2. Motion Representation

To solve any problem involving motion in 3-D we have to have a way to represent 3-D motion. Motion representation is a well understood problem but a non trivial one. There are many possible representations and each one has advantages and disadvantages. The troublemaker is rotation. Poor translation has only a couple. But rotation! Rotation has the Rodrigues parameters, the Euler angles (24 different flavors of them, 12 of which are really called fixed), quaternions, angle and axis, raw rotation matrices. Before we get scared, it is not that mathematicians are so much smarter. They just take care of their image very well. The relation between quaternions and Rodrigues parameters is trivial, the Rodrigues could be named axis and tangent of half angle, the Euler angles is what a roboticist would come up with. There are twenty four of them to accommodate all possible definitions of coordinate systems attached to robotic body parts. And no representation of rotation is any good if not associated with a rotation matrix. As for disadvantages each one has its own. The Rodrigues parameters cannot represent rotations of 180 degrees. All Euler angles have two representations (two sets of angles represent the same rotation) plus they have singularities (infinite number of solutions at a few certain points). Quaternions can represent anything without singularities but involve 4 numbers instead of three. And a raw rotation matrix is the best in any other respect but it involves 9 numbers. And only the Rodrigues parameters and the raw matrix can be extended to more than three dimensions (which is good to know in case you need to give driving directions to 4-dimensional aliens).

To make things slightly more complex the representations of rotation and translation can be used either directly or embedded in a homogeneous transformation which is just a 4×4 matrix. The homogeneous route is used in graphics libraries due to their simplicity: instead of having to carry a rotation matrix and a translation vector, they carry a 4×4 matrix only. It is often preferred in robotics and computer vision mostly for the same reasons. The issue of convenience, carried several steps further gave rise to projective geometry. Another mathematical trick to impress the common folk.

Finally, we seem to take for granted that any motion can be represented by a rotation and a translation. While the assertion is certainly correct and plainly obvious, one has to prove it. Unfortunately proving that any rigid motion can be represented by a rotation and a translation is quite hard. Fortunately it offers no intuition and no insight and can be safely omitted. The opposite, proving that any rotation and translation constitute a rigid motion is much simpler and offers some useful insight.

2.1. Translation Vector Poor translation vector. It has only one representation really. Let the translation vector be T

$$T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

and a point P

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

is translated to P', just by adding vector T to it

$$P' = P + T.$$

It is not hard to prove that translation is a rigid motion. We only have to prove that the distance between any pair of of points P_1 and P_2 does not change with motion. So let us prove that

$$\|P_1 - P_2\| = \|P'_1 - P'_2\|$$

where the primed symbols are the points after the motion. Starting from the right hand side

$$\|P'_1 - P'_2\| = \|P_1 + T - (P_2 + T)\| = \|P_1 - P_2\|.$$

That's a nice one line proof.

2.2. Rotation Matrix Now the hard part. To make a hard problem slightly easier lets sojourn to the world of a chicken that did not cross the road all the way. In this 2-D world we can rotate only around a point (pivot) by some angle θ and unless specified otherwise, the pivot is the origin of the coordinate system. So a 2-D point P

$$Q = \begin{bmatrix} q_x \\ q_y \end{bmatrix}$$

rotates to point

$$Q' = \begin{bmatrix} q'_x \\ q'_y \end{bmatrix}$$

and the coordinates of these points are related as follows:

$$q'_x = q_x c - q_y s$$
$$q'_y = q_x s + q_y c$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$. And if we want to pretend that we are adults and write it in matrix form

$$\begin{bmatrix} q'_x \\ q'_y \end{bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} q_x \\ q_y \end{bmatrix}$$
$$Q' = RQ$$

where

$$R = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \tag{2.1}$$

is the rotation matrix. We might be tempted to derive some properties from this simple matrix and generalize to three (or more) dimensions. Unfortunately many properties are specific to the world of flat chicken, so we have to be careful. One property that holds for all rotation matrices is the value of the determinant

$$|R| = c^2 + s^2 = 1.$$

2.2.1. Rotation and Rotation Matrices

As opposed to translation where we added two vectors, in rotation we multiply the point vector with the rotation matrix. To explore our new found mathematical structure further we try to establish that the rotation is a rigid transformation. We return from the world of flat chicken back to 3-D and try to find if the distance between any pair of points P_1 and P_2 does not change with rotation

