

Coordinate Transforms

Brad Thoen, Staff Engineer
MTS Systems Corporation
Rev. 28-Dec-05

In degree-of-freedom (DOF) systems, motion is both commanded and controlled in a coordinate system other than that of the displacement, force, and acceleration sensor feedbacks. Therefore these feedbacks must first be transformed to what is called the world coordinate system, consisting of three translations in a cartesian frame $[x, y, z]$, followed by three rotations $[\theta_x, \theta_y, \theta_z]$ about the translated cartesian frame. In addition, because the controller computes correction signals in the world coordinate system, the signals must be transformed back to the actuator coordinate system. Derivation of these coordinate transforms is the focus of this paper.

Important note about nomenclature: The term “world coordinate system” as defined by the vector $[x, y, z, \theta_x, \theta_y, \theta_z]$ is somewhat improper, because it is in fact a mixture of both world and table coordinates. The translations $[x, y, z]$ are truly world coordinates, but the rotations $[\theta_x, \theta_y, \theta_z]$ are actually table coordinates because the rotations occur about the translated table origin rather than the fixed world origin. Nevertheless, in the following text the term “world coordinate system” is used for brevity. In addition, the term “degrees-of-freedom” is used synonymously with “world coordinates”.

In this paper, three basic transformations are derived: displacement, force, and acceleration. It is assumed that the actuators are inline style, i.e., the actuators are connected between the foundation and the table with no intervening linkages except for pivots. If the actuators are sliding head style (i.e., where a bearing surface replaces the head pivot), complications occur that are dealt with in Appendix B. Similarly, sliding base style actuators (where a bearing surface replaces the base pivot) are dealt with in Appendix C. Lastly, pedestal-link style actuators are dealt with in Appendix D.

Displacement Coordinate Transforms

Both directions of displacement coordinate transformation are required. The world-to-actuator transform is the most straightforward, and it forms the basis for the more difficult actuator-to-world transform. In addition, an approximation to the world-to-actuator transform, the Jacobian matrix, is used to transform degree-of-freedom controller output signals into actuator drive signals.

World-to-Actuator Displacement Coordinate Transform

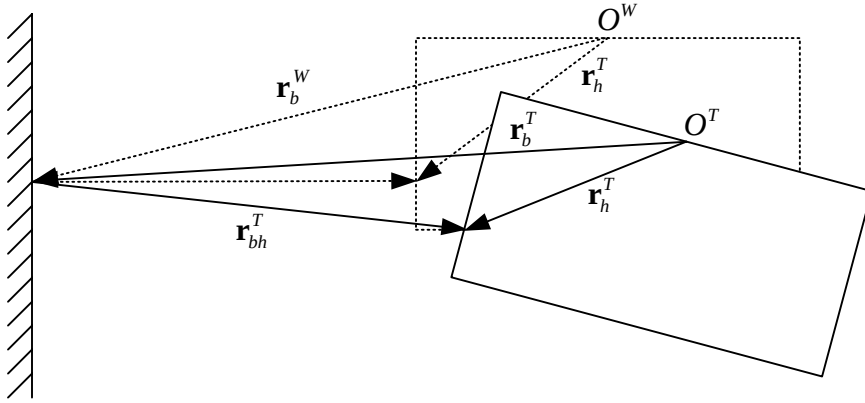
Problem Statement: Given world displacement $\mathbf{d}_w = [x, y, z, \theta_x, \theta_y, \theta_z]$, find actuator displacements $\mathbf{d}_A = [d_1, d_2, \dots, d_N]$, where N is the number of actuators.

Referring to the figure below, radius vectors $\mathbf{r}_b^w$ and $\mathbf{r}_h^T$ are fixed vectors that define the location of the actuator base pivot and the actuator head pivot, respectively, when the table origin O^T and

world origin O^W are collocated (i.e., the table is at its “home” position). These are fixed input parameters to the coordinate transform that represent table/actuator geometry. As the table frame at O^T translates and rotates with respect to the world frame at O^W , radius vector $\mathbf{r}_b^T$ continues to point to the actuator base pivot relative to table origin O^T , by performing the same translation and rotation on base radius vector $\mathbf{r}_b^W$ as that experienced by the table frame. Radius vector translation and rotation is performed by multiplying it by a translation-rotation matrix. (Note: A great deal of use will be made of translation-rotation matrices, so a brief treatment is given in Appendix A.) The vector corresponding to the actuator length and orientation, $\mathbf{r}_{bh}^T$, is formed as the vector difference

$$\mathbf{r}_{bh}^T = \mathbf{r}_h^T - \mathbf{r}_b^T = \mathbf{r}_h^T - \mathbf{T}_{W \rightarrow T} \mathbf{r}_b^W.$$

Note that in the above expression, $\mathbf{r}_h^T$ and $\mathbf{r}_b^W$ are fixed vectors representing table/actuator geometry; while matrix $\mathbf{T}_{W \rightarrow T}$ varies according to world displacement vector $\mathbf{d}_W = [x, y, z, \theta_x, \theta_y, \theta_z]$.



