3D Rotation

Note: This section is currently under revision.

3D Rotation can be accomplished in a number of ways. Computer systems often favor Quaternions for certain mathematical properties. However Quaternions are not terribly easy for humans to interpret or understand specific values.

It must be stated that rotational systems are all mathematically consistent and equally valid. The method discussed here is simply easier for use of humans.

3x3 Rotation

This system describes an arbitrary rotation in 3D space with roll, pitch, and yaw, labeled $\psi, \phi,$ and θ.

Yaw θ describes rotation about z-axis. Pitch ϕ describes rotation about the y-axis. Roll ψ describes rotation about the x-axis. All rotations described here are right-handed.

These three values can be used to generate a 3×3 orthonormal matrix, with a determinant of 1, that rotates any $\begin{bmatrix} x,y,z \end{bmatrix}$ vector.

Roll

To Roll a vector about the x-axis, left-multiply it by the rotation vector R_ψ.

$$ R_\psi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\psi) & -sin(\psi) \\ 0 & sin(\psi) & cos(\psi) \end{bmatrix} $$

Pitch

To Pitch a vector about the y-axis, left-multiply it by the rotation vector R_ϕ.

$$ R_\phi = \begin{bmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{bmatrix} $$

Yaw

To Pitch a vector about the y-axis, left-multiply it by the rotation vector R_ϕ.

$$ R_\theta = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

Complete Rotation

A full 3D rotation includes a roll, pitch, and yaw. With these three rotations, we can describe any arbitrary orientation.

Order of operation is important. The complete R matrix describes the vehicle first yawing around its own z-axis, then pitching along its own y-axis, and then finally rolling about its own x-axis. As an example, the rotation $R([180 10 30])$ would have the submarine pointed to the left 30^o, then pitched slightly upwards by 10^o, and then rolling onto its back at 180^o.

However, our rotation matrices do not provide rotations about our vehicle's intrinsic axes. They rotate vectors about the global, static x,y,z axes. Thus, to achieve a complete rotation, the vector must be first rolled, then pitched, then yawed, relative to these constant axes. These matrices are meant to operate on 3×1 column vectors on right right-hand side. As such, order of operation goes from right to left as more rotations are tacked onto the system.

$$ R_{\psi,\phi,\theta} = R_\theta R_\phi R_\psi \\ R = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\psi) & -sin(\psi) \\ 0 & sin(\psi) & cos(\psi) \end{bmatrix} \\ R([\psi,\phi,\theta]) = \begin{bmatrix} cos(\theta)sin(\phi) & cos(\theta)sin(\phi)sin(\psi) - sin(\theta)cos(\psi) & cos(\theta)sin(\phi)cos(\psi) + sin(\theta)sin(\psi) \\ sin(\theta)cos(\phi) & sin(\theta)sin(\phi)sin(\psi) + cos(\theta)cos(\psi) & sin(\theta)sin(\phi)cos(\psi) - cos(\theta)sin(\psi) \\ -sin(\phi) & cos(\phi)sin(\psi) & cos(\phi)cos(\psi) \end{bmatrix} $$

Additionally, it is important to be able to reverse the process, and identify which combination of roll, pitch, and yaw describes the current orientation. This will be a non-unique combination of values, as there are an arbitrary number of ways to reach a given orientation. However, this process will yield consistent results.

$$ \theta = \arctan(R_{21},R_{11}),\quad [-180^o,180^o] \\ \phi = \arctan(-R_{31},\sqrt{R_{32}^2 + R_{33}^2}),\quad [-90^o,90^o] \\ \psi = \arctan(R_{32},R_{33}),\quad [-180^o,180^o] $$

In the event that $\phi = \pm90^o$ the other values must be determined using the more complicated four elements in the upper-right corner. When pitched by $\pm90^o$ yaw and roll become meaningless as independent values - only their sum or difference remain. Thus we can describe the orientation vector as $\begin{bmatrix}0 & \phi & \theta^* \end{bmatrix}$ or as $\begin{bmatrix}\psi^* & \phi & 0\end{bmatrix}$. We will go with the first version, reporting the roll parameter as equal to zero. For an arbitrary rotation including a pitch of $\phi = +90^o$ the yaw is reported as $\theta^* = \theta-\psi$. For an arbitrary rotation including a pitch of $\phi = +90^o$ the yaw is reported as $\theta^* = \theta+\psi$.