$$\|P_1 - P_2\| = \|P'_1 - P'_2\| \tag{2.2}$$

where the primed symbols are the points after the motion

$$P'_i = RP_i \tag{2.3}$$

Starting from the right hand side of Eq. (2.2) we write

$$\begin{aligned} &\|P'_1 - P'_2\| = \|RP_1 - RP_2\| = \\ &(RP_1 - RP_2)^T(RP_1 - RP_2) = (P_1 - P_2)^T R^T R(P_1 - P_2) \end{aligned} \tag{2.4}$$

If $R^T R$ is anything other than the identity matrix $\mathbf{1}$ then we cannot prove Eq. (2.2) for *all* vectors P_i. So the only way that (2.3) can represent rigid motion is if

$$R^T R = \mathbf{1} \tag{2.5}$$

which is a definitive property of rotation matrices. In fact the definition of a rotation matrix is that it satisfies Eq. (2.5) and has unit determinant. Armed with Eq. (2.5) we can continue Eq. (2.4)

$$(P_1 - P_2)^T(P_1 - P_2) = \|P_1 - P_2\|^2$$

2.2.2. Representation The rotation matrix represents a rotation in the most general form, can be extended to arbitrary (but finite) dimensions, has no singularities of any kind and is unique (if two rotation matrices represent the same rotation then the two matrices are equal). But it involves too many numbers and these numbers have no immediate physical meaning. The redundancy will give us trouble if we try to estimate them and the lack of direct physical interpretation will limit the connection to the underlying physical problem.

2.2.2.1. Fixed Angles

We have already seen the 2-D rotation in Eq. (2.1) and does not look too bad. It involves one parameter only, the angle of rotation, and the pivot of the rotation is by convention the origin. The matrix is only 2×2 and of very simple form. How about extending it to three dimensions. We all know that orientation in 3-D involves three motions: pan (left and right), tilt (up and down) and roll (what is left), or if you are a sailor roll, pitch and yaw. These correspond to rotations around the X, Y and Z axes. So forget all you learned in your math courses, head for the port and ask an old salt. Any sailor that has survived a force nine storm will tell you that the rotation around the X axis is

$$R_x(\theta_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_1 & -s_1 \\ 0 & s_1 & c_1 \end{bmatrix} \tag{2.6}$$

the rotation around the Y axis is

$$R_y(\theta_2) = \begin{bmatrix} c_2 & 0 & s_2 \\ 0 & 1 & 0 \\ -s_2 & 0 & c_2 \end{bmatrix} \tag{2.7}$$

and the rotation around the Z axis is

$$R_z(\theta_3) = \begin{bmatrix} c_3 & -s_3 & 0 \\ s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.8}$$

Now all we have to do is multiply them together and get

$$R = R_x(\theta_1) R_y(\theta_2) R_z(\theta_3)$$

which we can expand and get

$$R = \begin{bmatrix} c_2\, c_3 & -c_2\, s_3 & s_2 \\ s_1\, s_2\, c_3 + c_1\, s_3 & -s_1\, s_2\, s_3 + c_1\, c_3 & -s_1\, c_2 \\ -c_1\, s_2\, c_3 + s_1\, s_3 & c_1\, s_2\, s_3 + s_1\, c_3 & c_1\, c_2 \end{bmatrix}.$$

This matrix R certainly does not look friendly. While any Computer Scientist or Engineer can compute R numerically given the three angles (which keeping with our holy traditions we call forward kinematics), it does not seem easy to go the inverse way and compute the angles given R. But then we do like attempting the impossible when we know it is possible.

