Introduction to Homogeneous Transformations & Robot Kinematics

Jennifer Kay, Rowan University Computer Science Department

January 2005

1. Drawing 3 Dimensional Frames in 2 Dimensions

We will be working in 3-D coordinates, and will label the axes **x**, **y**, and **z**. Figure 1 contains a sample 3-D coordinate frame.

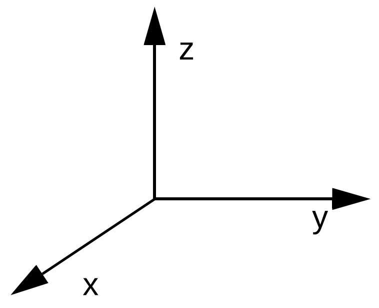


Figure 1. A 3-D coordinate frame.

Because we are representing 3-D coordinate frames with 2-D drawings, we have to agree on what these drawings mean. Clearly the y axis in Figure 1 points to the right, and the z axis points up, but we have to come up with a convention for what direction the x axis is pointing. Since the three axes must be perpendicular to each other, we know that the x axis either points into the paper, or out of the paper. Most people instantly assume one or the other is the case. To be able to view both cases, it helps to look at the axes overlaid on a cube. Consider the two views of the same cube in Figure 2. In view (a) we are looking at the cube from below, in view (b) we are looking at the cube from above. Let's try and overlay the 3-D coordinate frame from Figure 1 onto these two views. Before you turn the page, make sure you can see both views of the cube in Figure 2!

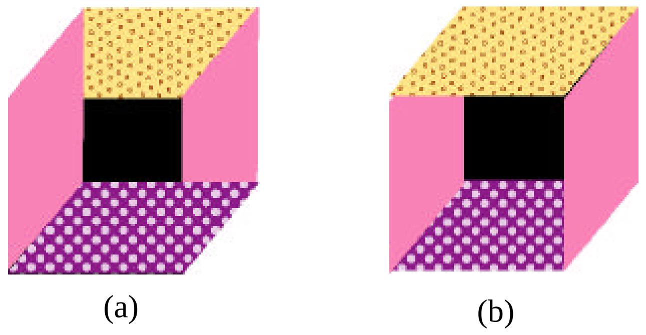


Figure 2. Two views of the same cube. The cube is missing the front side, has a solid back and sides, and patterned top and bottom. In view a we are looking at the cube from below, in view (b) we are looking at the cube from above.

Figure 3 and Figure 4 show the same two views of the cube, this time with the 3-D coordinate frame from Figure 1 overlaid onto the cube. Note that in Figure 3 the x axis points into the paper, away from you, and in Figure 4 the x axis is pointing out of the paper towards you!

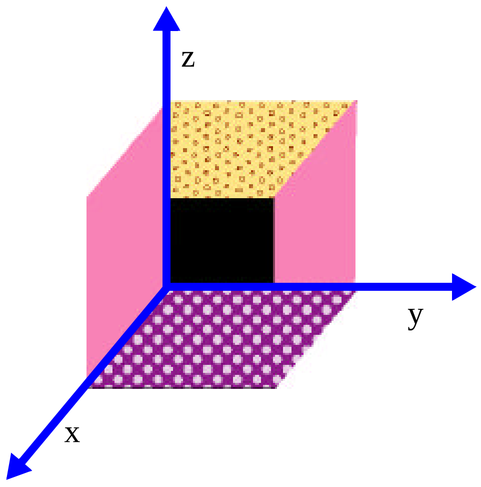


Figure 3. A cube viewed from below. The edge labelled x points into the page away from you.

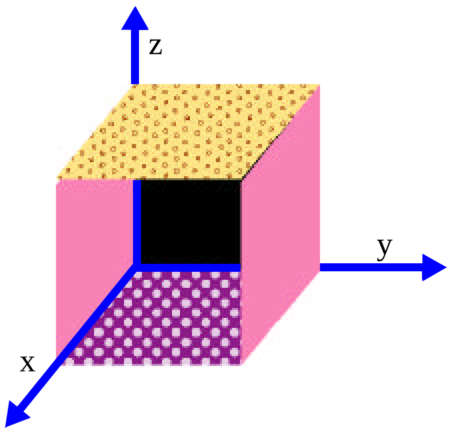


Figure 4. The same cube viewed from above. The edge labelled x points out of the page towards you.

For the purposes of this document, we will assume that Figure 4 shows the interpretation we will use. In other words, if you see 3 axes drawn as they are in Figure 1, you should assume that the x axis points out of the paper towards you. If you actually wanted the x axis to be pointing into the paper, you should use the illustration shown in Figure 5.

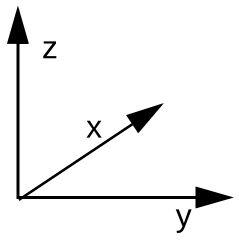


Figure 5. Another 3-D coordinate frame. In this frame, y points to the right, z points up, and x points away from you into the paper.

2. Right Handed Coordinate Systems

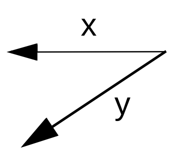
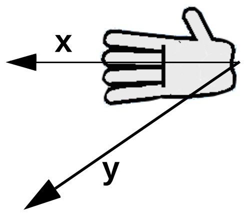
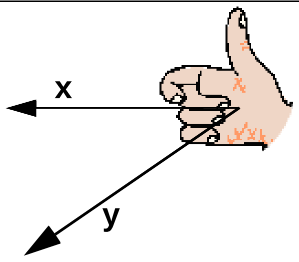


Figure 6. x points to the left, and y points out of the paper towards you.

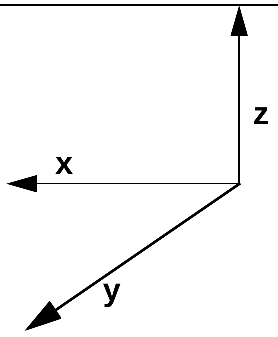
Most of the time we are going to use *right handed* coordinate systems. In a right handed coordinate system, if you know the directions of two out of the three axes, you can figure out the direction of the third. Let's suppose that you know the directions of the x and y axes. For example, suppose that x points to the left, and y points out of the paper, as shown in Figure 6. We want to determine the direction of the z axis. To do so, take your **right hand**, and hold it so that your fingers point in the direction of the x axis in such a way that you can curl your fingers towards the y axis.When you do this, your thumb will point in the direction of the z axis. This process is illustrated in Figure 7. The chart in figure 7 details how to compute the direction of any axis given the directions of the other two.



Step 1: hold your right hand in such a way that your fingers point in the direction of the x axis and when you curl your fingers, they curl towards the y axis.



Step 2: As you curl your fingers from the x axis towards the y axis, stick your thumb in the air. This will be the direction of the z axis



The final 3-D coordinate system with the z axis shown.

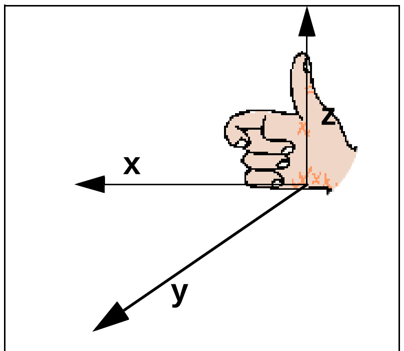
Figure 7. Using the right hand rule to compute the direction of the z axis.

3. Direction of Positive Rotation

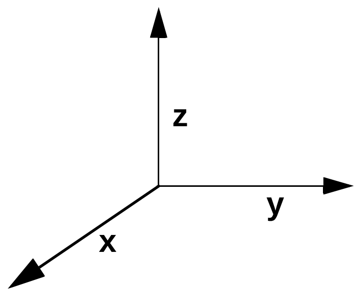
Sometimes we want to talk about rotating around one of the axes of a coordinate frame by some angle. Of course, if you are looking down an axis and want to spin it, you need to know whether you should spin it clockwise or counter-clockwise. We are going to use another right hand rule to determine the direction of positive rotation.