Looking at the formulation for the elements of R above and assuming $\phi = \pm90^o$ we can find how to reverse-calculate our θ^*

$$ \theta^* = \arctan(-R_{12},R_{22}) $$

Inverse Rotation

To reverse the rotation of an $ \begin{bmatrix} x,y,z \end{bmatrix} $ matrix, you multiply it by the inverse of the rotation matrix R^{-1}. The inverse of an orthagonal matrix is equal to its transpose.

$$ RU = U^R \\ \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x^{R} \\ y^{R} \\ z^{R} \end{bmatrix} \\ \\ R^{\mathrm {T}}RU = R^{\mathrm {T}}U^R = U \\ \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} ^{\mathrm {T}} \begin{bmatrix} x^R \\ y^R \\ z^R \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} $$

It is worth stating explicitly that $R^{-1} \neq R(-\psi,-\phi,-\theta)$. If you yaw, then pitch, then roll into an orientation, you *cannot* anti-yaw, then anti-pitch, then anti-roll from that orientation to get back to the origin. You'd have to anti-roll, then anti-pitch, then anti-yaw. It must be rotated completely in reverse. It must be multiplied by its *transpose* $R^{\mathrm {T}}$.

$$ U \neq (R_{-\theta} R_{-\phi} R_{-\psi})(R_\theta R_\phi R_\psi) U \\ U = (R_{-\psi} R_{-\phi} R_{-\theta}) (R_\theta R_\phi R_\psi) U \\ U = (R_{-\psi} (R_{-\phi} (R_{-\theta} R_\theta) R_\phi) R_\psi) U \\ U = (R_{-\psi} (R_{-\phi} (I) R_\phi) R_\psi) U \\ U = (R_{-\psi} (R_{-\phi} R_\phi) R_\psi) U \\ U = (R_{-\psi} (I) R_\psi) U \\ U = (R_{-\psi} R_\psi) U \\ U = (I) U \\ $$

The Equivalent yaw/pitch/roll combination for a Rotation Matrix's transpose will not necessarily have any values corresponding to the original roll/pitch/yaw rotation.

Equivalent Rotations

It is often necessary to find an equivalent rotation from a series or rotations, or as the difference between two rotations.

We often want to calculate where our vector is at after rotating first by R_1, then by R_2, and finally by R_3. If you recall from above, these complete Rotation matrices will behave exactly as the specific roll, pitch, and yaw matrices. That is to say, the rotations they perform are all relative to the fixed global x,y,z axes. Which means that the rotation performed last, R_3, must be allowed to act on the vector first.

Were we to tell our submarine's control system to make a relative goal of $R(\psi_1,\phi_1,\theta_1)$, and then once accomplishing it make another relative rotation $R(\psi_2,\phi_2,\theta_2)$, and then finally tell it to make a third relative rotation $R(\psi_3,\phi_3,\theta_3)$, we would calculate the result as

$$ R_{123}U = (R_1(R_2(R_3U))) $$ and **NOT** $$ R_{123}U \neq (R_3(R_2(R_1U)) $$

Be sure to remember this, or you'll get headaches down the line.

Now we have a different question. Say we are at orientation R_1 and we want to rotate to another arbitrary rotation R_2. For instance, we are at orientation $R_1(20,-40,7)$ and we tell our control system we want to go to the absolute orientation $R_1(70,10,-40)$. What rotation will move us between these two orientations?

Using what we know from above, about the order of operation, we can make an equation. Recall that R_{err} will rotate us from where we are currently, at R_1. Thus we must apply it *before* R_1:

$$ R_1R_{err}U = R_2U \\ R_1^{\mathrm {T}}R_1R_{err}U = R_1^{\mathrm {T}}R_2U\\ \\ R_{err} = R_1^{\mathrm {T}}R_2 \\ $$

To find the rotation *between* two rotations, it is helpful to ask the question “What rotation would I need to achieve R_2 if R_1 was at the origin?” The answer is, of course, just R_2. We find our relative motion by first un-rotating R_2 by R_1

Special Properties

Rotation matrices have some very exploitable properties. It stands to reason that there is an underlying structure, as we are using 9 elements to represent only 3 unique values.

Othanormal

A Rotation matrix's Transpose is equal to its inverse.

$$ 1=\det(I)=\det(R^{\mathrm {T} }R)=\det(R^{\mathrm {T} })\det(R)=(\det(R))^{2} \\ \det(R) = \pm1 $$

Code

To produce a 3×3 rotation matrix from roll ψ, pitch ϕ, and yaw θ use the following matlab code or it's C++ equivalent:

```
function [R] = r3D(omega) % omega = [roll; pitch; yaw]

R = [cosd(omega(3)) -sind(omega(3)) 0;...
    sind(omega(3)) cosd(omega(3)) 0;...
    0 0 1] * ...
    [cosd(omega(2)) 0 sind(omega(2));...
    0 1 0;...
    -sind(omega(2)) 0 cosd(omega(2))] * ...
    [1 0 0;...
    0 cosd(omega(1)) -sind(omega(1));...
    0 sind(omega(1)) cosd(omega(1))];

end
```

To find an equivalent roll ψ, pitch ϕ, and yaw θ *given* a 3×3 rotation matrix R use the following code:

```
function [omega] = ir3D(R)
```

```
    theta = atan2di(R(2,1),R(1,1));
    psi = atan2di(R(3,2),R(3,3));
    phi = atan2di(-R(3,1),sqrt(sum(R(3,2:3).^2)));

    if(abs(phi)==90)
        theta = atan2di(-R(1,2),R(2,2));
        psi = 0;
    end


    omega = [psi;phi;theta]; %omega = [roll;pitch;yaw]
end
```