After staring at the beast for a few minutes we notice that r_{13}, the upper right element is equal to s_2, the sine of θ_2. So we can get two solutions for θ_2 and we can assume that c_2 is known (it has two possible values so we repeat the procedure twice). Armed with this we notice that

$$c_3 = \frac{r_{11}}{c_2}$$
$$s_3 = -\frac{r_{12}}{c_2}$$

and so we can find a solution for θ_3

$$\theta_3 = atan2(-\frac{r_{12}}{c_2}, \frac{r_{11}}{c_2}).$$

We can repeat the procedure for θ_1

$$s_1 = -\frac{r_{23}}{c_2}$$
$$c_1 = \frac{r_{33}}{c_2}$$

and so

$$\theta_1 = atan2(-\frac{r_{23}}{c_2}, \frac{r_{33}}{c_2}).$$

We are almost done. What if $s_2 = 1$ which makes $c_2 = 0$. It will be impossible to divide by c_2 then. The secret in Computer Vision and Robotics (as well as intergalactic travel) is DO NOT PANIC. The lower left 2×2 submatrix of R has not been used so far. It is not too late. If $s_2 = 1$ then this matrix becomes

$$\begin{bmatrix} s_1c_3 + c_1s_3 & c_1c_3 - s_1s_3 \\ s_1s_3 - c_1c_3 & s_1c_3 + c_1s_3 \end{bmatrix}$$

which can also be written as

$$\begin{bmatrix} s_{13} & c_{13} \\ -c_{13} & s_{13} \end{bmatrix}$$

where $s_{13} = \sin(\theta_1 + \theta_3)$ and $c_{13} = \cos(\theta_1 + \theta_3)$. While it does not look intimidating it does not have unique solution. We can compute

$$\theta_1 + \theta_3 = atan2(r_{21}, r_{2,2})$$

but we have infinite solutions for each individual angle. In other words a singularity. It seems that we killed this singularity rather too easily. In practice it is very hard to decide when a number is zero or one. Since the computers have finite accuracy, a zero might be represented by a very small number and an one by a number very close to but not exactly one. To compound our misery, the round-off error might be slightly different on different computers or with different compilers. The only solution is trial and error.

But before we go to the next chapter, we need to find out why is this representation called *Fixed Angles*. It does not seem likely that they were introduced by a guy named Joe Fixed, or these angles can be viewed as neutered. To see why, let us go back to the definition of R

$$R = R_x(\theta_1)R_y(\theta_2)R_z(\theta_3)$$

and observe that rotation matrices mean nothing unless we use them to rotate something. So let ${}^3P$ be a point in the third coordinate system (we need more than one since we have three rotations), which initially is coincident with the world frame. Frames 1 and 2 are also coincident with the world frame. We first rotate frame 3 around the Z axis of frame 2 (same as frame 1 and world frame) and get

$${}^2P = R_z(\theta_3)\ {}^3P.$$

We then rotate frame 2 (with frame 1 rigidly attached to it) around the Y axis of frame 1 (same as world frame) and get

$${}^1P = R_y(\theta_2)\ {}^2P = R_y(\theta_2)R_z(\theta_3)\ {}^3P.$$

And finally we rotate frame 1 around the X axis of the world frame and get

$${}^wP = R_x(\theta_1)\ {}^1P = R_x(\theta_1)R_y(\theta_2)R_z(\theta_3)\ {}^3P.$$

It is now clear that the representation is called fixed angles because each time we rotate around an axis that is *fixed* to the world coordinate system. If we were modeling the motion of a robot wrist we would rotate the last joint of the wrist first by θ_3 then the second last joint by θ_2 and finally the first joint of the wrist by an angle θ_1. Most books call this representation *Fixed Angles* $Z-Y-X$ and the angles are written as $(\theta_3, \theta_2, \theta_1)$. There are twelve variants of these like $Z-Y-Z$, $X-Y-Z$, $X-Y-X$, etc.

2.2.2.2. Euler Angles It is easy to rotate around a cardinal axis using the notation in Eqs. (2.6), (2.7) and (2.8). But if we reverse the order of the operations in the wrist example above and rotate the first joint of the wrist first, when we rotate the second joint we are no longer rotating around the original Y axis but a rotated version of it. And this sounds like an awfully complex thing to do. But it is not. After you have rotated the three joints of a wrist it does not matter which one you rotated first. Any backhoe operator can verify this for you but a mathematician can prove it. So the representation above is also *Euler Angles* $X-Y-Z$ and the angles are written as $(\theta_1, \theta_2, \theta_3)$. There are another 12 variants of these too.