If you know the direction of these axes.	Point the fingers of your right hand in the direction of this axis.	Curl you right fingers towards the direction of this axis.	Your thumb will point in the direction of this axis.
x & y	x	y	z
y & z	y	z	x
x & z	z	x	y

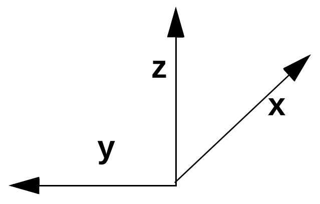
Figure 8. Using the right hand rule to compute the direction of any axis given the directions of the other two.



(a) The original axes with a right hand determining the direction of positive rotation around the z axis.



(b) After rotating *the original* axes (a) 90^o around the z axis.



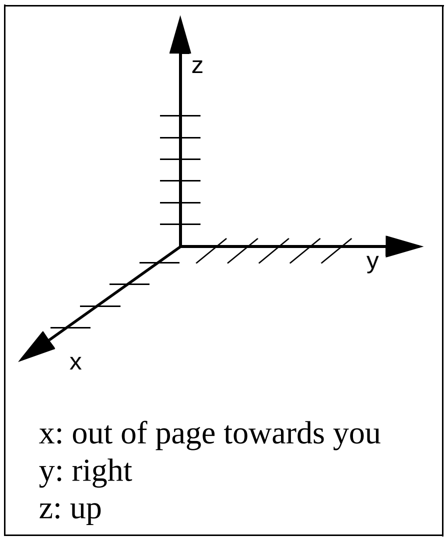
(c) After rotating *the original* axes (a) -90^o around the z axis.

Figure 9. The right hand rule to determine the direction of positive angles. Point your right thumb along the positive direction of the axis you wish to rotate around. Curl your fingers. The direction that your fingers curl is the direction of positive rotation.

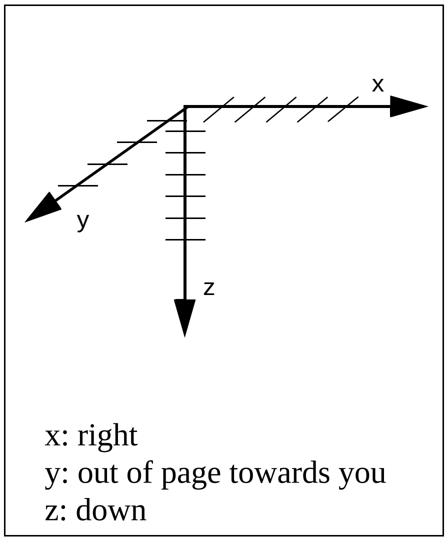
4. Plotting Points in 3 Dimensions

All of us have experience in plotting points on 2-D axes. When it comes to plotting points on 3-D axes, things become a bit more difficult. In this section, we will discuss how to plot a number of points on the 3-D axes presented in Figure 1.

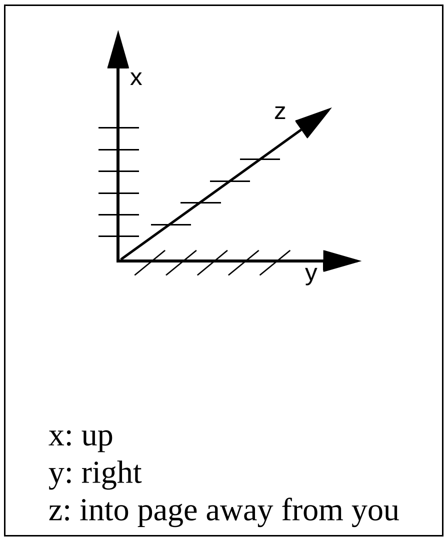
The first step is to draw tick marks on the axes to indicate scale. For the purposes of this document, we will assume that each tick represents one unit. Figure 10 shows several different right handed coordinate systems with tick marks added. Note that each tick mark is parallel to one of the other axes. This helps the viewer to visualize the 3-D effect.



x: out of page towards you
y: right
z: up

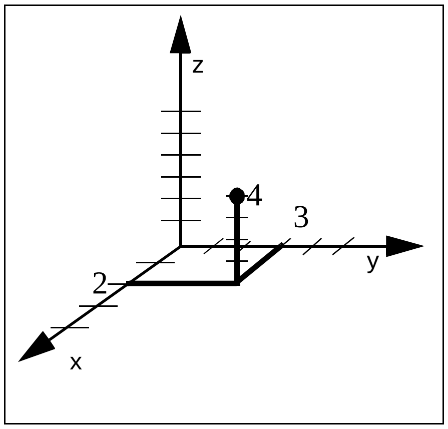


x: right
y: out of page towards you
z: down

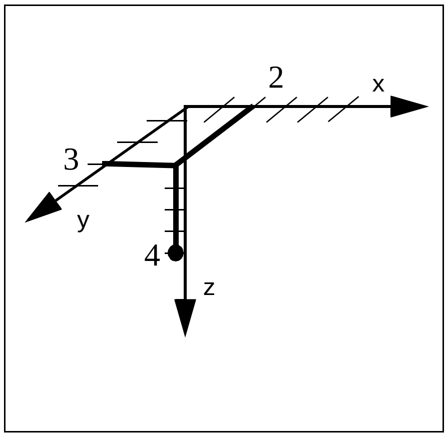


x: up
y: right
z: into page away from you

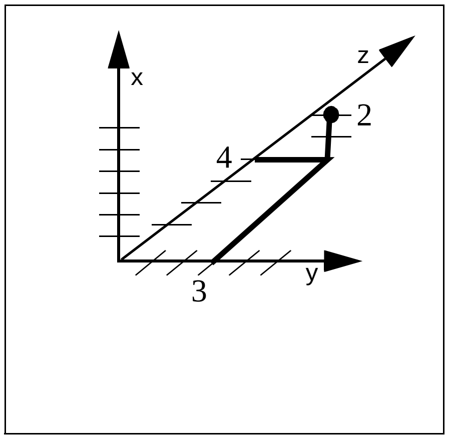
Figure 10. Several different right handed 3-D coordinate frames with tick marks to indicate scale.



(a)



(b)



(c)

Figure 11. The point (2,3,4) plotted on different axes.

Figure 11 shows the point (2,3,4) plotted on the different axes of Figure 10. The technique is quite straightforward if two of your axes form a plane parallel with the ground. First, draw lines to indicate the projection of the point on that plane. Then, draw a line through that point that is parallel to the remaining axis, add tick marks to it, and plot your point.

For example, in Figure 11 (a) the x and y axes form the *groundplane*, and so we draw lines to indicate where (2,3,0) would be. Then, we draw a line through the point (2,3,0) that is parallel to the z axis, add tick marks to it, and finally plot our point. Although Figure 11 (b) looks different, the x and y axes still form the groundplane and so the procedure is virtually the same. The only difference is that the tick marks on the z axis have been left out because when they are included they are difficult to distinguish from the tick marks on the vertical line that connects to the point (2,3,4). In Figure 11 (c), the y and z axes form the groundplane. Thus, we first plot the point (0, 3, 4), then draw a line through the point (0,3,4) parallel to the x axis, add tick marks to it, and again plot our point (2,3,4).

5. Working With Multiple Coordinate Frames

Sometimes we may want the coordinates of a point given with respect to two or more coordinate frames. For example, consider an airport scenario. The coordinate frame of the airport might have its origin at the base of the control tower, with the x axis pointing North, the y axis pointing West, and the z axis pointing up. While this is a useful coordinate frame for the air traffic controllers to use, a pilot may be more interested in where objects are relative to her airplane. Thus, we might have two coordinate frames, “tower coordinates” and “plane coordinates” as illustrated in Figure 12. We use subscripts to distinguish between x_t, y_t, and z_t (the tower coordinate frame) and x_p, y_p, and z_p (the plane coordinate frame).

