# A Gentle Introduction to Coordinate Frame Conversions

C. Padwick

January 15, 2025

## Contents

**1 Introduction** **2**

**2 Coordinate Frames and Vectors** **2**

**3 2D Rotation Matrices** **3**

3.1 Representing The Rotation As A Matrix . . . . . 5

3.2 Primer on Matrix/Vector Multiplication . . . . . 5

3.3 Applying The Rotation Matrix . . . . . 5

**4 Extending to 3D Rotation Matrices** **6**

4.1 Rotations About Each Axis . . . . . 6

4.2 Order of Rotations Matters A Lot . . . . . 7

4.3 Roll Pitch Yaw . . . . . 7

4.4 Combining the Rotation Matrices . . . . . 8

**5 Coordinate Transformations** **8**

**6 Conclusion** **10**

**Abstract**

Imagine you have a robot with a camera mounted on it. The camera measures something of interest, like the distance to the nearest chair or the distance to the nearest wall, and you want the robot to navigate to that object. Since it is usually impossible to locate the camera exactly at the origin of the robot, it is necessary to convert the distance from the camera to the object into a distance from the robot. This is called a coordinate frame conversion. Generally speaking this coordinate frame conversion consists of rotation and translation from the camera frame to the robot frame. This document is a short introduction to Euclidean coordinate frame conversions. This document provides a gentle introduction to coordinate transformations (rotations and translations) with the intent of giving the reader a solid understanding of the fundamentals without needing an advanced mathematics or engineering course in linear algebra.

## 1 Introduction

Image this: you are building a robot for a First Robotics competition and your vision mentor says "we need to correct the apriltag vector detected by the vision system to align with the robot's coordinate frame." If you don't know what this means, then you are reading the right document!

You don't need a familiarity with robotics to read this document. The same concepts will apply to any device which takes measurements in a given coordinate frame and those measurements need to be converted to a different coordinate frame. Rotation matrices are essential in many areas of mathematics, engineering, and computer science. In this doc I'm going to go over some basics and give a gentle introduction so you can get the working knowledge you need and move on to more advanced topics later.

## 2 Coordinate Frames and Vectors

A coordinate frame is a set of vectors, or axes, which are orthogonal to each other. Orthogonal means the angle between any two axes is 90 degrees. A simple 2D coordinate frame is shown in figure 1.

The coordinate frame is useful because we can locate points in the coordinate frame by their X and Y coordinates. For example, the point V in figure 1 has an x and a y coordinate that can be determined by its position on the coordinate frame. In the figure these components on the X and Y axes are shown by the dashed lines.

In terms of expressing the position of a point in the coordinate frame, we can write a vector as follows:

$$\vec{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}. \tag{1}$$

where $v_x$ is the x-component of the vector and $v_y$ is the y-component of the vector. You can compute the length of the vector easily too by taking the square root of the sum of the squares of the components:

$$|\vec{v}| = \sqrt{v_x^2 + v_y^2}. \tag{2}$$

The extension to 3-dimensions or even N-dimensions is straight forward - you just add one more component for each dimension.

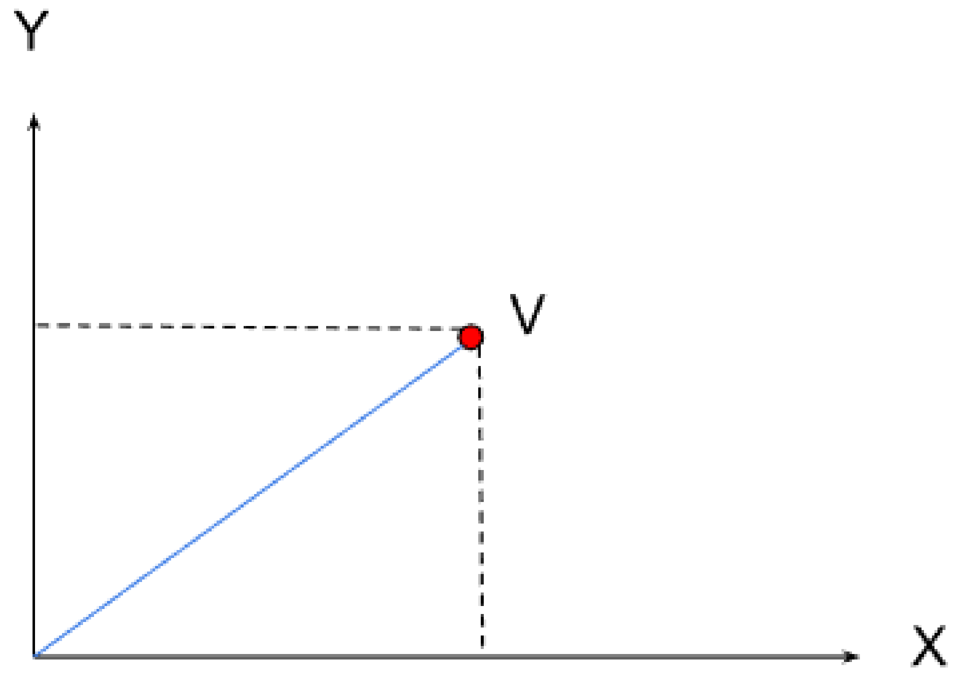

Figure 1: An 2D coordinate frame defined by X and Y.