The length of the actuator is obtained from the magnitude of $\mathbf{r}_{bh}^T$:

$$\ell = \|\mathbf{r}_{bh}^T\|.$$

Actuator displacement is obtained by subtracting the length of the actuator when the table is at home position, ℓ_0 :

$$d = \ell - \ell_0.$$

The actuator orientation unit vector $\mathbf{e}_{bh}^T$, which will be used in subsequent force transforms, is obtained by dividing $\mathbf{r}_{bh}^T$ by its magnitude:

$$\mathbf{e}_{bh}^T = \frac{\mathbf{r}_{bh}^T}{\|\mathbf{r}_{bh}^T\|}.$$

The above operations are performed for each of N actuators, yielding actuator displacement vector

$$\mathbf{d}_A = [d_1, d_2, \dots, d_N]$$

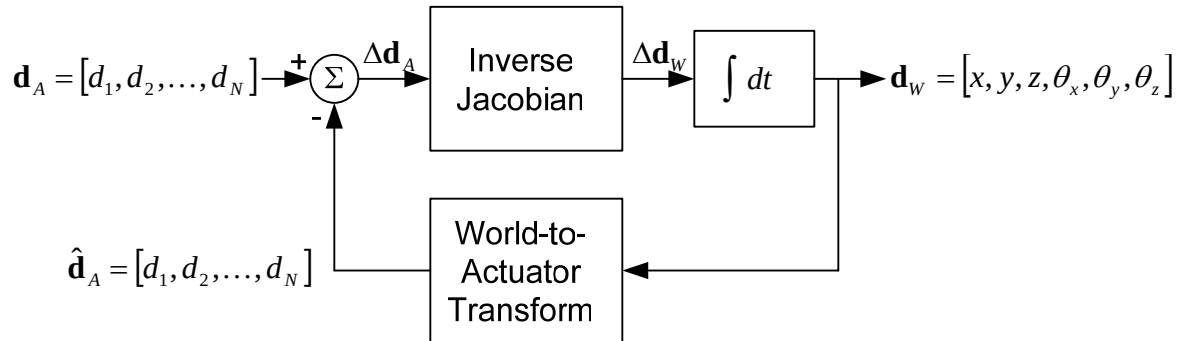
and actuator orientation matrix

$$\mathbf{E}_A^T = [\mathbf{e}_{bh_1}^T, \mathbf{e}_{bh_2}^T, \dots, \mathbf{e}_{bh_N}^T].$$

Actuator-to-World Displacement Coordinate Transform

Problem Statement: Given N actuator displacements $\mathbf{d}_A = [d_1, d_2, \dots, d_N]$, find the world displacement $\mathbf{d}_W = [x, y, z, \theta_x, \theta_y, \theta_z]$.

While the world-to-actuator transform is a simple closed-form expression, the reverse direction is not so straightforward. In fact, it must be solved by iterative means. This iterative method is shown schematically in the following diagram:



Conceptually, this iterative transform can be viewed as a feedback control system. Measured actuator displacements vector $\mathbf{d}_A$ is compared against estimated actuator displacements vector $\hat{\mathbf{d}}_A$ to yield an estimate error vector $\Delta \mathbf{d}_A$. This estimate error is multiplied by the inverse Jacobian matrix, defined as

$$\mathbf{J}^{-1} = \frac{\partial \mathbf{d}_W}{\partial \mathbf{d}_A},$$

which serves to approximately transform estimate error in the actuator coordinate system to estimate error in the world coordinate system. The resulting world estimate error $\Delta \mathbf{d}_W$ is integrated to obtain world displacement vector $\mathbf{d}_W$, which is then transformed back to the actuator coordinate system via the world-to-actuator transform to form a new estimated actuator displacement $\hat{\mathbf{d}}_A$. As the estimate error $\Delta \mathbf{d}_A$ approaches zero, as it must because of the integrative process, the world displacement vector $\mathbf{d}_W$ approaches the true, exact transformed value of actuator displacement vector $\mathbf{d}_A$.