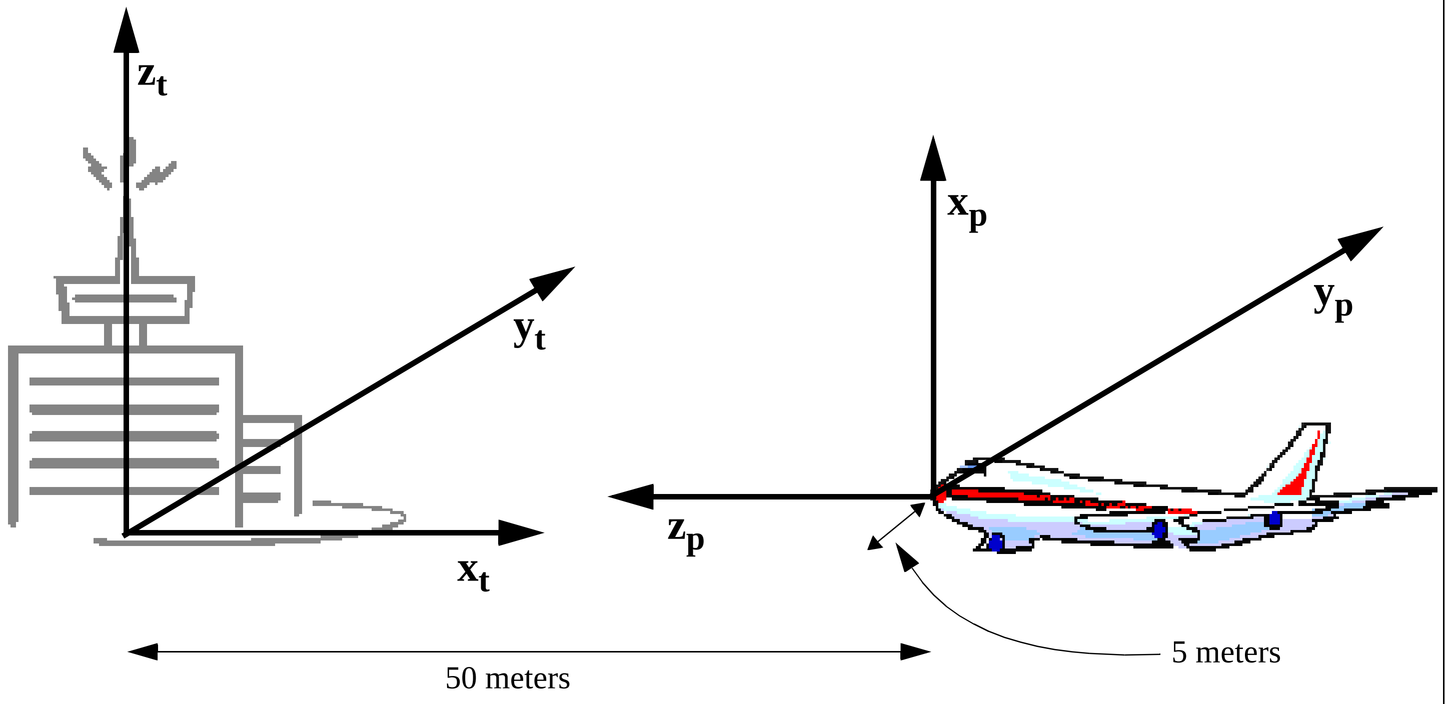


Figure 12. The plane is on the ground preparing for takeoff. Tower coordinates are centered at the base of the air traffic control tower. x points North, y points West, and z points up. The airplane coordinate system is centered at the nose of the plane. The x axis points towards the top of the plane, the y axis points out to the right as you are sitting in the captain’s chair, and the z axis points straight out the front of the plane.

When the plane is stopped on the runway as depicted in Figure 12, the nose of the plane might be at location (50, 5, 0) in tower coordinates, but it is at the origin (location (0, 0, 0)) in plane coordinates. Similarly, the base of the tower is at location (0, 0, 0) in tower coordinates, but at location (0, -5, 50) in plane coordinates.

Note that the location of the nose of the plane is fixed with respect to the plane, but not with respect to the tower. When the plane begins to take-off as depicted in Figure 13, its nose is still at location (0,0,0) in plane coordinates, but it is at location (30, 15, 5) in tower coordinates.

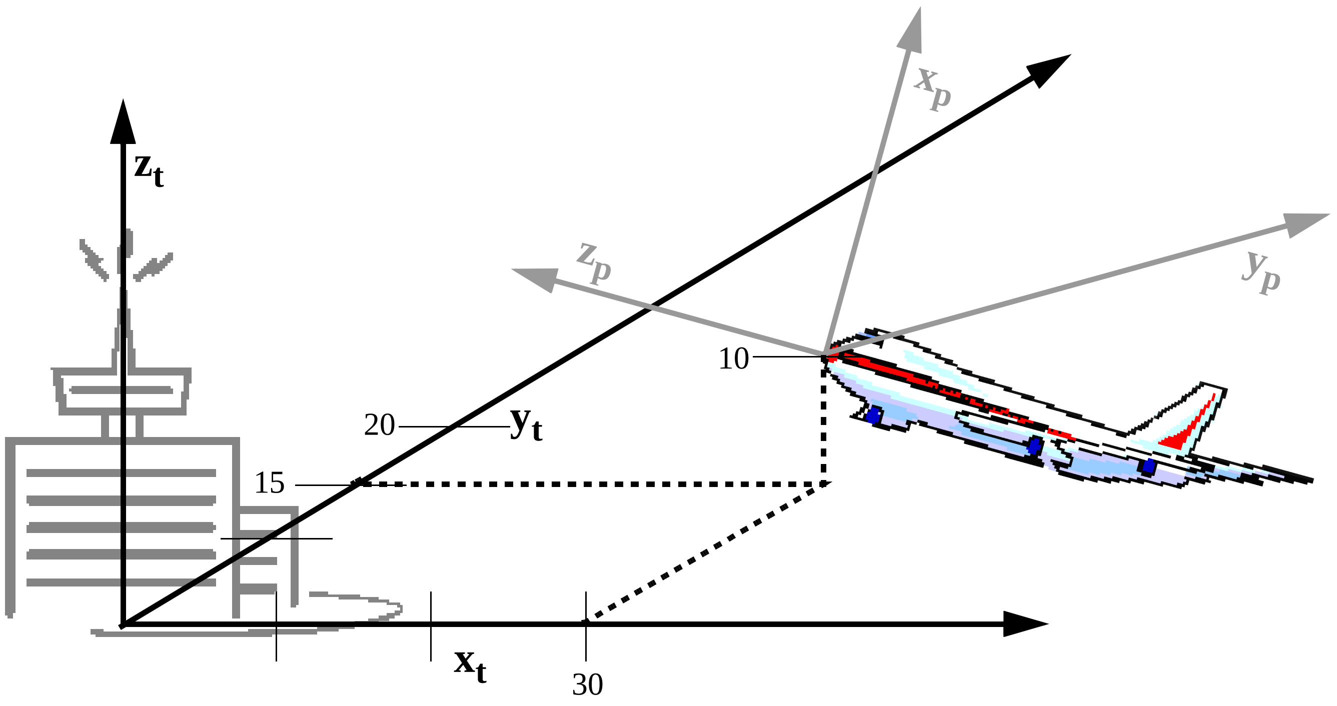


Figure 13. When the plane takes-off, its nose is still at location (0,0,0) in plane coordinates, but it is at a different location in tower coordinates. To determine where it is in tower coordinates, the tower's x and y axes have been extended, and the nose of the plane has been plotted in the manner of Figure 11. Thus we see that the nose of the plane is at location (30, 15, 10) in tower coordinates.

6. Homogeneous Transformations

6.1 Representing Points

Up to this point, we have been using the traditional (x,y,z) notation to represent points in 3-D. However, for the remainder of this document, we are going to use a vector notation to represent points. The point (x,y,z) is represented as the vector $\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$. The 1 is a weighting factor. So the vectors $\begin{bmatrix} 2x \\ 2y \\ 2z \\ 2 \end{bmatrix}$ and $\begin{bmatrix} 257x \\ 257y \\ 257z \\ 257 \end{bmatrix}$ all represent that same point (x, y, z). Most of the time we will simply use a weighting factor of 1. Consider the more concrete example depicted in Figure 13. The plane is at location (30, 15, 10) in tower coordinates. We will normally represent that location as $\begin{bmatrix}30\\15\\10\\1\end{bmatrix}$, however it can also be represented as $\begin{bmatrix}60\\30\\20\\2\end{bmatrix}$, $\begin{bmatrix}-30\\-15\\-10\\-1\end{bmatrix}$, $\begin{bmatrix}75\\37.5\\25\\2.5\end{bmatrix}$, and an infinite number of other vectors.

In order to save space in this document, we will often write the point $\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$ as $\begin{bmatrix}x & y & z & 1\end{bmatrix}^T$ (the transpose of $\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$).

