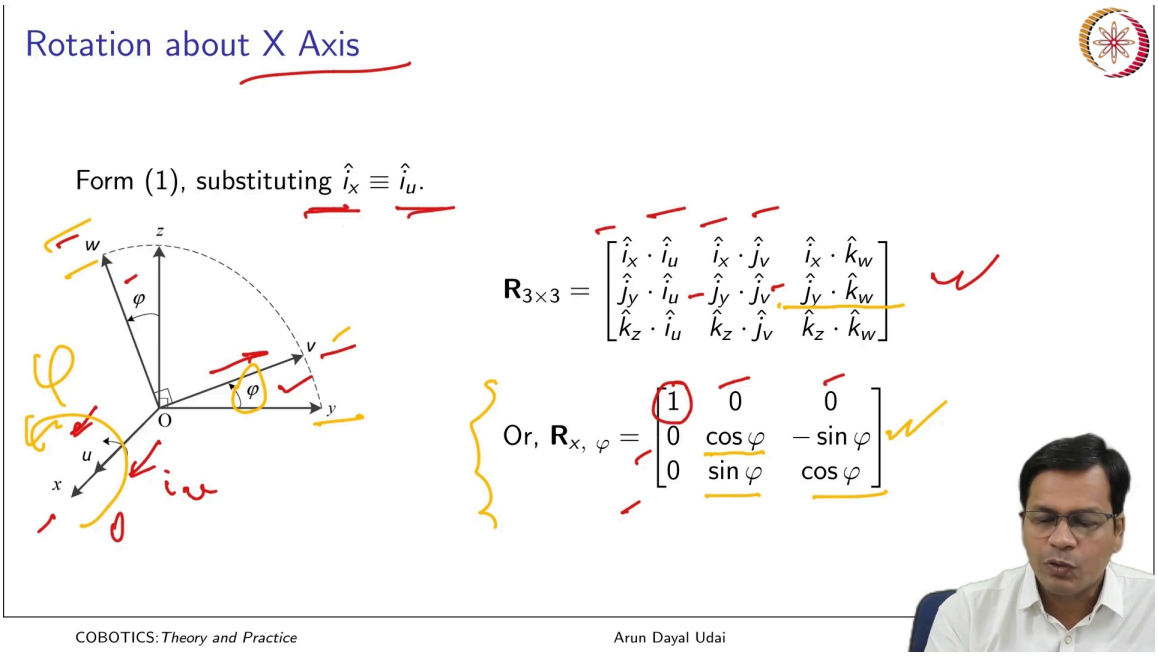
So, now this becomes my rotation operator;

$$\begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix}$$

This is my initial point before rotation,

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

And this is the point after rotation.



Rotation about the X-Axis. How to obtain that? I want to rotate about the x-axis by an

angle psi. So, what I will do here, you see, ix is not changing. ix remains aligned with iu. So, iu remains wherever it was earlier before rotation. What changes is v and w, that goes off by an angle psi.

So, now, if I substitute ix is equal to iu in this, this is the general rotation matrix. So, the angle between ix and iu is 0 degrees. So, those two are unit vectors. So, ix dot iu will give you 1, and the angle between them is 0 degrees.

Now, let us look at Ix and Jv, Ix is here, Jv is here, and the angle between them is 90 degrees, which should give me 0, 0, 0, and 0. Now, let us look at Jy and Jv. You see, there is an angle that is psi over here. So, Jy and Jv, if you take the dot product, should give you the cosine of psi. Similarly, jy and kw. This one, jy and kw, so there is the angle here, jy and kw, so 90 degrees plus psi, which should give you minus sine psi. Similarly, this is sine psi, and again, cos psi. What we obtained here is the rotation matrix operator, which will rotate the frame and bring it to a new location, that is, rotation about the x-axis by an angle psi, and it will take it to this new position. So, this way, you can apply it to a point, and that point can get rotated by an angle psi about the x-axis. So, this is the rotation matrix about the x-axis.

Rotation about Y and Z axis

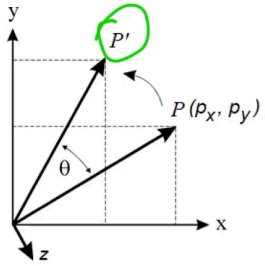
The rotation matrices for Y may be derived as:

$$\mathbf{R}_{y,\,\phi} = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}_{3\times 3}$$

The rotation matrices for Z may be derived as:

$$\mathbf{R}_{z,\,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}_{3\times 3}$$

$$\equiv \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{4\times 4}$$



$$\mathbf{R}_{z,\,\theta}\mathbf{p} = \mathbf{p}'$$

$$\underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Rotation Matrix}} \underbrace{\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}}_{\text{Point P}} = \underbrace{\begin{bmatrix} p_x\cos\theta - p_y\sin\theta \\ p_x\sin\theta + p_y\cos\theta \\ p_z \\ 1 \end{bmatrix}}_{\text{Rotated Point}}$$

Similarly, in order to rotate about the y-axis, you can substitute; you go back and see what you can substitute here. Next time, you have to substitute Jy is equivalent to Jv. If

you substitute this again here and apply a similar technique, you just check their angle should be 0 degrees, 90 degrees, or the required angle, the angle of rotation. So, over here, it is phi, that angle is phi, and that can now be seen as this. So, this becomes my new rotation matrix which, when applied to Puvw, can give me a point in the XYZ frame that is a rotated point by an angle phi about the axis Y.

Similarly, The rotation matrix for the z-axis may be obtained like this, and this time, it is rotation about the z-axis by an angle theta. Again, the angles between the vectors would be 0 degrees, 90 degrees, or angle theta. The substitution could be Kw is equivalent to Kz. So, that should be substituted into this (R_{3x3}) equation again, and you get to the rotation matrix about the z-axis by an angle theta. Rz theta, when applied to Puvw, gives me a point Pxyz. So, this was my initial position of the point. So, that point is now rotated about the z-axis by an angle theta and gives me the new point. So, this is what it means.

In terms of a 4x4 homogeneous transformation matrix, it can be written like this. So, the whole of this 3x3 matrix comes to this location without any translation part over here. Okay, this is pure rotation. So, this is the physical meaning of that. So, p, when applied with the rotation matrix operator, goes to a new position p dash. That is rotation about the z-axis by an angle theta.

So, now you see this is the initial point applied with 4x4. This transformation matrix gives me the rotated point, which is this. So, this is the new location of point P in the initial frame x, y, z frame. Got it? This can also be obtained using a geometrical method.

Composite rotation matrix

Sequence of finite rotations are important!

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

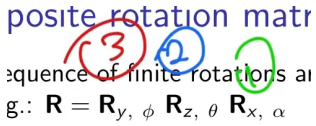
$$= \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha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

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

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).

So, now the composite rotation matrix. Let us say there are multiple rotations happening one after the other about the x-axis, y-axis, z-axis, or in a different order.