Notes:

- 1) Although the inverse Jacobian is itself an approximate transform, the closed-loop nature of the iterative algorithm coupled with the exact, approximation-free world-to-actuator transform combine to make the overall actuator-to-world transform exact.
- 2) The Jacobian matrix can be determined analytically or numerically. Equations for the Jacobian matrix (beyond the scope of this paper) are derived in the paper titled *Analytic Jacobian*. Numerical computation of the Jacobian matrix can be done by incrementally perturbing and transforming the current world displacement

$$\mathbf{d}_w \pm \Delta \mathbf{d}_w = [x \pm \Delta x, y \pm \Delta y, z \pm \Delta z, \theta_x \pm \Delta \theta_x, \theta_y \pm \Delta \theta_y, \theta_z \pm \Delta \theta_z]$$
 to yield incremental actuator displacement values $\Delta \mathbf{d}_A = [\Delta d_1, \Delta d_2, \dots, \Delta d_N]$, then dividing to form $\mathbf{J} = \frac{\partial \mathbf{d}_A}{\partial \mathbf{d}_w}$.
- 3) The inverse Jacobian matrix can be found by simple matrix inversion if the Jacobian matrix is square, i.e, if the number of actuators equals the number of degrees-of-freedom. If the number of actuators exceeds the number of degrees-of-freedom, as is often the case, the Jacobian matrix will not be square and must be inverted using pseudoinverse matrix methods. One such method of finding pseudoinverse matrix $A^\#$ that satisfies $AA^\# = I$ uses the formula $A^\# = (A'A)^{-1}A'$.

Force and Moment Coordinate Transforms

Forces are computed in the world coordinate system. Conversely, moments are computed in the table coordinate system because for table control purposes, moments relative to the current table origin are more relevant than moments relative to the fixed world origin.

Only one direction of force and moment transformation, actuator-to-world, is required.

Actuator-to-World Force Coordinate Transform

Problem Statement: Given N actuator forces $\mathbf{f}_A = [f_1, f_2, \dots, f_N]$ and actuator orientation matrix $\mathbf{E}_A^T = [(\mathbf{e}_{bh}^T)_1, (\mathbf{e}_{bh}^T)_2, \dots, (\mathbf{e}_{bh}^T)_N]$ obtained from the displacement transform, find the world force $\mathbf{f}_w = [f_x, f_y, f_z]$.

The 3x1 world force vector $\mathbf{f}_w$ is obtained from the dot product of the 3xN world actuator orientation matrix $\mathbf{E}_A^w$ with Nx1 actuator force measurement vector $\mathbf{f}_A$:

$$\mathbf{f}_w = \mathbf{E}_A^w \bullet \mathbf{f}_A$$

Since the actuator orientation matrix that results from the displacement transform is in the table coordinate system, it must be converted to the world coordinate system by multiplying it by 3x3 rotation matrix $\mathbf{R}_{T \rightarrow W}$

$$\mathbf{E}_A^W = \mathbf{R}_{T \rightarrow W} \mathbf{E}_A^T$$

yielding the final result

$$\mathbf{f}_W = \mathbf{R}_{T \rightarrow W} \mathbf{E}_A^T \bullet \mathbf{f}_A.$$

Actuator-to-Table Moment Coordinate Transform

Problem Statement: Given N actuator forces $\mathbf{f}_A = [f_1, f_2, \dots, f_N]$ and actuator orientation matrix $\mathbf{E}_A^T = [(\mathbf{e}_{bh}^T)_1, (\mathbf{e}_{bh}^T)_2, \dots, (\mathbf{e}_{bh}^T)_N]$ obtained from the displacement transform, find the table moment $\mathbf{m}_T = [m_x, m_y, m_z]$.

The 3x1 table moment vector $\mathbf{m}_T$ is the cross product of the table force vector with the moment arm radius vectors. The table force is computed in a manner similar to the world table force,

$$\mathbf{f}_T = \mathbf{E}_A^T \bullet \mathbf{f}_A,$$

and the moment arm radius vectors are simply the fixed radius vectors $\mathbf{r}_h^T$ that define the location of the actuator head pivots, yielding the final result

$$\mathbf{m}_T = \left(\left[(\mathbf{r}_h^T)_1, (\mathbf{r}_h^T)_2, \dots, (\mathbf{r}_h^T)_N \right] \times \mathbf{E}_A^T \right) \bullet \mathbf{f}_A.$$

Acceleration Coordinate Transforms

Both directions of acceleration coordinate transforms are required. The world-to-accelerometer transform is the most straightforward, and is required only because it forms the basis of the more difficult accelerometer-to-world transform.