6.2 The Use of Multiple Coordinate Frames in Robotics

It is very common in robotics to use two or more coordinate frames to solve a problem. Suppose the airplane in Figure 12 were automatically controlled. It would be very useful to keep track of some things in tower coordinates. For example, the altitude of the plane is simply the z coordinate of its location in tower coordinates. It also would be useful to keep track of other things in airplane coordinates. For example, the direction the plane should head to in order to avoid a mountain. Indeed, for many mobile robot applications, it is desirable to know the locations of objects in both “world coordinates” and “robot coordinates.”

Multiple coordinate frames are also useful in traditional robotics. For example, consider the simple robot arm depicted in Figure 14. If we want to have the gripper pick up a widget off of a table, then we need to figure out the widget’s location. Perhaps we have a camera that we use to initially determine the location of the widget (in camera coordinates). We might need to transform that location into world coordinates to evaluate if it is accessible to the robot at all, and to gripper coordinates to determine when we should close the jaws of the gripper.

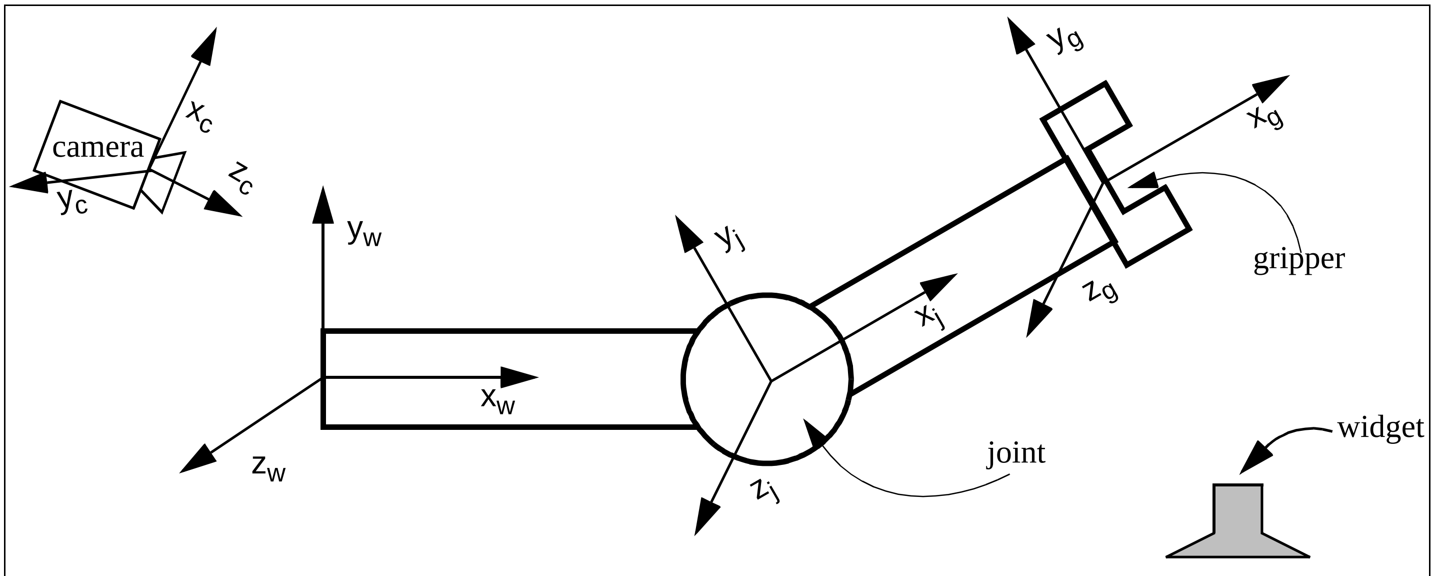


Figure 14. A very simple robot arm with one joint and one gripper. The world, camera, joint, and gripper coordinate frames are indicated.

6.3 Transforming Points Between Coordinate Frames

Suppose that you know the location of a point in one coordinate frame (for example, airplane coordinates) and you want to know its location in another frame (for example, tower coordinates). How do you do it? We will start with a very simple case, and then move into more complex examples.

Let's begin by considering the two coordinate systems in Figure 15, world coordinates and robot coordinates. Notice that the only difference between the two coordinate frames is that the robot frame has been translated by 3 units along the y axis from the world coordinate frame.

Figure 16 is a table of some sample points in world coordinates, and their corresponding values in robot coordinates. For the moment, ignore the third column of Figure 16, and just look at the first two columns. Notice that any point $\begin{bmatrix} a & b & c & 1 \end{bmatrix}^T$ in world coordinates is the same as the point $\begin{bmatrix} a & (b-3) & c & 1 \end{bmatrix}^T$ in robot coordinates. Similarly, any point $\begin{bmatrix} d & e & f & 1 \end{bmatrix}^T$ in robot coordinates is the same as the point $\begin{bmatrix} d & (e+3) & f & 1 \end{bmatrix}^T$ in world coordinates.

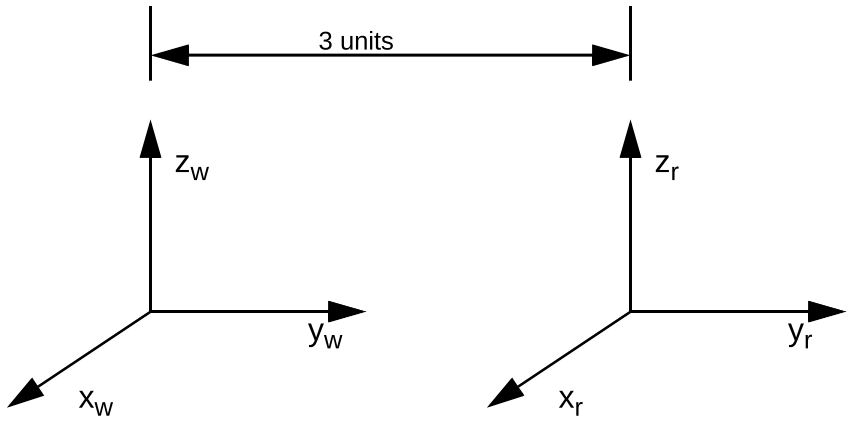


Figure 15. Two coordinate frames, world (w) and robot (r). The origin of the robot frame is located at the point (0,3,0) in world coordinates. Using our new notation, we can represent this as $\begin{bmatrix} 0 & 3 & 0 & 1 \end{bmatrix}^T$. The two x axes are parallel, as are the two y and the two z axes. Note that the origin of the world coordinate axes is at $\begin{bmatrix} 0 & -3 & 0 & 1 \end{bmatrix}^T$ in robot coordinates.

We have seen that one way to convert world coordinates to robot coordinates for the system in Figure 15 is to subtract 3 from the y value in world coordinates. Surprisingly, another way to convert points from the world coordinate frame of Figure 15 to the robot coordinate frame of Figure 15 is to pre-multiply the point by the matrix $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. The third column of Figure 16 does exactly this and results in the same answer!

Location of a Point in World Coordinates of Figure 15	Location of the Same Point in Robot Coordinates of Figure 15	Pre-multiplying the point in world coordinates by $\begin{bmatrix}1&0&0&0\\0&1&0&-3\\0&0&1&0\\0&0&0&1\end{bmatrix}$
$\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$	$\begin{bmatrix}0\\-3\\0\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\0&1&0&-3\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\end{bmatrix}=\begin{bmatrix}0\\-3\\0\\1\end{bmatrix}$
$\begin{bmatrix}0\\3\\0\\1\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\0&1&0&-3\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0\\3\\0\\1\end{bmatrix}=\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$
$\begin{bmatrix}5\\10\\15\\1\end{bmatrix}$	$\begin{bmatrix}5\\7\\15\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\0&1&0&-3\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}5\\10\\15\\1\end{bmatrix}=\begin{bmatrix}5\\7\\15\\1\end{bmatrix}$
$\begin{bmatrix}84\\84\\84\\1\end{bmatrix}$	$\begin{bmatrix}84\\81\\84\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\0&1&0&-3\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}84\\84\\84\\1\end{bmatrix}=\begin{bmatrix}84\\81\\84\\1\end{bmatrix}$
$\begin{bmatrix}4\\-4\\4\\1\end{bmatrix}$	$\begin{bmatrix}4\\-7\\4\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\0&1&0&-3\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}4\\-4\\4\\1\end{bmatrix}=\begin{bmatrix}4\\-7\\4\\1\end{bmatrix}$
$\begin{bmatrix}a\\b\\c\\1\end{bmatrix}$	$\begin{bmatrix}a\\b-3\\c\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\\0&1&0&-3\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}a\\b\\c\\1\end{bmatrix}=\begin{bmatrix}a\\b-3\\c\\1\end{bmatrix}$

Figure 16. Converting between world and robot coordinates as depicted in Figure 15.