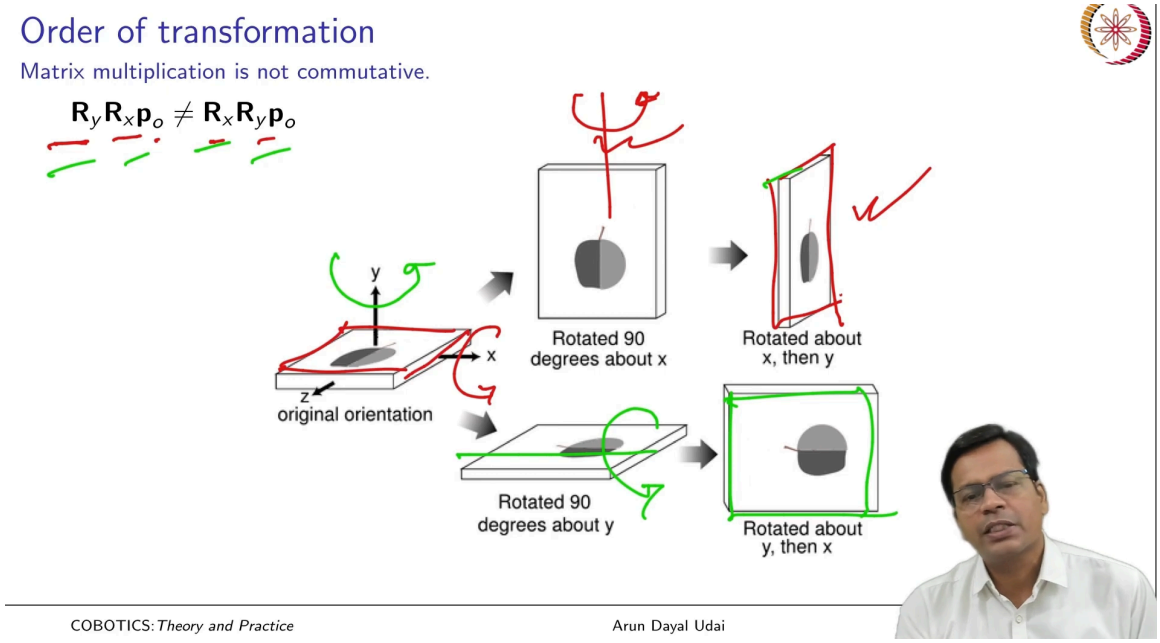


So, that can go like this. So, a finite sequence of rotations is arranged in a way where the first rotation comes first, and the second rotation will come here, the third rotation will go here, and they are pre-multiplied. Okay? Because let's say this is the point. You applied a transformation matrix one here. This gives you a new point, P dash. This is applied with the next rotation matrix, and this gives you a new point, P double dash, and again this is applied with a third rotation matrix, and this takes you to the final point, P triple dash. Got it?

So, every multiplication is an operator on the previous result. So, that is why it is one after the other applied about the same reference frame. So, that is the reason it is pre-multiplied one after the other. So, the order goes like 1, 2, and 3. This time, it is Rx alpha, rotation about the x-axis by an angle alpha, that comes here, rotation about the z-axis by an angle theta, rotation about the y-axis by an angle phi. So that comes here. If

you multiply that, you get to this. Okay, so you can try changing the order. You should get a different result. We'll discuss that.

So, in order to do such calculations, you can easily use the symbolic computation toolbox, which is there in MATLAB also, or you can use MuPAD, okay. Wolfram Mathematica you can use, which is for Windows, Raspberry Pi, or Android platforms. wxMaxima, which is there in Ubuntu also. You can use any of these symbolic computation tools in order to do this kind of complex multiplication. Need not do it by hand. So you can try changing the order.



So, what you see here by changing the order, let's say you applied Rx first and then Ry on Po. So what would happen here?

Let's say this is the initial orientation of this book. The rotation matrix is all about orientation changes. First, you rotate about the x-axis by an angle using the right-hand thumb rule. You should be doing this. Okay, it comes to this.

Next, it is rotated about the y-axis by 90 degrees. So, this should be here. Again, it goes like this. By an angle of 90 degrees, the final orientation is something like this.

Now, if I had changed the order, I did y first, then x, what would happen? So, the first rotation about the y-axis, y angle 90 degrees, it comes to this. The next rotation is a

rotation about the x-axis. So, I will do a rotation about the x-axis, y angle 90 degrees, and it comes to a new location, which is like this. So, you see by changing the order of rotation, the operator, what you got, you got two different orientations. That means rotation matrices are not commutative. In general, Matrix multiplication is not commutative, and changing the order will take you to a different state.

Orthogonality

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

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.

COBOTICS: *Theory and Practice* Arun Dayal Udai

Now, let us recall our matrix, which looks like this. So, the general rotation matrix applied over the P_{uvw} point, which was the initial position vector, gave me a P_{xyz} point, which is a rotated point. So, or a position vector or a vector. So, this is my initial point: this is the rotated point about a certain axis, whatever is told by this rotation operator.

So, now let us assume there is an operator, which is the inverse of this. So, let us take it as Q. So, what is Q? It takes you otherwise. Actually, Puvw was here, Pxyz is here. At this time, it is inverse, so it is Pxyz here and Puvw here. Essentially, this should be equal to the R inverse of this. When I took the inverse, I got to this. So, if you take this, calculate this, you should be getting this. I just rearranged this, I got this, and as the dot product is commutative, you can rearrange. Okay, you can flip these values which are here. Okay, and what I found here is this, essentially, this matrix is R transpose. So, what I found here, I started by taking the inverse of the rotation matrix. I just rearranged it, assuming the dot product to be commutative, and I got to R transpose.

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

That means Q is equal to R inverse, which is equal to R transpose. So, what is it? This is basically R. So, R is a matrix which is an orthogonal matrix. For an orthogonal matrix, R inverse is equal to R transpose. That is what is the meaning of this. So, what we have found here is the rotation matrix is an orthogonal matrix.

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.

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.