World-to-Accelerometer Acceleration Coordinate Transform

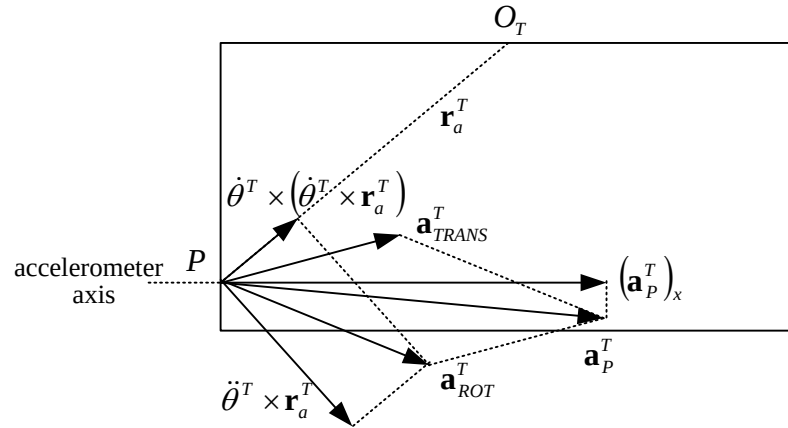
Problem Statement: Given world acceleration $\mathbf{a}_W = [\ddot{x}, \ddot{y}, \ddot{z}, \ddot{\theta}_x, \ddot{\theta}_y, \ddot{\theta}_z]$ and table rotational velocities $\dot{\theta}_T = [\dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z]$, find accelerations $\mathbf{a}_X = [a_1, a_2, \dots, a_M]$ measured at each of M accelerometers.

The total acceleration at any point P has three contributing terms:

$$\mathbf{a}_P^T = \mathbf{a}_{TRANS}^T + \ddot{\theta}^T \times \mathbf{r}_a^T + \dot{\theta}^T \times (\dot{\theta}^T \times \mathbf{r}_a^T)$$

where $\dot{\theta}^T$ and $\ddot{\theta}^T$ are the rotational velocity and acceleration about table origin O_T , and $\mathbf{r}_a^T$ is the radius vector from table origin O_T to point P . The first term, $\mathbf{a}_{TRANS}^T$, is the translational component. The second term, $\ddot{\theta}^T \times \mathbf{r}_a^T$, is the tangential component due to rotational

acceleration. The third term, $\dot{\theta}^T \times (\dot{\theta}^T \times \mathbf{r}_a^T)$, is the normal component due to rotational velocity. This equation is shown schematically in the figure below:



The above equation will now be rearranged into matrix form. For notational simplicity, many of the subscripts and superscripts will be omitted. Expanding vector quantities into their individual components:

$$\mathbf{a}_P^T = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} + \begin{bmatrix} \ddot{\theta}_x \\ \ddot{\theta}_y \\ \ddot{\theta}_z \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} + \begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{bmatrix} \times \left(\begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \right).$$

After evaluating the cross products and rearranging (details omitted),

$$\mathbf{a}_P^T = \begin{bmatrix} 1 & 0 & 0 & 0 & r_z & -r_y \\ 0 & 1 & 0 & -r_z & 0 & r_x \\ 0 & 0 & 1 & r_y & -r_x & 0 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\theta}_x \\ \ddot{\theta}_y \\ \ddot{\theta}_z \end{bmatrix} + \begin{bmatrix} -(\dot{\theta}_y^2 + \dot{\theta}_z^2) & \dot{\theta}_x \dot{\theta}_y & \dot{\theta}_x \dot{\theta}_z \\ \dot{\theta}_x \dot{\theta}_y & -(\dot{\theta}_x^2 + \dot{\theta}_z^2) & \dot{\theta}_y \dot{\theta}_z \\ \dot{\theta}_x \dot{\theta}_z & \dot{\theta}_y \dot{\theta}_z & -(\dot{\theta}_x^2 + \dot{\theta}_y^2) \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

The above equation gives the acceleration at any point P on the table. If an accelerometer is placed at that point, the measured acceleration is the projection of $\mathbf{a}_P^T$ onto the axis of the accelerometer. Since all accelerometers are aligned on the table's x, y, or z axis, the measured acceleration is simply the x, y, or z component of vector $\mathbf{a}_P^T$. Thus, the acceleration measured by an accelerometer located at point P is

$$(\mathbf{a}_P^T)_x = \begin{bmatrix} 1 & 0 & 0 & 0 & r_z & -r_y \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\theta}_x \\ \ddot{\theta}_y \\ \ddot{\theta}_z \end{bmatrix} + \begin{bmatrix} -(\dot{\theta}_y^2 + \dot{\theta}_z^2) & \dot{\theta}_x \dot{\theta}_y & \dot{\theta}_x \dot{\theta}_z \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

for x-oriented accelerometers,

$$(\mathbf{a}_p^T)_y = \begin{bmatrix} 0 & 1 & 0 & -r_z & 0 & r_x \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\theta}_x \\ \ddot{\theta}_y \\ \ddot{\theta}_z \end{bmatrix} + \begin{bmatrix} \dot{\theta}_x \dot{\theta}_y & -(\dot{\theta}_x^2 + \dot{\theta}_z^2) & \dot{\theta}_y \dot{\theta}_z \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

for y-oriented accelerometers, or

$$(\mathbf{a}_p^T)_z = \begin{bmatrix} 0 & 0 & 1 & r_y & -r_x & 0 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\theta}_x \\ \ddot{\theta}_y \\ \ddot{\theta}_z \end{bmatrix} + \begin{bmatrix} \dot{\theta}_x \dot{\theta}_z & \dot{\theta}_y \dot{\theta}_z & -(\dot{\theta}_x^2 + \dot{\theta}_y^2) \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

for z-oriented accelerometers.