It should be clear that there is nothing magic about the number -3 in our 4x4 matrix other than the fact that we moved 3 units along the y axis between the two frames. Indeed, if we had moved 63 units, then pre-multiplying by the matrix $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -63 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ would convert points from world coordinates to robot coordinates.

Similarly, there's nothing magic about the fact that we did this move along the y axis. We could come up with similar matrices for changes in frames that occurred along the y or z axis. Or indeed any combination of moves along all three matrices.

Suppose that to get from the coordinate frame p to the coordinate frame q you need to move a units along p's x axis, b units along p's y axis, and c units along p's z axis. Then to take a point from q coordinates to p coordinates, you need to premultiply it by the matrix $\begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$ For example, consider the two coordinate frames of Figure 17. To transform the robot coordinate frame into the world coordinate frame, you need to translate 5 units along the robot's x axis, -4 units along the robot's y axis, and -1 unit along the robot's z axis. Thus to take a point in world coordinates and transform that into a point in robot coordinates, you need to premultiply that point by the matrix $\begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. Some examples of this can be found in Figure 18 .

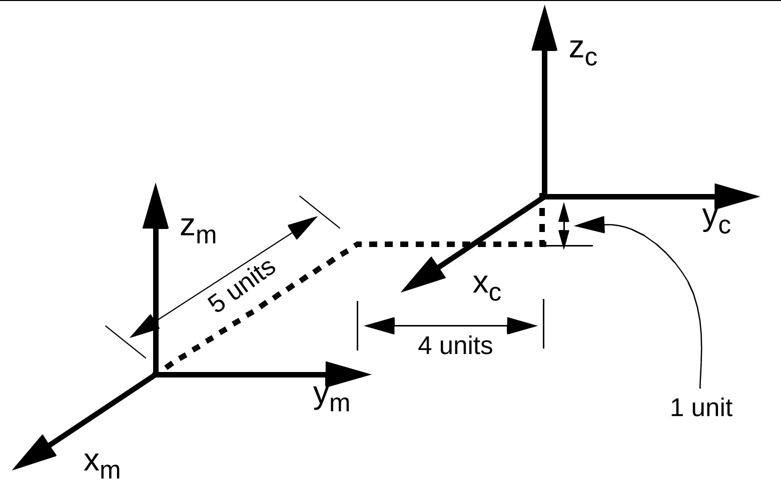


Figure 17. Two coordinate frames that differ by only a translation. To get from car coordinates (c) to mountain coordinates (m) you must translate 5 units along the car's x axis, -4 units along the car's y axis, and -1 unit along the car's z axis.

Location of a point in Figure 17's mountain coordinates	Location of the same point in Figure 17's car coordinates	Pre-multiplying the point in mountain coordinates by $\begin{bmatrix}1&0&0&5\\0&1&0&-4\\0&0&1&-1\\0&0&0&1\end{bmatrix}$
$\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$	$\begin{bmatrix}5\\-4\\-1\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&5\\0&1&0&-4\\0&0&1&-1\\0&0&0&1\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\end{bmatrix}=\begin{bmatrix}5\\-4\\-1\\1\end{bmatrix}$
$\begin{bmatrix}0\\3\\0\\1\end{bmatrix}$	$\begin{bmatrix}5\\-1\\-1\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&5\\0&1&0&-4\\0&0&1&-1\\0&0&0&1\end{bmatrix}\begin{bmatrix}0\\3\\0\\1\end{bmatrix}=\begin{bmatrix}5\\-1\\-1\\1\end{bmatrix}$
$\begin{bmatrix}5\\10\\15\\1\end{bmatrix}$	$\begin{bmatrix}10\\6\\14\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&5\\0&1&0&-4\\0&0&1&-1\\0&0&0&1\end{bmatrix}\begin{bmatrix}5\\10\\15\\1\end{bmatrix}=\begin{bmatrix}10\\6\\14\\1\end{bmatrix}$
$\begin{bmatrix}84\\84\\84\\1\end{bmatrix}$	$\begin{bmatrix}89\\80\\83\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&5\\0&1&0&-4\\0&0&1&-1\\0&0&0&1\end{bmatrix}\begin{bmatrix}84\\84\\84\\1\end{bmatrix}=\begin{bmatrix}89\\80\\83\\1\end{bmatrix}$
$\begin{bmatrix}4\\-4\\4\\1\end{bmatrix}$	$\begin{bmatrix}9\\-8\\3\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&5\\0&1&0&-4\\0&0&1&-1\\0&0&0&1\end{bmatrix}\begin{bmatrix}4\\-4\\4\\1\end{bmatrix}=\begin{bmatrix}9\\-8\\3\\1\end{bmatrix}$
$\begin{bmatrix}a\\b\\c\\1\end{bmatrix}$	$\begin{bmatrix}a+5\\b-4\\c-1\\1\end{bmatrix}$	$\begin{bmatrix}1&0&0&5\\0&1&0&-4\\0&0&1&-1\\0&0&0&1\end{bmatrix}\begin{bmatrix}a\\b\\c\\1\end{bmatrix}=\begin{bmatrix}a+5\\b-4\\c-1\\1\end{bmatrix}$

Figure 18. Converting points from the mountain coordinate frame of Figure 17 to the car coordinate frame of Figure 17.

We call the matrix that converts a point from j coordinates to k coordinates the *homogeneous transformation from j coordinates to k coordinates*, and we abbreviate this as T_j^k. Figure 19 summarizes how to compute the matrix that converts between two frames that only differ by a translation.

If the coordinate frames j and k only differ by a translation and to get from k coordinates to j coordinates you translate (a, b, c) along k's x, y, and z axes, then T_j^k, the matrix that takes a point in j coordinates to a point in k coordinates is $\begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$

We can summarize this with the equation

$$\boxed{\text{Trans (a,b,c)} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

Figure 19. Converting points between coordinate frames that only differ by a translation.

6.4 Review: Determining the Homogeneous Transformation when Frames Differ only by Translation

It is very important to note that we have been considering two distinct concepts in our previous discussion:

1. How do we take a point that is in frame a-coordinates and convert it to frame b-coordinates? (In other words, what is T_a^b
2. How do we compute the transformation between frame a and frame b (i.e. how would we move frame a to line it up with frame b)? (We don't have an abbreviation for this yet).

The two concepts are closely related, but not the same. If we want to know how to take a point in frame a coordinates and convert it to frame b coordinates, the easiest way to do that is to first compute how to move frame b so that it lines up with frame a.

Now go look at Figure 19 again! Make sure you understand the TWO concepts before you continue on.

In order to remind ourselves that there are TWO concepts, we will use two different sets of notation to represent them. We have already discussed T_a^b, the transformation that takes a point in a-coordinates and computes its location in b-coordinates. For the remainder of our discussion, we will use the notation F_a^b to represent the transformation that moves the a-coordinate frame into alignment with the b-coordinate frame. Note that the F_a^b notation is non-standard. We summarize the notation in Figure 20.

T_j^k	The transformation that you use to take a point in j-coordinates and compute its location in k-coordinates
F_j^k	The transformation that you use to take the j coordinate frame and move it in such a way that it aligns with the k coordinate frame. Note that this is a non-standard notation.
$T_j^k = F_k^j$	Relationship between T & F

Figure 20. Summary of our notation.