## 3 2D Rotation Matrices

We're going to work out some simple rotations in 2D so we can get some experience with rotation matrices and how they work. This can all be extended to 3D but it's easier to visualize in 2D and nicer to start with. Let's draw an X-Y axis in 2D and a rotated version of the axis, A-B, shown in figure 2. The rotation angle is $\theta$. The rotation angle is positive, and is measured counterclockwise. For simplicity let's set the length of A and B both to 1, .e.g

$$|A| = |B| = 1 \tag{3}$$

.

Let's ask the following question: How do we get the coordinates of the points A and B in the X-Y frame? We are given the angle of rotation $\theta$. We need to compute the X-Y coordinates of each point A and B in the X-Y frame.

Let's start with the point A. From trigonometry we can figure out the X-Y coordinates of A by noticing $cos(\theta) = x/|A|$ or $x = cos(\theta) * |A|$. Since we said that $|A| = 1$, we can just write $x = cos(\theta)$. Similarly, we can figure out the y coordinate of A by noticing that $sin(\theta) = y/|A|$ or $y = sin(\theta) * |A|$, and we can just write $y = sin(\theta)$. This gives us the X-Y coordinates of A as:

$$\vec{A} = \begin{bmatrix} cos(\theta) \\ sin(\theta) \end{bmatrix}. \tag{4}$$

We can do the same for the point B. We can see that the X coordinate of B is going to be negative since it interects with the negative X axis. We can do the same thing we did for the A