Combining the outputs of M accelerometers into vector form yields a matrix equation that is the final result:

$$\mathbf{a}_x = \mathbf{A} \bullet \mathbf{a}_w + \mathbf{a}_{\dot{\theta}}$$

where $\mathbf{a}_x$ is an Mx1 vector of accelerometer measurements, $\mathbf{A}$ is an Mx6 constant matrix determined by accelerometer locations, $\mathbf{a}_w$ is the 6x1 world acceleration vector, and $\mathbf{a}_{\dot{\theta}}$ is an Mx1 vector determined by table rotational velocities and accelerometer locations.

Accelerometer-to-World Acceleration Coordinate Transform

Problem Statement: Given accelerations $\mathbf{a}_x = [a_1, a_2, \dots, a_M]$ measured at each of M accelerometers and table rotational velocities $\dot{\theta}_T = [\dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z]$, find world acceleration $\mathbf{a}_w = [\ddot{x}, \ddot{y}, \ddot{z}, \ddot{\theta}_x, \ddot{\theta}_y, \ddot{\theta}_z]$.

The world acceleration can be found by solving the system of M equations determined in the previous section for the six unknowns in $\mathbf{a}_w$:

$$\mathbf{a}_x = \mathbf{A} \bullet \mathbf{a}_w + \mathbf{a}_{\dot{\theta}}.$$

Since the number of accelerometers (M) is usually greater than the number of degrees-of-freedom (six), the system of equations above is overdetermined and cannot be inverted. However, it can be “pseudo-inverted” in a least-squares sense. The weighted least-squares pseudo-inverse solution of the above equation is

$$\mathbf{a}_w = \left[\left[(\mathbf{WA})' \mathbf{WA} \right]^{-1} (\mathbf{WA})' \right] \bullet (\mathbf{a}_x - \mathbf{a}_{\dot{\theta}})$$

where the prime symbol denotes matrix transposition. MxM matrix $\mathbf{W}$ represents the desired weighting for the accelerometers; an identity matrix results in equal weight for all accelerometers in the solution for $\mathbf{a}_w$. Because both $\mathbf{A}$ and $\mathbf{W}$ are constant matrices, $\left[\left[(\mathbf{WA})' \mathbf{WA} \right]^{-1} (\mathbf{WA})' \right]$ can be precomputed.

Notes:

- 1) To include the effect of gravity on the accelerometers when transforming world acceleration to accelerometer acceleration, first subtract the gravity constant from world vertical acceleration $\mathbf{a}_w$ and then compute accelerometer values $\mathbf{a}_x$ as usual. Optionally, the effect of adding electrical offset to conditioned vertical accelerometer signals so that they output zero volts at rest can be simulated by adding the gravity constant to vertical accelerometer components of $\mathbf{a}_x$. In the reverse direction (accelerometer acceleration to world acceleration), these steps are reversed.
- 2) Table rotational velocities can be computed by feedback observers using table rotational displacements and accelerations.

Appendix A: Translation-Rotation Matrices

A translation-rotation matrix is a 4x4 matrix that will take an augmented three-dimensional cartesian vector $[x, y, z, 1]$, translate it, then rotate it by three angles: roll (rotation around the x axis), pitch (rotation around the y axis), and yaw (rotation around the z axis). The order of rotation affects the orientation of the resulting vector, i.e., different sequences of rotation will yield different results. However the choice is usually arbitrary; a commonly used sequence is: first yaw (θ_z), then pitch (θ_y), then roll (θ_x). The rotations can be performed with three separate matrices, each matrix representing one rotation, or all three rotations can be performed simultaneously by using the product of the three matrices:

$$\mathbf{T}_{T \rightarrow W} = \begin{bmatrix} \cos \theta_z & -\sin \theta_z & 0 & 0 \\ \sin \theta_z & \cos \theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} \cos \theta_y & 0 & \sin \theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta_y & 0 & \cos \theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x & 0 \\ 0 & \sin \theta_x & \cos \theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(rotation θ_z) (rotation θ_y) (rotation θ_x)