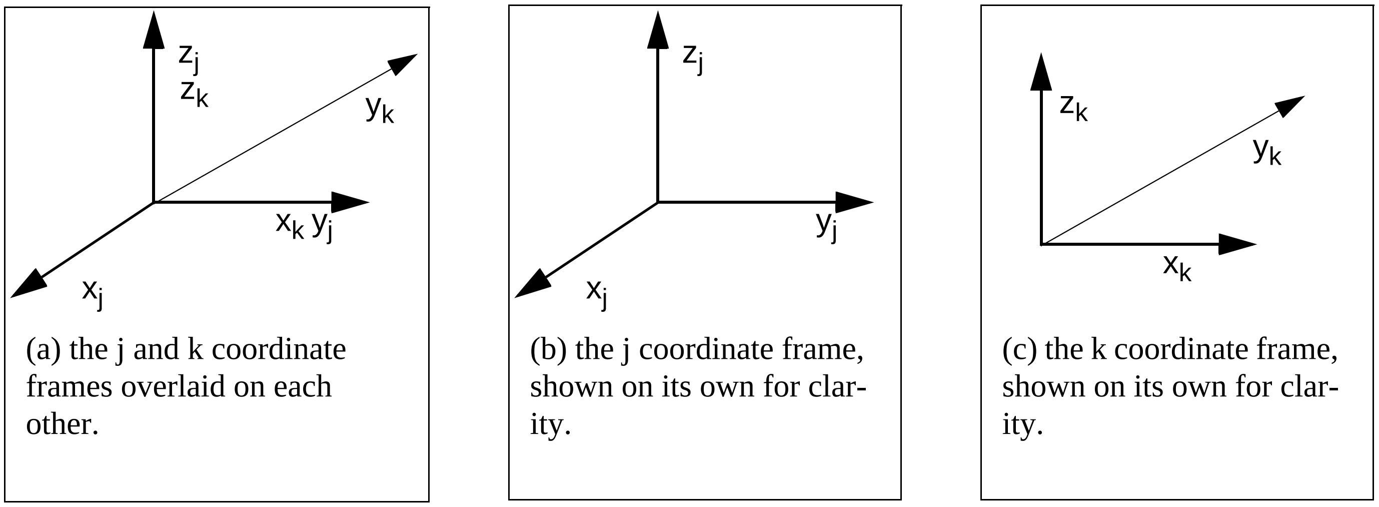
6.5 Coordinate Frames that Differ by a Rotation Around One Axis

Consider the two frames depicted in Figure 21. To transform the k coordinate frame into the j coordinate frame (F_k^j), we perform a rotation about k’s z axis by -90^o. By looking at Figure 21 (a), we can see that $x_k = y_j$, $z_k = z_j$, and $x_j = -1*y_k$. Let’s look at a few examples. The origin of the j axis in j coordinates: the point $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ is the same as the origin of the j axis in k coordinates $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$. The point $\begin{bmatrix} a & b & c & 1 \end{bmatrix}^T$ in j coordinates is located at

$\begin{bmatrix} b & -a & c & 1 \end{bmatrix}^T$ in k coordinates. Again, there is a matrix that we can premultiply points in j coordinates by to transform them into points in k coordinates. If to transform the k coordinate frame into the j coordinate frame F_k^j you rotate about the z axis by θ, then you can pre-multiply a point in j coordinates by the matrix

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (T_j^k)$$ to get the location of the point in k coordinates .

Let's check this on the two examples that we've done already.



(a) the j and k coordinate frames overlaid on each other.

(b) the j coordinate frame, shown on its own for clarity.

(c) the k coordinate frame, shown on its own for clarity.

Figure 21. Two coordinate frames, j and k, that differ only by a rotation about the z axis. To transform the k coordinate frame into the j coordinate frame, we rotate by -90^o about k's z axis.

In our example, θ is 90^o. $\cos -90^o = 0$. $\sin -90^o = -1$. So our matrix becomes . When we compute

our matrix * $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$, we get $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ as expected. When we multiply our matrix * $\begin{bmatrix} a & b & c & 1 \end{bmatrix}^T$ we get

$\begin{bmatrix} b \\ -a \\ c \\ 1 \end{bmatrix}$ as expected.

At this point we know that we can design a matrix to convert points between two coordinate frames that only differ by a translation, or by a rotation about the z axis. It should not surprise you to learn that you can also design matrices to convert points between two coordinate frames that only differ by a rotation about the x or y axes too.

6.6 Putting it All Together

So far we have learned how to create the matrix that will compute the coordinates of a point in one coordinate frame given the coordinates of that point in another coordinate frame, subject to the following condition: the two frames may only differ by a translation (along the 3 axes), or by a rotation about one axis. It is indeed possible to convert points between two coordinate frames that differ, perhaps by two translations, or by a rotation then a translation and then another rotation.

The key to understanding how to do this is to understand that there are two ways to view any sequence of translations and rotations. The first way we will call “moving” coordinate systems. In moving coordinate systems, each step happens relative to the steps that have come before it. For example, Figure 23 shows the world and gripper coordinate frames for a particular robotic system. Note that not only is the gripper coordinate frame translated from the world coordinate frame, but there also must be some sort of rotation that caused the gripper’s x axis to point up instead of out of the page towards you as the world coordinates do.

In the “moving axes” approach, we say that to get from world coordinates to gripper coordinates, (F^g_w)you need to do the following sequence of moves: Rotate about x_w by -90 degrees. Call this new frame intermediate frame 1, and we’ll call its axes x_1, y_1, and z_1. Next rotate about the new z_1 by -90 degrees. Call this new frame intermediate frame 2, and we’ll call its axes x_2, y_2, and z_2. Finally, translate by (0,0,5) relative to intermediate frame 2. This results in the gripper coordinate frame.

The alternative approach is the “fixed axes” approach. In this technique, all of your moves are relative to the <u>original</u> world coordinate frame. In the “fixed axes” approach, the picture is still as depicted in Figure 23, but this time the sequence of steps is: Rot x_w(-90) then Rot y_w (-90), then Trans(0,5,0) relative to world coordinates.

Whether you choose to use the “moving axes” approach or the “fixed axes” approach, your final matrix, F^g_w, will be the same. However the way you compute it will differ.

If to convert the k coordinate frame into the j coordinate frame you have to: F^j_k	Then to convert a point in j coordinates into a point in k coordinates, premultiply that point by: T^k_j	The nickname for this transformation is:
Translate along k's x axis by a, along k's y axis by b, along k's z axis by c	$\begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Trans (a, b, c)
Rotate about k's x axis by θ	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Rot x (θ)
Rotate about k's y axis by θ	$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Rot y (θ)
Rotate about k's z axis by θ	$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Rot z (θ)

Figure 22. Summary of transformation matrices

6.7 Computing the Transformation Matrix Using Moving Axes

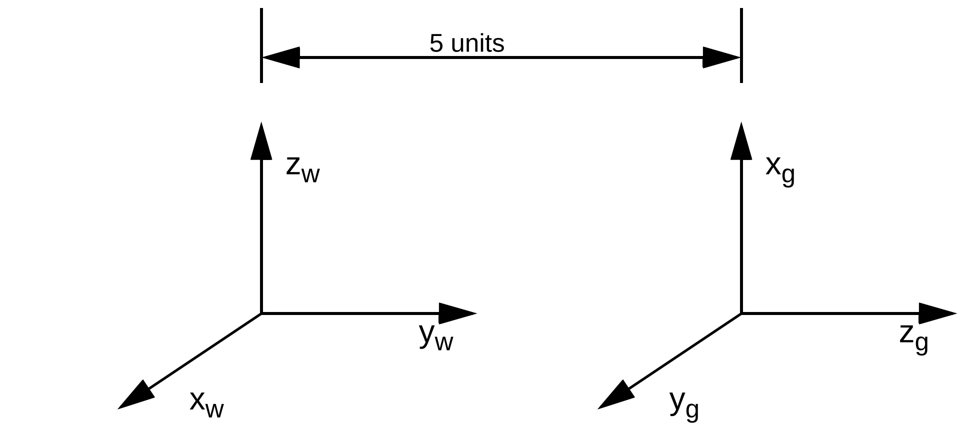


Figure 23. World and gripper coordinate frames for some robot. There are several ways we can think about transforming between the two coordinate frames.

Recall that the moving axes approach is as follows: to get from world coordinates to gripper coordinates, F^g_w, you need to do the following sequence of moves: Rotate about x_w by -90 degrees. Call this new frame intermediate frame 1, and we'll call its axes x_1, y_1, and z_1. Next rotate about the new z_1 by -90 degrees. Call this new frame intermediate frame 2, and we'll call its axes x_2, y_2, and z_2. Finally, translate by (0,0,5) relative to intermediate frame 2. This results in the gripper coordinate frame.