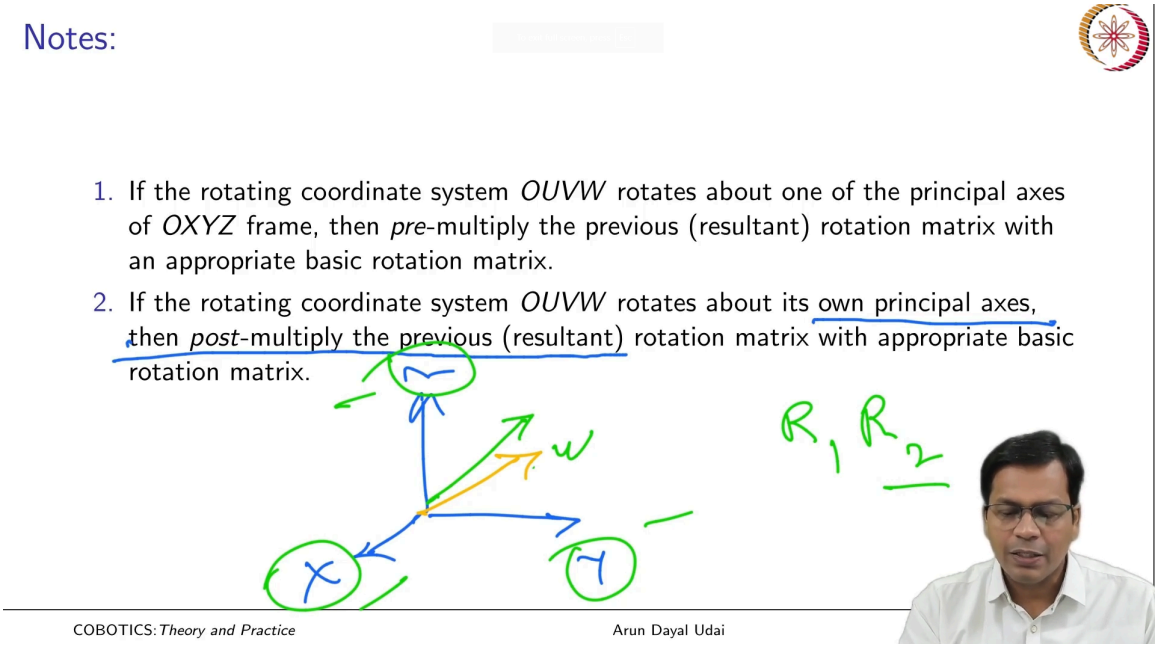
$$P_{xyz} = R_2 R_1 P_{uvw}$$

$$R_2 [P_{uvw}]$$

$$R_2 R_1$$

So, please note these few points. If the rotating coordinate system OUVW rotates about one of the principal axes of the OXYZ frame, then pre-multiply the previous resultant rotation matrix with the appropriate basic rotation matrix. What does it mean, basically? Again, I will come back to this. So, if this was my initial point Puvw, I applied a rotation matrix. It gave me Pxyz. So, this was the meaning. If there is a second rotation, which is again about the base reference frame, that is, X, Y, and Z, again you are doing rotation. So, if this is the first rotation matrix, the second one will come here. Why is it so? As I have explained earlier, the second rotation is happening on top of the result of the first rotation. Whatever this gives me, a new point Puvw dash, so rotation two is happening on top of this. So, it has to come before. If I open up the bracket, it comes as R1, R2, like that. So, it says you have to pre-multiply the previous rotation matrix with the appropriate basic rotation matrix. R1 is the first one, and R2 is the second one.

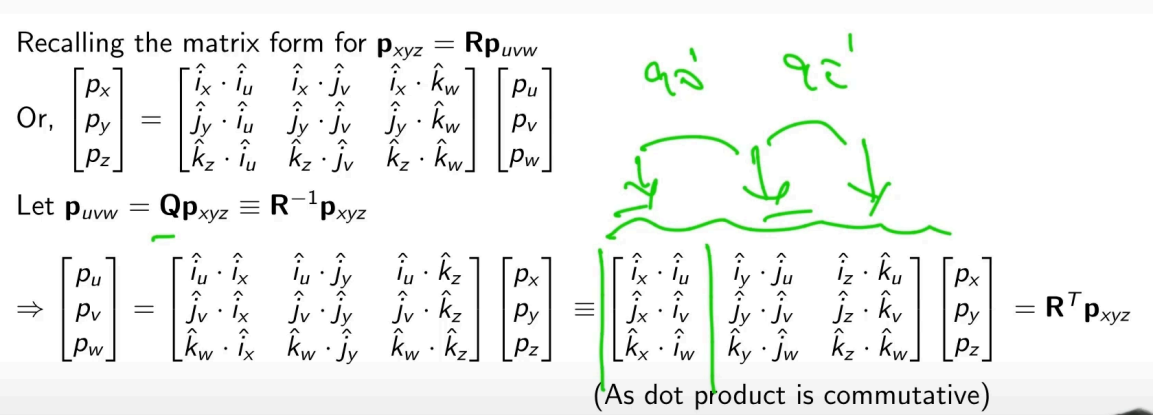
Notes:



Now, if the rotating coordinate system rotates about its own principal axis, then post-multiply the previous resultant rotation matrix with the appropriate basic rotation matrix. Now, this time, what does it mean? Let's say this is your initial frame X, Y, and Z. This was your point. Now you have rotated the point from here, and it went to a new location here. The second rotation is not about the base rotation matrix. This time, it is with respect to the rotated reference frame. Wherever it is standing, it has rotated. It came to a new point. From there, you did another rotation. From this location, you did another rotation, not again with respect to the principal frames. You did a rotation with respect to the new frame after the rotation. The first rotation has happened. From there, you have rotated again by a certain angle. So, this time it would be a post-multiplication. So, rotation one and you post-multiply, and that becomes your second rotation. In this case, it is in a relative reference frame, not in an absolute reference frame.

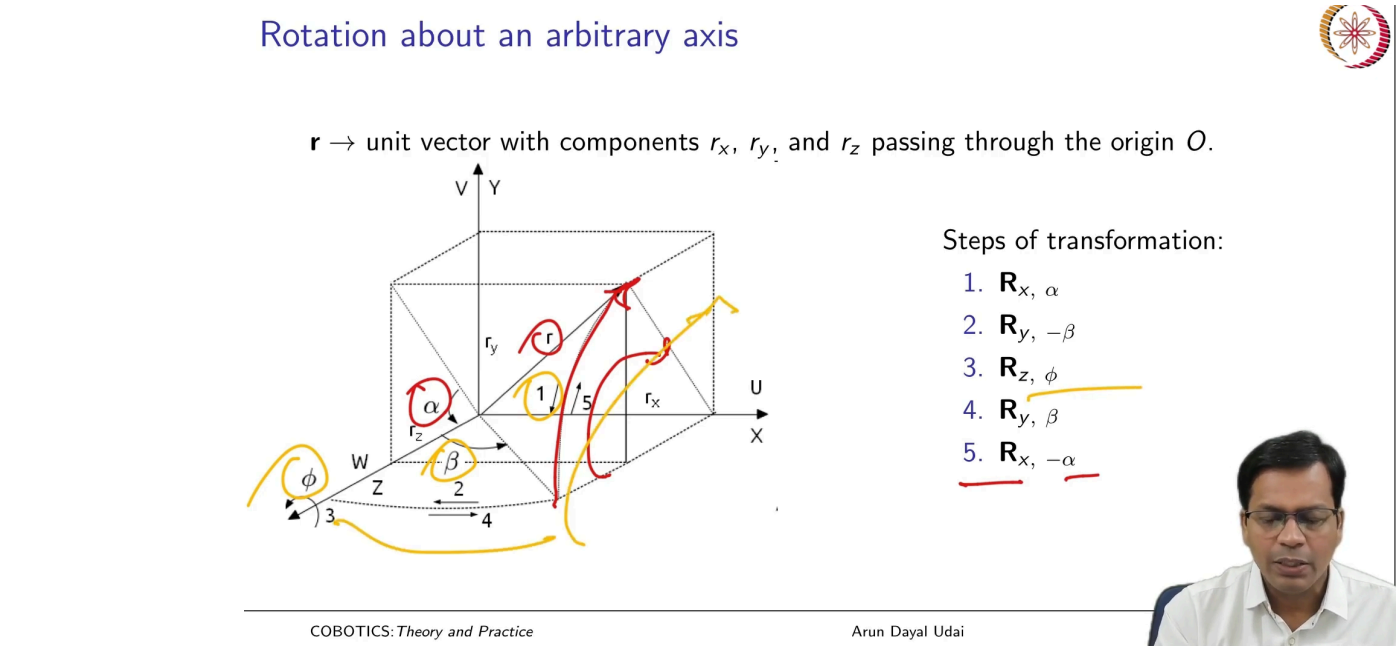
Now, this would happen even in the case of a 4x4 matrix. So, irrespective of 3x3, and 3x3 in these two cases, it could also be in 4x4. That means the 3x3 sub-matrix can remain here, and translation will go here, and these will be 0. So, using 4x4 also, these two rules apply.