Combining the three matrices into one and adding in translation, the matrix has the form

$$\mathbf{T}_{T \rightarrow W} = \begin{bmatrix} \mathbf{R}_{T \rightarrow W} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\mathbf{p}$ is a 3x1 vector corresponding to the translation part, and $\mathbf{R}_{T \rightarrow W}$ is a 3x3 matrix corresponding to the rotational part:

$$\mathbf{R}_{T \rightarrow W} = \begin{bmatrix} cy \cdot cz & sx \cdot sy \cdot cz - cx \cdot sz & cx \cdot sy \cdot cz + sx \cdot sz \\ cy \cdot sz & sx \cdot sy \cdot sz + cx \cdot cz & cx \cdot sy \cdot sz - sx \cdot cz \\ -sy & sx \cdot cy & cx \cdot cy \end{bmatrix}$$

where

$$\begin{aligned} cx &= \cos \theta_x & sx &= \sin \theta_x \\ cy &= \cos \theta_y & sy &= \sin \theta_y \\ cz &= \cos \theta_z & sz &= \sin \theta_z . \end{aligned}$$

Often it is required to translate and rotate in the reverse direction, in which case the matrix becomes

$$\mathbf{T}_{W \rightarrow T} = \mathbf{T}_{T \rightarrow W}^{-1} = \begin{bmatrix} \mathbf{R}_{T \rightarrow W}^{-1} & -\mathbf{R}_{T \rightarrow W}^{-1} \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It should be noted that because the rotation matrix $\mathbf{R}_{T \rightarrow W}$ is an orthonormal matrix, its inverse is simply the transpose, i.e., $\mathbf{R}_{T \rightarrow W}^{-1} = \mathbf{R}_{T \rightarrow W}'$.

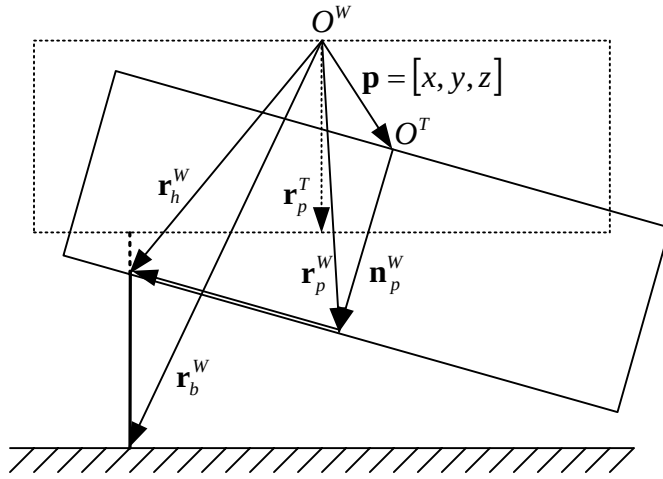
For a more complete treatment of translation-rotation matrices, refer to: McKerrow, Introduction to Robots, Addison-Wesley, 1991, pp. 142-143.

Appendix B: Complications posed by sliding head actuators

For an actuator that interfaces to the table via a bearing surface instead of a traditional pivot, its radius vector $\mathbf{r}_h^T$ is not fixed because the interface point varies with table translation and rotation. This appendix describes how $\mathbf{r}_h^T$ can be determined, after which the usual coordinate transformation process can proceed.

Assuming the actuator is aligned in the direction of one of the cartesian axes, two of the three components of $\mathbf{r}_h^T$ are fixed and can be determined from the corresponding base pivot location vector $\mathbf{r}_b^T$. The third unknown component can be found by solving where the actuator intersects

with the plane of the table. This is shown below for the case of a vertically-aligned actuator (similar results hold for a horizontally-aligned actuator):



The first step in finding the intersection point is to transform a convenient vector normal to the table bottom, $\mathbf{r}_p^T$, to the world coordinate system via multiplication with the translation-rotation matrix

$$\mathbf{r}_p^W = \mathbf{T}_{T \rightarrow W} \mathbf{r}_p^T.$$

Next, the radius vector normal to the table bottom, $\mathbf{n}_p^W$, is found by

$$\mathbf{n}_p^W = \mathbf{r}_p^W - \mathbf{p},$$

where $\mathbf{p} = [x, y, z]$ is the translational part of translation-rotation matrix $\mathbf{T}_{T \rightarrow W}$. Then using the normal vector $\mathbf{n}_p^W$ and a vector known to lie in the plane, namely $\mathbf{r}_h^W - \mathbf{r}_p^W$, solve the plane equation

$$\mathbf{n}_p^W \bullet (\mathbf{r}_h^W - \mathbf{r}_p^W) = 0$$

to determine the unknown z component of $\mathbf{r}_h^W$ (the x and y components are taken from base attachment vector $\mathbf{r}_b^T$)

$$\mathbf{r}_h^W = \begin{bmatrix} (\mathbf{r}_b^W)_x \\ (\mathbf{r}_b^W)_y \\ (\mathbf{r}_p^W)_z - \left(\frac{(\mathbf{n}_p^W)_x (\mathbf{r}_h^W - \mathbf{r}_p^W)_x + (\mathbf{n}_p^W)_y (\mathbf{r}_h^W - \mathbf{r}_p^W)_y}{(\mathbf{n}_p^W)_z} \right) \end{bmatrix}.$$