When we use "moving axes" we list the moves that we did from left to right, compute the individual matrices for each part, and then multiply them together. For example, in this situation, our sequence of equations is:

Rot x(-90) * Rot z(-90) * Trans(0,0,5)

The matrices for this product are as follows: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

The resulting matrix, F^g_w, will transform a point from gripper coordinates to world coordinates (i.e., it's also T^w_g). For example, consider the point (1, 2, 3) in gripper coordinates. We compute that in world coordinates by premultiplying by our new matrix as follows:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 8 \\ 1 \\ 1 \end{bmatrix}$$

Which you should be able to verify is correct by looking at Figure 23.

6.8 Using Fixed Axes

Recall that the fixed axes approach does *everything relative to the original world coordinate frame*. The sequence of transformations in this case was as follows: Rot x_w(-90) then Rot y_w (-90), then Trans(0,5,0) relative to world coordinates.

When we using "fixed axes" computation, (i.e. each new rotation is relative to the original gripper coordinate frame), we write the equations from right to left.

Trans (0,5,0) * Rot y (-90) * Rot x(-90)

The matrices for this product are as follows:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Check it out! this is the same equation we got when we did the computation using moving axes! (Phew!)

6.9 Fixed Axes WARNING

It is very important to note that rotations under the fixed axis approach can be very deceiving. For example, consider the axes of Figure 24 and suppose you want to rotate the w frame by -90 degrees about the yg axis. The rotation is depicted in Figure 25 is not what one might expect! To visualize what is happening, you need to imagine that the two frames are locked together as you perform the rotation.

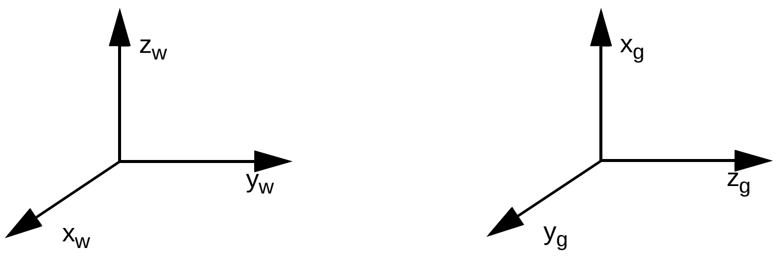


Figure 24. Two coordinate frames.

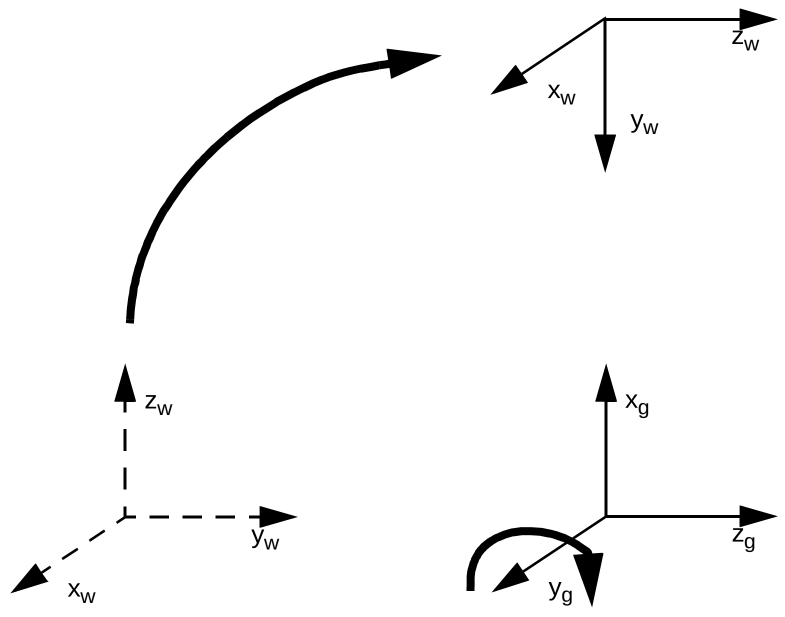


Figure 25. A rotation of the original w coordinate frame from Figure 24 (shown as dotted arrows) about the yg coordinate frame from Figure 24.

7. Forward Kinematics

One task that we often wish to do is the following: given the joint angles of a robot arm, compute the transformation between world and gripper coordinates. Of course, at this point given any fixed joint angles, we already have the tools to compute this transformation. However, what we really would like to do is to come up with a transformation matrix that is a function of the joint angles of the robot.

7.1 A First Example

Consider the robot arm given in Figure 26. This arm has two links (of length L1 & L2) and one joint which can rotate about its Z axis. There are 3 coordinate frames, world coordinates, joint coordinates, and gripper coordinates. Figure 27 contains a picture of the same arm, but this time, the joint has been rotated by 30 degrees (about its z axis - the only axis about which it can rotate).

Now look at the * in both figures. It should be clear that the world coordinates of the * do not change between Figure 26 and Figure 27, but the location of * in link coordinates does change, as does its location in gripper coordinates.

There is one point in this image whose location does not move in any frame as the joint moves. The point located at the very center of the joint (i.e. with Joint coordinates (0,0,0)) is always at world coordinates location (L1, 0, 0), and always at (-L2, 0, 0) in gripper coordinates, no matter how you move the joint.

Suppose that we want to be able to convert between gripper coordinates and world coordinates, as a function of the angle of the joint.