The inverse of a rotation matrix is equivalent to its transpose. We have seen that the rotation matrix is an orthogonal matrix. There are still many other rules that you can notice while we discuss the rotation matrix.



Coming back to that, let us say you just take up any rotation matrix. So, all these elements are basically components of the same vector along x, along y, along z, and that vector was a unit vector. So, each of these columns represents a unit vector. This is a unit vector, this is a unit vector, and, moreover, The next to next rows. The angle between this row and this row is 90 degrees. This is 90 degrees. So, all the consecutive rows are also orthogonal. This is orthogonal to this. This is orthogonal to this. And this is orthogonal to this. So, one by one, they are all orthogonal to each other, and each column is a unit vector.

Rotation about an arbitrary axis



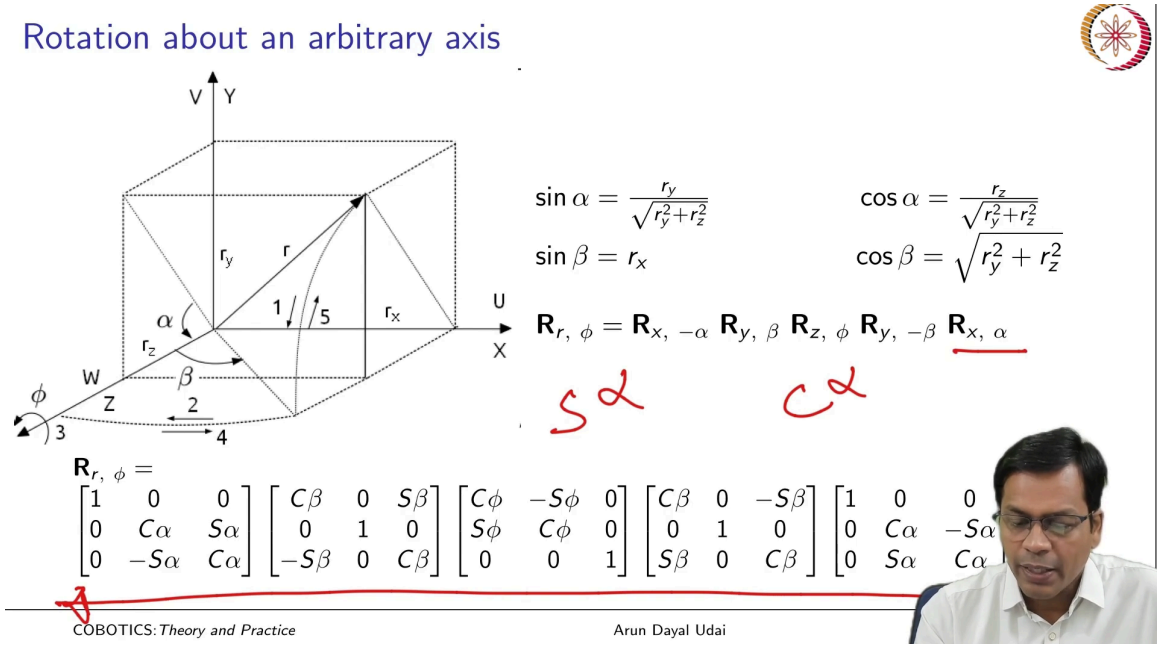
Now, let us see how these rotations that we did are rotations about the base frame. That is rotation about the base x-axis, y-axis, or z-axis. Let us say I have to rotate about any arbitrary axis in space, not about x, y, or z. How do I go about it?

So, this is the method for this. So, rotation about an arbitrary axis, I will do that now. I will find out the rotation matrix that can actually do this. So, in order to do that, What we tend to do here is r is the unit vector with components rx, ry, rz. So, let us say this is the axis about which I want to rotate, okay? I want to rotate, and this passes through origin O, okay? About this, I want to do the rotation. By a certain angle, okay? How to do that? We just derived rotation about the x-axis, rotation about the y-axis, or rotation about the z-axis. This has a component; this r vector has components rx, ry, and rz along the principal axes, that is, x, y, and z, okay? And this vector passes through the origin.

The steps would be: we will try to align r, after consecutive rotations, to one of the principal axes. In this case, I choose to be along the z-axis. So, I will do in steps a few rotations so that r gets aligned to the z-axis here, and I will do the rotation and bring it back to the original state. So, what I will do here in the first step I will rotate about the x-axis by an angle alpha so as to bring this r vector in the z-x plane. So, this comes here after rotating like this. So, this is my first step. Hope you could follow this.

So, now again there is a second rotation. This time, I have to rotate about the y-axis by an angle minus beta. Minus or plus is governed by the thumb rule, that is, the right-hand thumb rule. So, this time, I have to rotate about the y-axis by an angle minus beta, okay? This side is positive; this side is negative. So, you see, by doing negative beta, I can bring the R, which was in the first step, came to the z-x plane. Now, it should get aligned with this. So, now R finally comes along the z-axis.

Now, I will do the desired rotation, which is rotation about the z-axis by an angle phi. So, this is the desired rotation. So, what I did was I brought R from here to here and finally to here and did the rotation. Now, I have to bring it back to the original location because what I have is a rotation matrix about x, about y, and about z. I do not have a rotation matrix to rotate about an arbitrary axis or any other axis. So I did it, and then I'll bring it back. So now what I'll do is initially what I did: I rotated about y by minus beta. This time, I'll bring it back by an angle beta. So, rotation about y by an angle beta, I will come here, and I have to go back to this location, okay, where I was. So, in order to do that, I'll do rotation about the x-axis by an angle minus alpha this time, and I come back to the initial location R. So, this was done in 5 steps: rotation about x by an angle alpha. Rotation about y by an angle minus beta, doing the final operation, the actual desired operation that comes here, and bringing it back to the initial location. So, these are the five operations. Mind it, all the rotations were happening in the principal axis. So, it has to be pre-multiplied.