Lastly, transform $\mathbf{r}_h^W$ to the table coordinate system:

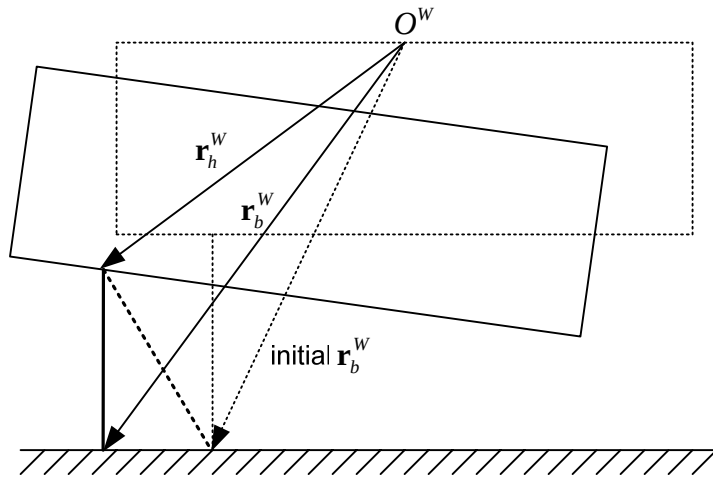
$$\mathbf{r}_h^T = T_{W \rightarrow T} \mathbf{r}_h^W$$

and proceed with the usual coordinate transformation process.

Appendix C: Complications posed by sliding base actuators

For an actuator that interfaces to the foundation via a bearing surface instead of a traditional pivot, its radius vector $\mathbf{r}_b^W$ is not fixed because the interface point varies with table translation and rotation. This appendix describes how $\mathbf{r}_b^W$ can be determined, after which the usual coordinate transformation process can proceed.

Assuming the actuator is aligned in the direction of one of the cartesian axes, the component of $\mathbf{r}_b^W$ in the direction of alignment is fixed and known. The other two components can be determined from the corresponding table attachment radius vector $\mathbf{r}_h^W$. This is shown below for the case of a vertically-aligned actuator (similar results hold for a horizontally-aligned actuator):



In this case, transform the table attachment radius vectors $\mathbf{r}_h^T$ to the world coordinate system via multiplication with the translation-rotation matrix

$$\mathbf{r}_h^W = \mathbf{T}_{T \rightarrow W} \mathbf{r}_h^T$$

and insert the x and y components of $\mathbf{r}_h^W$ into base attachment vector $\mathbf{r}_b^W$:

$$\mathbf{r}_b^W = \begin{bmatrix} (\mathbf{r}_h^W)_x \\ (\mathbf{r}_h^W)_y \\ (\mathbf{r}_b^W)_z \end{bmatrix}$$

and proceed with the usual coordinate transformation process.

Appendix D: Complications posed by pedestal-link actuators

A pedestal-link actuator is one that is fixed to the foundation and is oriented along a cartesian axis, and interfaces to the table via a fixed-length link having a pivot at both ends.

Displacement Coordinate Transform

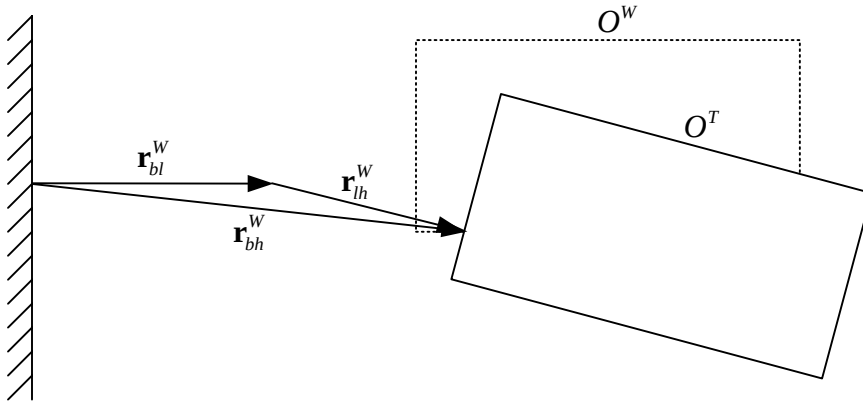
The approach used here is to compute the standard transform as though the actuators are inline-style, then perform additional operations to convert the inline results to pedestal-link results.