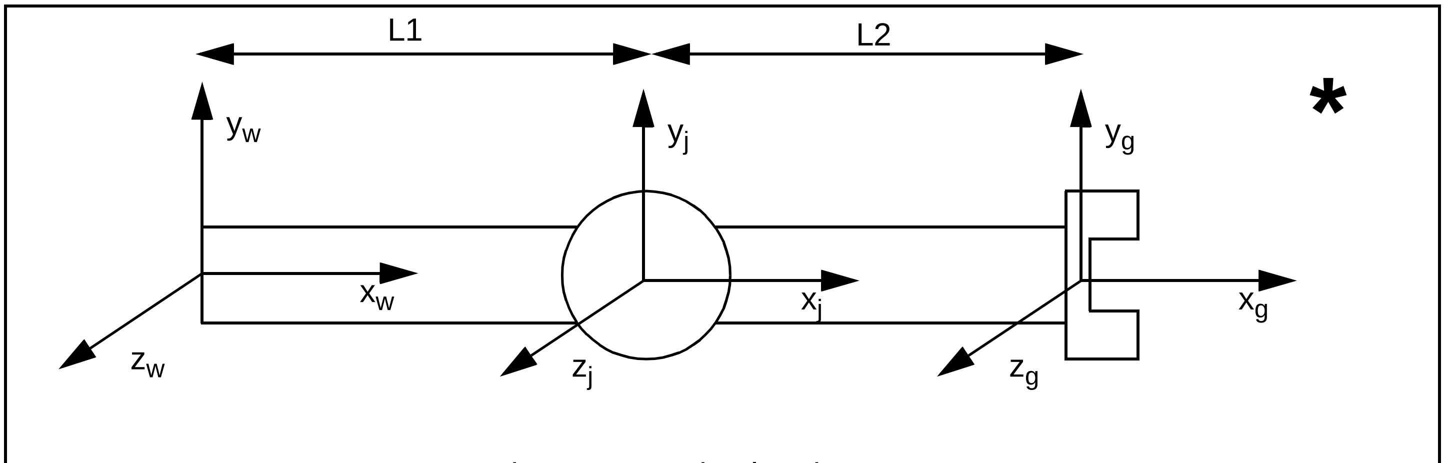


Figure 26. A simple robot

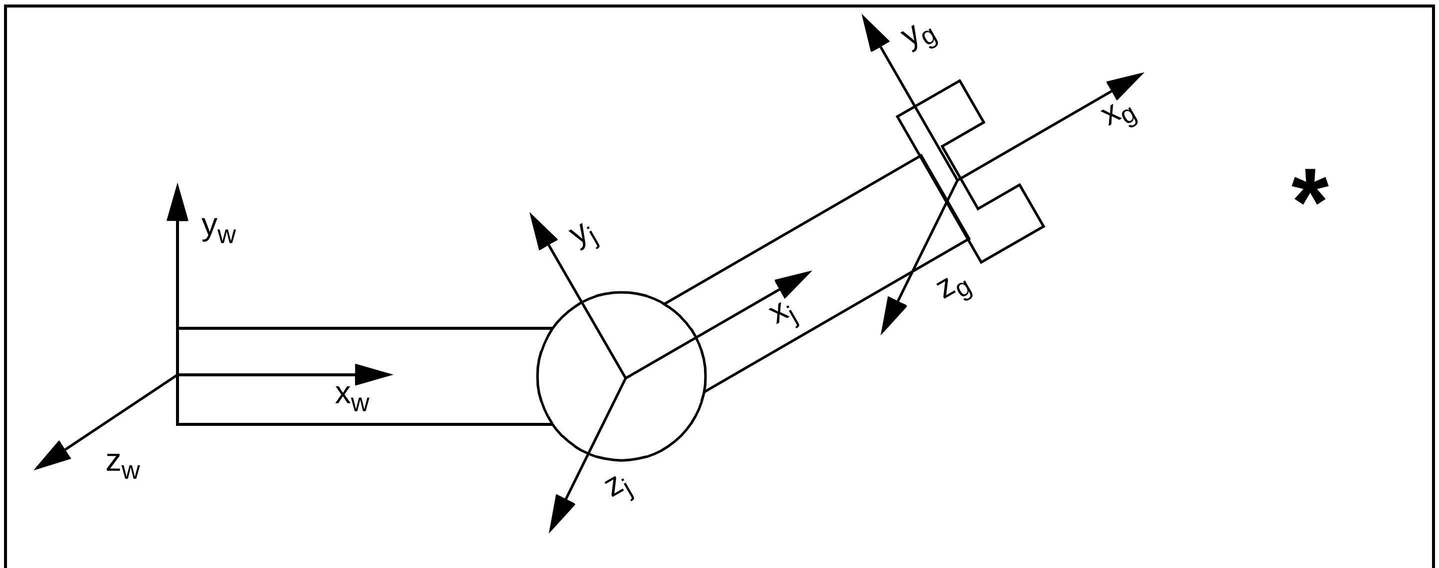


Figure 27. The arm from Figure 26 with the joint rotated by 30 degrees.

Let’s begin by just looking at Figure 26 again and considering the case where the joint is not rotated at all. In this case, to convert a point from gripper coordinates to world coordinates all we do is add L1+L2 to whatever the x value is. e.g., suppose the * is located at (4, 3, 0) in gripper coordinates. Then it’s obviously located at (4+L1+L2, 3, 0) in world coordinates.

But wait! To convert from gripper to world coordinates isn’t as simple as that. Because if the joint is rotated as shown in Figure 27, then it’s no longer a simple addition of L1+L2.

What we want is a way to easily convert between gripper and world coordinates *as a function of joint angle*. Let’s call the angle that the joint is rotated ψ. We want one matrix that has the variable ψ built into it, and if we plug in the value for ψ, that matrix will be our matrix that we multiply a point in gripper coordinates by to get the point in world coordinates.

Now let’s do the math. To move our frame from world coordinates to gripper coordinates, we need to translate a distance of L1 along the x axis, and then rotate by whatever angle our joint is twisted to (e.g. 0 degrees in Figure

26 or 30 degrees in Figure 27). We're going to do it as relative motion, so we end up multiplying the matrices Trans(L1, 0, 0) by Rot z (ψ) from left to right. So we have the following :

$$\begin{bmatrix} 1 & 0 & 0 & L1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\psi & -\sin\Psi & 0 & 0 \\ \sin\Psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\Psi & 0 & L1 \\ \sin\Psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 28. Moving a frame from world coordinates to joint coordinates.

But of course, this only moves us from world coordinates to joint coordinates. We wanted to move our frame from world coordinates to gripper coordinates. Once we have a frame in joint coordinates, moving it to gripper coordinates is simply a translation of [L2, 0, 0]. So we need to multiply the two matrices above by Trans[L2, 0, 0].

$$\begin{bmatrix} 1 & 0 & 0 & L1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\psi & -\sin\Psi & 0 & 0 \\ \sin\Psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & L2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\Psi & 0 & L2\cos\psi + L1 \\ \sin\Psi & \cos\psi & 0 & L2\sin\Psi \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 29. Moving a frame from world coordinates to gripper coordinates.

Now, our final equation looks pretty messy, but it's not too bad. Suppose that ψ is 0 (i.e. we've got Figure 26). Then we have:

$$\begin{bmatrix} \cos\psi & -\sin\Psi & 0 & L2\cos\psi + L1 \\ \sin\Psi & \cos\psi & 0 & L2\sin\Psi \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & L2{+}L1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 30. Moving a frame from world coordinates to gripper coordinates when the joint angle is zero.

Remember that we computed how to move a frame from world coordinates to gripper coordinates. Which turns points in gripper coordinates into points in world coordinates.

Wow! Figure 30 is actually exactly what we predicted it would be. Look at it. It's the matrix Trans(L1+L2, 0, 0).

7.2 A Second Example

Just for fun, let’s compute the transformation when the joint is rotated by 90 degrees, so the gripper is pointing straight up in the air. Well, we plug in 90 degrees for ψ and we get:

$$\begin{bmatrix} \cos\psi & -\sin\Psi & 0 & L2\cos\psi + L1 \\ \sin\Psi & \cos\psi & 0 & L2\sin\Psi \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & L1 \\ 1 & 0 & 0 & L2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 31. Moving a frame from world coordinates to gripper coordinates when the joint angle is 90 degrees.

Does this make sense? Think about it. Suppose that you have a point that is located right at the origin of the gripper. So in gripper coordinates, it’s location is (0,0,0). Where is that in world coordinates? Let’s multiply:

$$\begin{bmatrix} 0 & -1 & 0 & L1 \\ 1 & 0 & 0 & L2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} L1 \\ L2 \\ 0 \\ 1 \end{bmatrix}$$

Figure 32. Transforming a point from gripper coordinates to world coordinates when the joint angle is 90 degrees.

Hey, it works!

7.3 Some Tips on Using Mathematica

Clearly it’s a major pain to do anything with more than one or two manipulations by hand. Mathematica can really help. Here are a couple of hints on how I computed the information for this document. I’m assuming some familiarity with the very basics of mathematica.

For starters, I wrote mathematica functions to represent each of the rotation and translation matrices. Just in case you haven’t seen a mathematica function before, here’s how you write a function that takes to variables, a and b, and returns the mean (average) of a and b (note: the underscore defines what’s a variable. Don’t use underscores for anything else or you’ll have problems with your code:

```
avg[a_, b_] := ((a+b)/2)
```

Figure 33. A simple mathematica function to compute the mean of two variables

Matrices are represented as lists of lists. So, here is my function for Rotx:

```
        Rotx[theta_] := (
                              {{1, 0, 0, 0},
                               {0, Cos[theta], -1*Sin[theta], 0},
                               {0, Sin[theta], Cos[theta], 0},
                               {0, 0, 0, 1}}
                        )
```

Remember that Mathematica uses radians. So to Rotx by 90 degrees I say

```
                                    Rotx[Pi/2]
```

Now, let's use this function. If I want, I could run my function to see what the matrix is to Rotate x by 0 (this should be the identity matrix, right?!) (Note: From this point onwards I'll show you the "in" and "out" messages from Mathematica so you can distinguish my input from its output)

```
In[2]:= Rotx[0]

Out[2]= {{1,0,0,0},{0,1,0,0},{0,0,1,0},{0,0,0,1}}
```

Hmmm, correct, but not really very pleasing to the eye. We can use the built-in MatrixForm function to display matrices with a little more beauty:

```
In[3]:= MatrixForm[Rotx[0]]

Out[3]//MatrixForm=
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

So what do we get when we rotate about x by Pi/2?:

```
In[4] := MatrixForm[Rotx[Pi/2]]

Out[3]//MatrixForm=
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Looking good!

8. References

Much of this tutorial is derived from “Essential Kinematics for Autonomous Vehicles” by Alonzo Kelly, Carnegie Mellon University Robotics Institute technical report number CMU-RI-TR-94-14, May 1994, available on the web at `http://www.frc.ri.cmu.edu/~alonzo/pubs/reports/pdf_files/kinematics.pdf`