So, Let me first find out the angles also. Using simple trigonometry and projection, I can obtain the angles also. r was the initial position vector about which I wanted to rotate. So, this is r. So, components along x, y, and z will be rx, ry, and rz. So, r is known, so rx, ry, and rz are known. I have to find out angles alpha and beta, cosine and sine because you know the rotation matrix has components of sine and cosines only, all the elements.

$$sin\alpha = \frac{r_y}{\sqrt{r_y^2+r_z^2}}$$

So, sine alpha will be ry by this root over ry square plus rz square, and you can find it out. So, this is your ry, okay? That is what you know, and that becomes this essentially, and how much is this? This one is ry square plus rz square.

So, this is ry, and this was rz. So, ry square plus rz square will give me this. So, that is it. So, alpha is obtained. Similarly, cosine alpha, sine beta, and cosine beta are obtained.

The magnitude of r is unity. That gives rise to, let us say, in the denominator it comes as rx square plus ry square plus rz square. So, it should give me one because it is a unit vector.

$$\mathbf{R}_{r,\,\phi} = \mathbf{R}_{x,\,-\alpha}\,\mathbf{R}_{y,\,\beta}\,\mathbf{R}_{z,\,\phi}\,\mathbf{R}_{y,\,-\beta}\,\mathbf{R}_{x,\,\alpha}$$

So, now, all the operations can be written in steps like this. This was first, this was second, this was third, this was fourth, this was fifth. So, the order would be like this because all the rotations were with respect to the global frame.

So, now, if I put them all here. So, cosine alpha, in short, I have written it as cos alpha; similarly, sin alpha, I have written it as s alpha, and they can be put in this order. So, this is rotation about x by alpha, rotation about x: cos, minus sin, sin, cos; similarly, rotation about y by minus beta. All the matrices are arranged this way.

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.

Upon taking the product of them, I got to this. So, where V(phi), it is versine phi, is equal to 1 minus cos(phi). So, wherever you find V(phi), it is actually equal to this. So, this is a combined rotation matrix that can directly be multiplied to the initial point so as to rotate about an arbitrary axis given by rx, ry, rz, that is, the r vector, by an angle phi. So, P can now be transformed using this rotation matrix.

So, there are many other methods to do such tasks, a few of which are Quaternions, Rodrigues matrix, and direction cosine. So, other methods can also be used to do this. But

this is one of the simplest ones which you have just covered because it is very simple and we had enough background to come to this result. So, I have used this.

Rotation Representation: 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

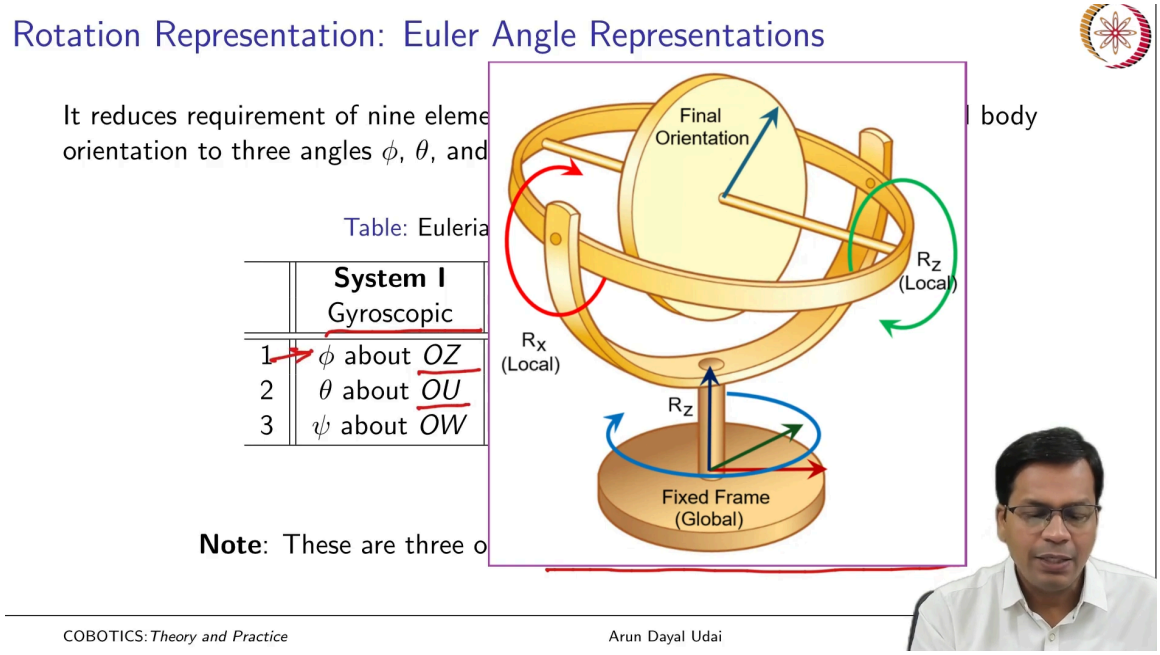
Note: These are three out of 24 different combination possible.

Now, Rotation Representation. Let us say your robot's end effector is in a state. I want to represent how it is. So, there are multiple ways to reach the same points. You have seen maybe a sequence of rotations rx, ry, and rz can lead you there. There can be relative transformations also. That is what we will talk about here. So, how to represent such rotations?

So, Rotation Representations are generally done using Euler Angle Representation. It reduces the requirement of 9 elements of the rotation matrix for specifying rigid body orientations to three angles: phi (ϕ), theta (Θ), and psi (Ψ). So, using any three rotations, you can get to any particular orientation. So, there are standard techniques to do that.

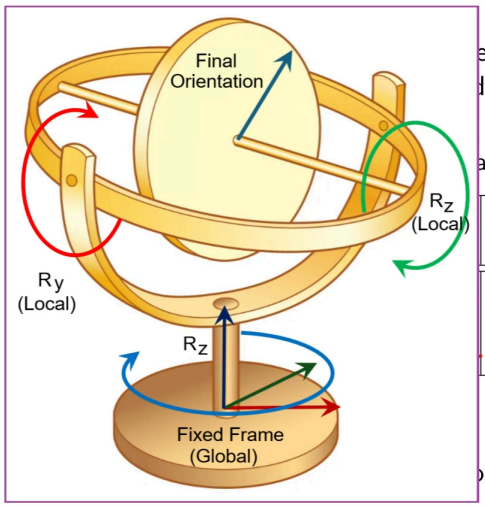
There are a total of 24 such combinations that are possible using absolute rotations, that is, with respect to the global reference frame and combinations of global and local reference frames. What does it mean? Basically, there is an object in space. You have rotated about the global frame about the x-axis, about the y-axis, about the z-axis, and you reach a certain orientation. You could have done rotation about the x-axis and then rotation about the current reference frame; wherever it is now, from there, you rotate again, and from the new position, you rotate once more. That means the first was with

respect to the global frame and the remaining with respect to the relative frame wherever it is from there, you rotated.



So, there is a reference frame which is known as the gyroscopic system. One tells rotation about Oz, that is, the global frame by an angle phi, then rotation about the local x-axis, that is, the object frame by an angle theta. Again, rotation about the object frame UVW is the object frame. XYZ is the global frame. So, u and w finally Psi angle about the W frame. Local Z axis that is W. Okay. So, global, local, local.