Starting with the vector corresponding to the actuator length and orientation, $\mathbf{r}_{ht}^T$, which was computed by the standard displacement coordinate transform, transform it to the world coordinate system via multiplication with the translation-rotation matrix

$$\mathbf{r}_{ht}^W = \mathbf{T}_{T \rightarrow W} \mathbf{r}_{ht}^T.$$

As shown in the figure below, this vector must be resolved into two components: $\mathbf{r}_{bl}^W$, the vector corresponding to the length and orientation of the pedestal actuator, and $\mathbf{r}_{lh}^W$, the vector corresponding to the length and orientation of the link:

$$\mathbf{r}_{bh}^W = \mathbf{r}_{bl}^W + \mathbf{r}_{lh}^W.$$



Because pedestal actuator vector $\mathbf{r}_{bl}^W$ is oriented in the x, y, or z direction, it has the form $[\ell_{bl}, 0, 0]$, $[0, \ell_{bl}, 0]$, or $[0, 0, \ell_{bl}]$. Link vector $\mathbf{r}_{lh}^W$ has unknown orientation, but known length L .

Rearranging the above expression to isolate $\mathbf{r}_{lh}^W$ on the lefthand side, and taking the vector magnitude of both sides,

$$\|\mathbf{r}_{lh}^W\| = \|\mathbf{r}_{bh}^W - \mathbf{r}_{bl}^W\| = \|[r_x, r_y, r_z] - [\ell_x, 0, 0]\|,$$

which evaluates to

$$L = \sqrt{(r_x - \ell_x)^2 + r_y^2 + r_z^2}.$$

Note that in the above expressions it has been assumed that the pedestal actuator is oriented in the x direction; similar expressions exist for y- and z-oriented actuators. Solving for unknown pedestal actuator length ℓ_x ,

$$\ell_x = r_x - \sqrt{L^2 - r_y^2 - r_z^2}.$$

Similarly, for y- and z-oriented pedestal actuators,

$$\begin{aligned}\ell_y &= r_y - \sqrt{L^2 - r_x^2 - r_z^2} \\ \ell_z &= r_z - \sqrt{L^2 - r_x^2 - r_y^2}.\end{aligned}$$

As before, pedestal actuator displacement is obtained by subtracting the length of the pedestal actuator when the table is at home position, ℓ_0 :

$$d = \ell - \ell_0.$$

This is the final result of the displacement transform.

Force and Moment Coordinate Transforms

The approach used here is to treat the link as though it is an inline actuator, first by making appropriate alterations to transform inputs to account for the pedestal-link geometry, then using the standard force and moment transforms.

For inline actuators, the inputs to the standard force and moment transforms are the N actuator forces $\mathbf{f}_A = [f_1, f_2, \dots, f_N]$ and the actuator orientation matrix $\mathbf{E}_A^T = [(\mathbf{e}_{bh}^T)_1, (\mathbf{e}_{bh}^T)_2, \dots, (\mathbf{e}_{bh}^T)_N]$ obtained from the displacement transform. Force measurement vector $\mathbf{f}_A$ will be replaced by $\mathbf{f}_L$, the forces in the links, and orientation matrix $\mathbf{E}_A^T$ will be replaced by $\mathbf{E}_L^T$, the orientation matrix of the links.

The unit orientation vector for a link $\mathbf{e}_{lh}^W$ in the world coordinate system is given by

$$\mathbf{e}_{lh}^W = (\mathbf{r}_{bh}^W - \mathbf{r}_{bl}^W) / L,$$

where all of the quantities on the righthand side were previously determined by the displacement transform. Assembling unit orientation vectors for all N links into one matrix,

$$\mathbf{E}_L^W = [(\mathbf{e}_{lh}^W)_1, (\mathbf{e}_{lh}^W)_2, \dots, (\mathbf{e}_{lh}^W)_N].$$

This matrix must be transformed to the table coordinate system prior to use in the force and moment transforms:

$$\mathbf{E}_L^T = T_{W \rightarrow T} \mathbf{E}_L^W .$$

The pedestal actuator force measurements $\mathbf{f}_p$ are the projection of the link unit vectors onto the pedestal actuator orientation vectors:

$$\mathbf{f}_p = \mathbf{E}_L^W \bullet \mathbf{f}_L .$$

Since the pedestal actuators are oriented in the x, y, or z directions, two components of $\mathbf{f}_p$ are zero, reducing the vector equation to scalar equation that can be easily solved for the unknown link forces:

$$(\mathbf{f}_L)_i = \frac{(\mathbf{f}_p)_i}{((\mathbf{e}_{lt}^W)_i)_{x,y, \text{ or } z}} , \quad 1 \leq i \leq N .$$

Now that the link forces and the link orientation matrix are known, the standard force and moment transforms can be computed.