The Euler angles are by far more popular! The reasons are that we can sound more scientific (very fashionable for almost two centuries now) without doing much, confuse the uninitiated, and avoid invoking feelings of guilt to typical dog and cat owners.

2.2.2.3. Cayley's Formula, Rodrigues Parameters, Quaternions and Lie Algebras

All of these are based on the same idea which with successive rumination has given us all the above concepts. Since they belong to the *seen-one-seen-them-all* class of ideas we treat them together. So we start with Cayley and his ground shaking formulas (there are two). Let S be

$$S = (R - \mathbf{1})(R + \mathbf{1})^{-1} \tag{2.9}$$

where R is as always a rotation matrix. One can prove that

$$S^T = -S.$$

S is in other words a skew symmetric matrix. It is fairly easy to do so. First notice that

$$S = (R - \mathbf{1})(R + \mathbf{1})^{-1} = R(R + \mathbf{1})^{-1} - (R + \mathbf{1})^{-1}$$

and so

$$\begin{aligned} S_T &= (R^T + \mathbf{1})^{-1} R^T - (R^T + \mathbf{1})^{-1} = \\ &(R^T + \mathbf{1})^{-1} R^{-1} - (\mathbf{1} R^T + R R^T)^{-1} = \\ &\left(R(R^T + \mathbf{1}) \right)^{-1} - \left((\mathbf{1} + R) R^{-1} \right)^{-1} = \\ &(\mathbf{1} + R)^{-1} - R(R + \mathbf{1})^{-1} = -S \end{aligned}$$

and if you are not feeling the ground shaking wait for the second installment of Cayley's formula:

$$R = (\mathbf{1} - S)^{-1}(\mathbf{1} + S) \tag{2.10}$$

in other words from S we can get back R. This establishes the one to one mapping between skew symmetric matrices like S and rotation matrices like R. So given a 3×3 rotation matrix that has 9 elements we can compress it down to a skew symmetric matrix that has only three (independent) elements. And a 4×4 rotation matrix can be compressed down to 6 numbers (something to know if your four dimensional relatives from Andromeda invite themselves and ask for directions to your home). The proof that the R given by Eq. (2.10) is the same as the one in Eq. (2.9) is straightforward but tedious. You have to substitute Eq. (2.9) into Eq. (2.10) and massage the expression until you get R. It is a good exercise.

Matrix S being skew symmetric has the form

$$S = \begin{bmatrix} 0 & -s_z & s_y \\ s_z & 0 & -s_x \\ -s_y & s_x & 0 \end{bmatrix}$$

and the three scalars s_x, s_y and s_z are called *Rodrigues parameters* and are often organized in a vector s

$$s = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix}$$

which has some nice properties. It is the axis of the rotation, which one can show easily by proving the $Rs = s$ and has length equal to the tangent of the half angle of the rotation which is slightly harder to show. Unfortunately, when the angle of rotation is 180 degrees, the half angle is 90 and the tangent is infinite. So the Rodrigues parameters cannot

represent 180 degree rotations.

This little nuisance can be corrected by the use of quaternions. These are four dimensional relatives of the Rodrigues parameters (not the kind that invite themselves to dinner) developed by Sir William Rowan Hamilton sometime in the nineteenth century. A quaternion is defined as

$$q = \frac{1}{\sqrt{1 + s_x^2 + s_y^2 + s_z^2}} \begin{bmatrix} s_x \\ s_y \\ s_z \\ 1 \end{bmatrix}.$$

The quaternions are unit vectors (at least when used to describe rotation) and can represent any rotation including rotations by 180 degrees, as long as one does not try to compute them using Rodrigues parameters as intermediate. The quaternions were proposed partly as a 4 dimensional extension to the complex numbers since this is what was hot back then. Quaternions are still used in some engineering applications for the several advantages they provide: they do not involve trigonometric functions, are more compact than rotation matrices, have no singularities, are numerically stable and very elegant. But their popularity is nowhere near what Sir Hamilton was hoping when he authored his 800 page strong book “Elements of Quaternions”. They were largely replaced by the much more convenient vector calculus by the middle of the twentieth century.

Shortly after the development of the quaternion *Lie Algebras* were in vogue and Cayley’s formula was once again employed to generalize geometric transformations. Lie are always rumored to have applications in a field other than your own.