Rotation Representation: Euler Angle Representations

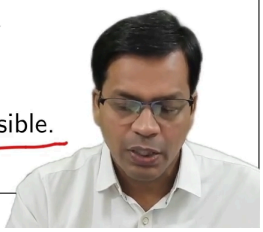


ents of rotation matrix for specifying rigid body
ψ.

an angle sequence of rotations

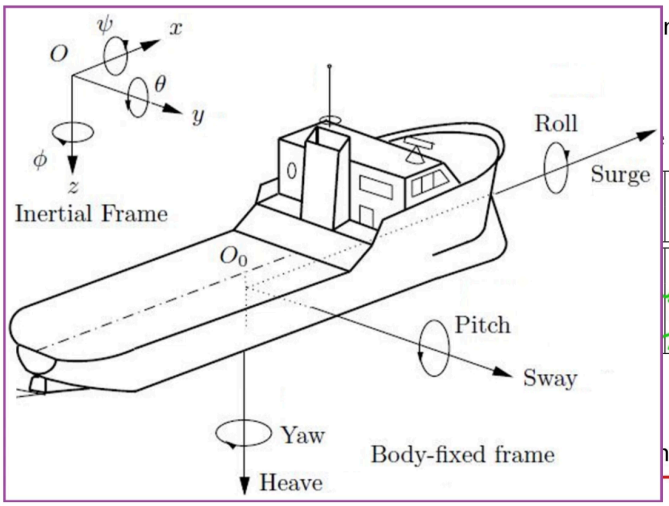
System II Z, V, W	System III Roll, Pitch, Yaw
ϕ about OZ	ψ about OX
θ about OV	θ about OY
ψ about OW	ϕ about OZ

ut of 24 different combination possible.



The next one is again a rotation about phi, about Z, theta, and V that is local, local Y and local Z. This was local X and Z. Okay. This is a gyroscopic system 2- z, v, w.

Rotation Representation: Euler Angle Representations

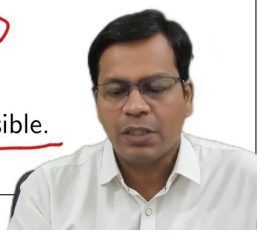


matrix for specifying rigid body

of rotations

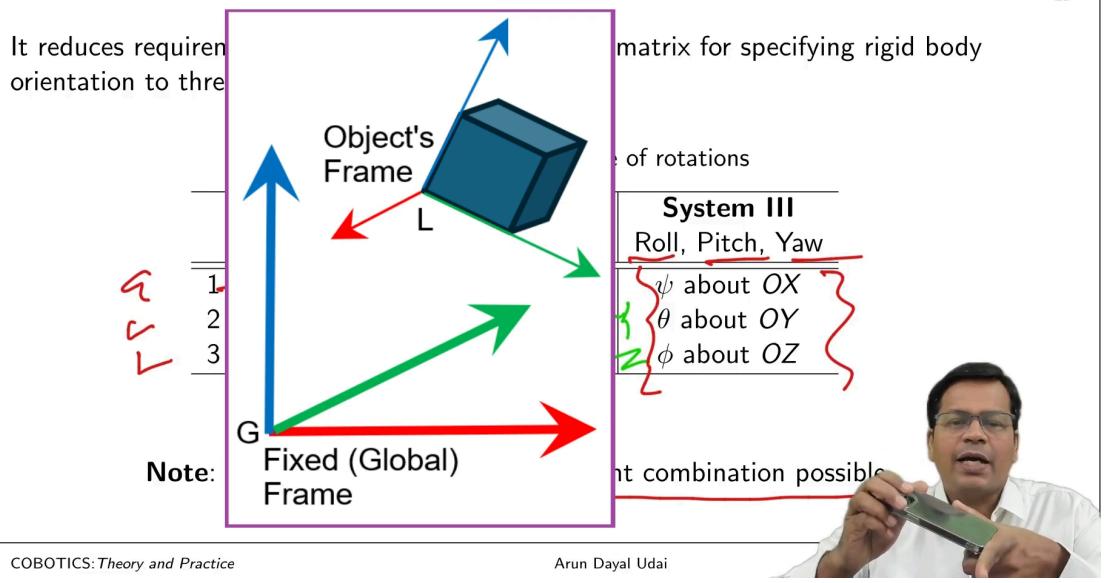
System III Roll, Pitch, Yaw
ψ about OX
θ about OY
ϕ about OZ

t combination possible.



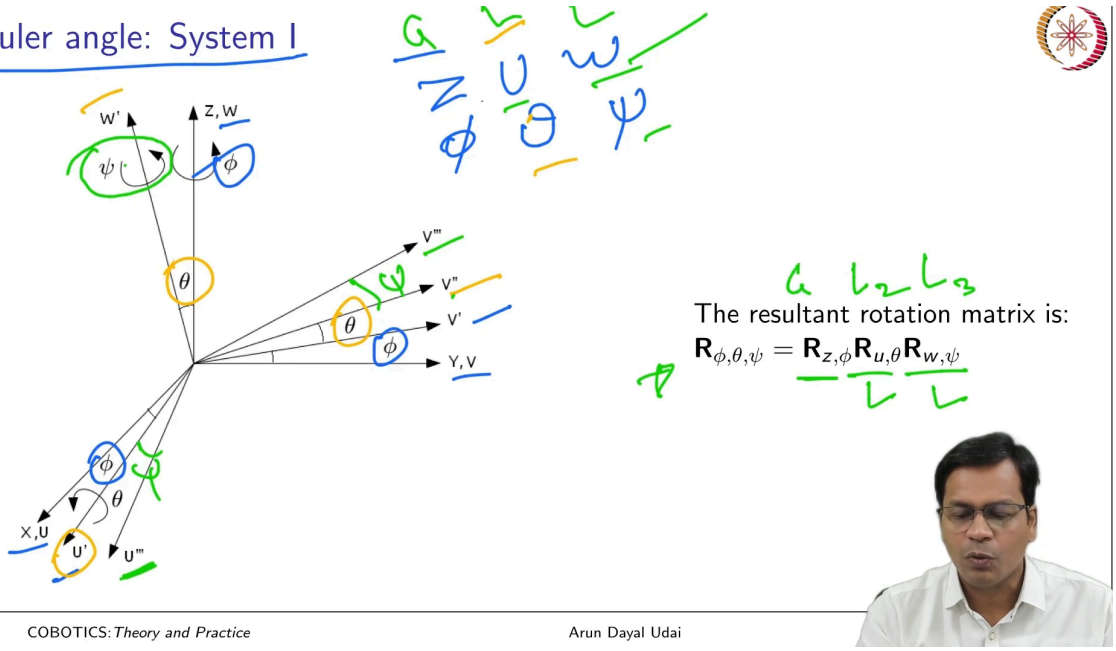
The third one is the simplest one that is most widely used also because of its simplicity and ease to understand. It is rotation about the x-axis, about the y-axis, about the z-axis that is roll, pitch, and yaw rotations. That is all three were about the global frame, not about the object frame.

Rotation Representation: Euler Angle Representations



Object frame basically means there is a frame that is attached to the object. The object in its own frame can rotate about its x-axis, about the y-axis, like that. Okay, so wherever it goes, when you do the first rotation, all the axes change. So from there, about the current x-axis or y-axis, it has further rotated. So, that is what is the local rotation.

Euler angle: System I

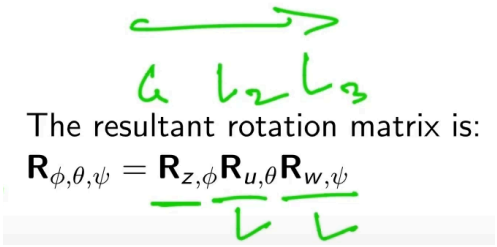


Now, let us talk about Euler angle system 1. What was this? It was z, u, and w by an angle phi, theta, and psi. So, z, u, and w by angles phi, theta, and psi.

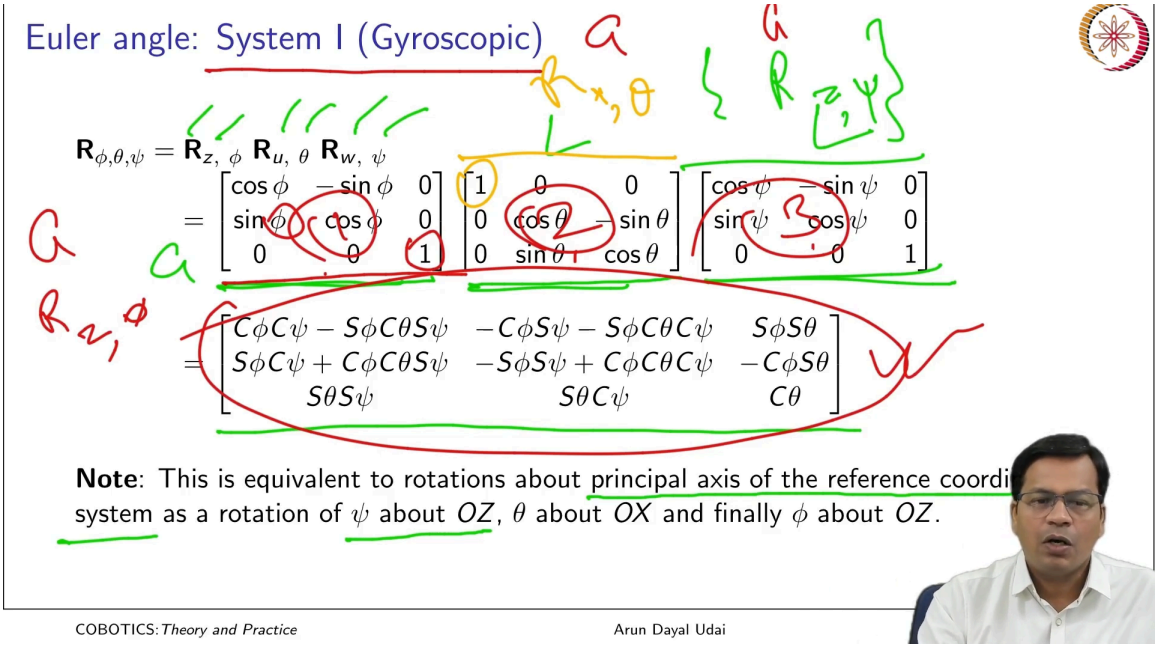
Now, let us see how it works. So, initially, u, v, and w were aligned with x, y, and z. After the first rotation, that is, rotation about z by an angle phi, you see u comes to u dash, v goes to v dash by an angle phi, and w remains wherever it was.

In the second operation, I have to do rotation about u, that is, the local frame by an angle theta. So, where is my u? u is here. So, rotation about u. So, the u dash will remain wherever it is because the rotation is about this axis itself. So, v goes to v double dash by an angle theta. w comes to w dash by an angle theta. So, the new locations are this.

Now, rotation about W by an angle psi. So, this time, rotate about W by an angle psi. So, W will remain wherever it is, and V comes to V triple dash. U comes to U triple dash, okay. These two basically get changed, so this is my angle, this is my angle. So, this is how, after three consecutive rotations, one global, two local, okay? That is the local reference frame. It has rotated, and it has come to an orientation.



So, the final orientation, which is the outcome of three rotations, is given as this. The final rotation matrix would be like this. You see, because the first one happened here, and the next two were local, local. So, effectively, those two are post-multiplied, not pre-multiplied. Second local transformation, third local transformation. The first one was global. So, the order is like this. It goes like this.



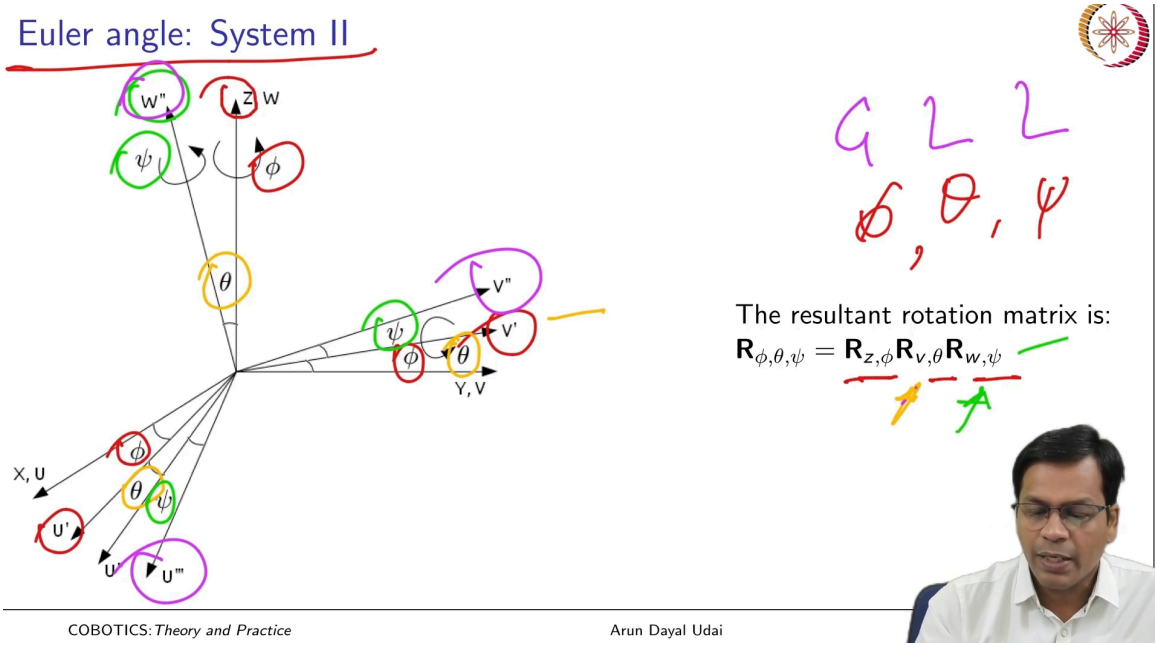
Suppose I put all the matrices here. So, This is the first global transformation, then the next local and local, phi, theta, and psi, z, u, and w, okay? z, u, and w mind it; z and w become different. Initially, they were aligned, but after even a single transformation about any of the axes, w comes out of z, and now z becomes the local frame. So, the order is this, this, and this. That is global, local, and local. This gives me this.

Irrespective of this, the same orientation can be achieved by a different approach. It could be that this is equivalent to a rotation about the principal axis of the reference coordinate system as the rotation of phi about z. So, this looks like rotation about the z-axis by an angle phi. Is it not? This is the same matrix.

The next one is rotation about the x-axis by an angle theta, and the third one is rotation about the z-axis by an angle phi. So, you see, it is in reverse order: global, global, and global. So, if I could rotate about global, global, and global and reach the same orientation, I could have easily done this. But in this case, the gyroscopic system is global, local, and local.

The second and third transformations. So, there are at least over here you have noticed there are at least two ways to reach the same orientation, which is given by this. So, this operator, when applied to a point, can give me rotation about z, u, and w. It can take you

to a particular orientation, or you can represent the orientation of the object using this. The orientation would be given by this matrix.



Similar is system 2. This time it is rotation about z, v, and w by angles phi, theta, and psi. You can work it out quite easily here. So, initially it is rotation about the z-axis by an angle phi. So, phi here and phi here are visible. This goes here. The next operation is about v by an angle theta. So, this comes out by an angle theta, goes by theta, and the third one is about w by an angle psi. So, about w by an angle psi, this comes out by psi further and further. So, the final orientation is given by u double das, v double das, and w double das. These are the final orientations of the system. So, again, it is global and local, local that is system 2. The only difference is instead of x, local x, it is local y, that is v over here.

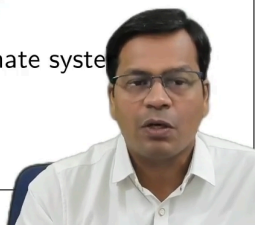
Euler angle: System II

$$\mathbf{R}_{\phi,\theta,\psi} = \mathbf{R}_{z,\,\phi}\,\mathbf{R}_{v,\,\theta}\,\mathbf{R}_{w,\,\psi}$$

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

$$= \begin{bmatrix} C\phi C\theta C\psi - S\phi S\psi & -C\psi C\theta S\psi - S\phi C\psi & C\phi S\theta \\ S\phi C\theta C\psi + C\phi S\psi & -S\psi C\theta S\psi + C\phi C\psi & S\phi S\theta \\ -S\theta C\psi & S\theta S\psi & C\theta \end{bmatrix}$$

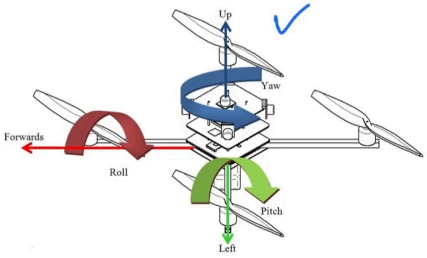
Note: Equivalent rotations about principal axis of the reference coordinate syste
rotation of ψ about OZ, θ about OY and finally ϕ about OZ.



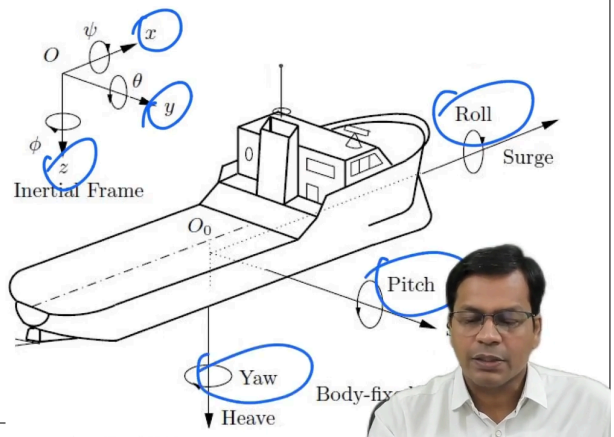
This again can be written like this: global, local, local in a Post-multiplication way. So, rotation about the z-axis, rotation about y, rotation about z. But because it is local, it comes in post-multiplied form. So, you reach a particular orientation. Again, this is equivalent to principal axis rotation. You can write it as global, global, or global. Instead of global, local, local, you can have global, global, and global. So, the order would be reversed. How much is it? It is rotation about z by an angle phi, rotation about y by an angle theta, and rotation about z by an angle phi. It will take you to the same orientation. This is Euler angle system 2. There are 24 such combinations combining global, local, and local. Maybe there are many such ways you can arrange it. There are a total of 24 such combinations which are possible.

Euler angle: System III (RPY)

Conveniently used for any floating base system.
e.g.: Aircraft, Marine vessels, Mobile phones, HMD, Quadcopters, Formula I Cars, Industrial robots, etc.



$$\mathbf{R}_{\phi,\theta,\psi} = \mathbf{R}_{z,\,\phi}\,\mathbf{R}_{y,\,\theta}\,\mathbf{R}_{x,\,\psi}$$



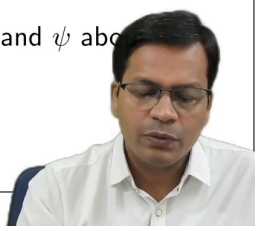
The best one that I found, which is most widely applied, is the Roll-Pitch-Yaw system, which is the Euler angle system 3, and it is very conveniently used in the case of any floating objects like aircraft, marine vessels, and mobile phones. These days, you have an accelerometer in your mobile where you can play games or use a head-mounted device. Quadcopters, Formula I cars, Industrial robots, and end-effector poses can all be expressed using the roll-pitch-yaw type of combination. In this case, all three rotations are with respect to the global frame. Rotation about the x-axis, rotation about the y-axis, and rotation about the z-axis are known as the roll, pitch, and yaw axes. So, this is x, this is y, and this is z. The roll, pitch, and yaw axes are shown with respect to the ship, but they are also applied to many other floating base systems.

Euler angle system III: Roll-Pitch-Yaw

$$\mathbf{R}_{\phi,\theta,\psi} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix}$$

$$= \begin{bmatrix} C\phi C\theta & C\phi S\theta S\psi - S\phi C\psi & C\phi S\theta C\psi + S\phi S\psi \\ S\phi C\theta & S\phi S\theta S\psi + C\phi C\psi & S\phi S\theta C\psi - C\phi S\psi \\ -S\theta & C\theta S\psi & C\theta C\psi \end{bmatrix}$$

Note: Equivalent rotation may be given as, ϕ about OZ, θ about OV and ψ abo OU axis.



If you write them in matrix form, this is your rotation about the x-axis by an angle phi, the y-axis by theta, and the z-axis by psi: phi, theta, psi. Multiply them, and you get this. Again, this can be expressed as global, local, or local.

That is all for this lecture. In the next lecture, we will discuss Robot Frames, Dh Parameters, and Link Transformation Matrices, and we will do Forward Kinematics.

That is all for this lecture.

Thanks a lot.