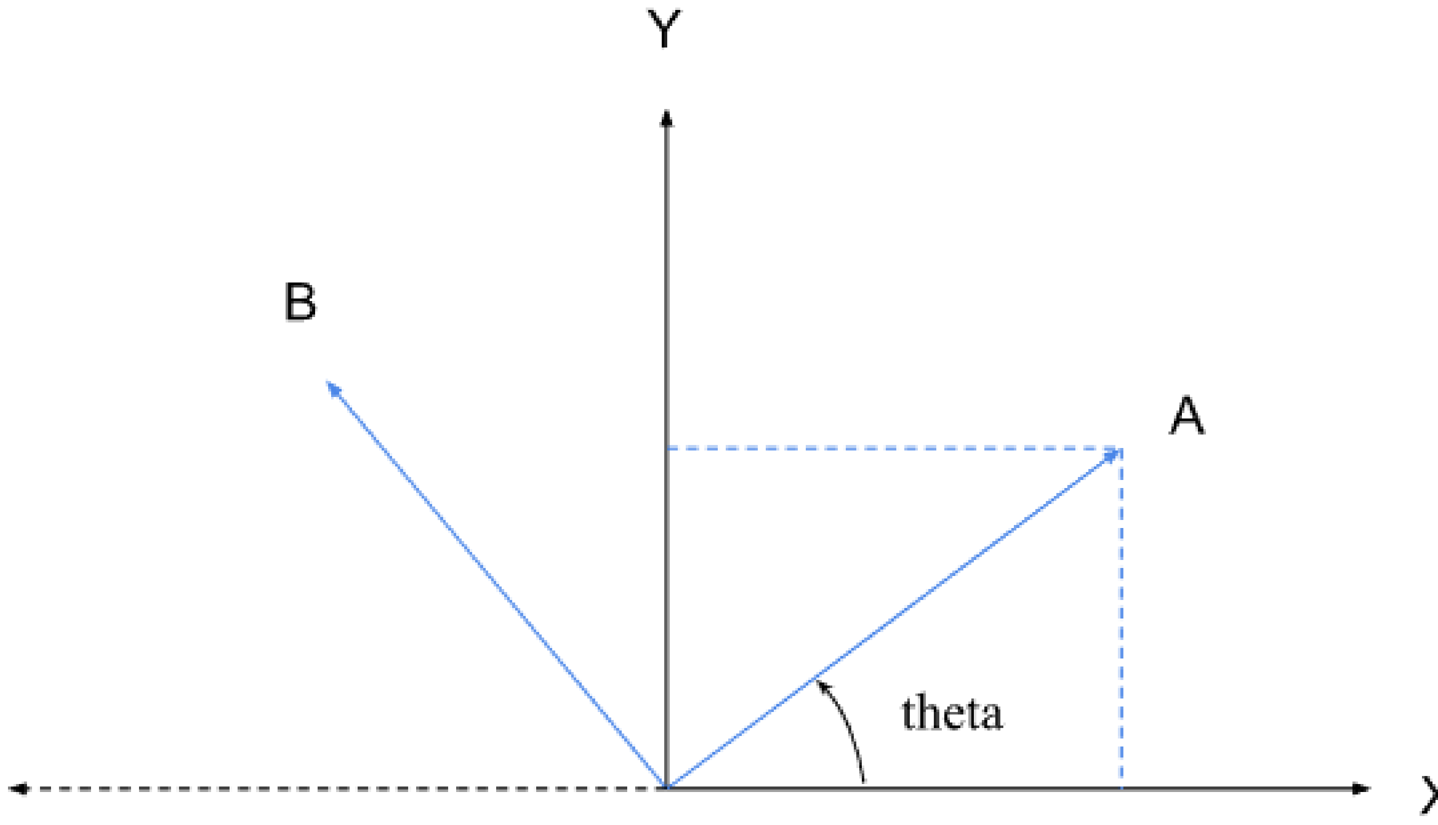

Figure 2: An 2D coordinate frame defined by X and Y, and a rotated version of the axis denoted by A-B.

coordinate, except this time we see that $sin(\theta) = -x/|B|$ or $x = -sin(\theta) * |B| => x = -sin(\theta)$. Take a moment and study the diagram and convince yourself that this is true. The y-coordinate of B is going to be $cos(\theta) = y/|B|$ or $y = cos(\theta) * |B| => y = cos(\theta)$. Pause for a few moments and study the diagram and convince yourself that this is true.

Writing out the coordinates B, we get:

$$\vec{B} = \begin{bmatrix} -sin(\theta) \\ cos(\theta) \end{bmatrix}. \tag{5}$$

If you wanted to express these points as vectors in the X-Y frame, you could write them as:

$$\vec{A} = sin(\theta)\hat{x} + cos(\theta)\hat{y} \tag{6}$$

$$\vec{B} = -sin(\theta)\hat{x} + cos(\theta)\hat{y} \tag{7}$$

The $\hat{x}$ and $\hat{y}$ are unit vectors in the X-Y frame. They are aligned with the X and Y axes respectively but have unit length.

## 3.1 Representing The Rotation As A Matrix

Now that we have the coordinates of A and B, we can represent the rotation as a matrix. This is actually quite simple. We can write the rotation matrix as:

$$R(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}. \tag{8}$$

If you look at the entries of the matrix, you can recognize that we have taken the equation for $\vec{A}$ and put the X-Y coordinates into the first column. The X coordinate goes in the first row and the Y coordinate goes in the second row. For the second column, we have taken the equation for $\vec{B}$ and put the X-Y coordinates into the second column. The X coordinate goes in the first row and the Y coordinate goes in the second row.

You might ask, why would we do this? Why not just use the equations for $\vec{A}$ and $\vec{B}$ above? It is customary to write something like "it turns out that we can use the rotation matrix more easily and more efficiently". It turns out that is true!

## 3.2 Primer on Matrix/Vector Multiplication

Ok, now let's multiply a vector by a matrix. In case you haven't done this before, here is a quick primer on the rules of matrix multiplication:

- When you multiply a vector by a matrix, the vector has to have the same number of rows as the matrix has columns. In this case we'll express the vector as a column vector with 2 rows and 1 column. The matrix has 2 rows and 2 columns so we can multiply it by a 2x1 vector.
- When we multiply a vector by a matrix, as long as the vector has the same number of rows as the matrix has columns, we get a vector of the same size as the input vector. Since our vector has 2 elements, the result will have two elements as well.
- When we're multiplying a matrix and a vector, we start at the first row of the matrix and multiply the column elements of the matrix by the rows of the vector and add them. We put that result in the first element of the result vector. We then move on to the second row of the matrix and multiply the column elements of the second row of the matrix by the rows of the vector and add them. We put that result in the second element of the result vector. We continue this process until we have computed all the elements of the result vector. In this case, the result vector will have 2 elements since we're only working with 2D rotation matrices.

Here is an example:

$$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} * \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} a_1v_1 + a_2v_2 \\ a_3v_1 + a_4v_2 \end{bmatrix} \tag{9}$$

## 3.3 Applying The Rotation Matrix

Alright, now that that is out of the way, let's do a rotation. Let's say we have a vector in the A-B frame of reference, and this vector is just the A axis. So the vector coordinates are just (1, 0) in the A-B frame of reference. So let's rotate this vector in the X-Y frame of reference:

$$A_{xy-frame} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} cos(\theta) + 0 \\ sin(\theta) + 0 \end{bmatrix} = \begin{bmatrix} cos(\theta) \\ sin(\theta) \end{bmatrix} \tag{10}$$

This is the same answer we got before, so this is good news! Let's do it again, but for the B vector in the XY-frame:

$$B_{xy-frame} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} * \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 - sin(\theta) \\ 0 + cos(\theta) \end{bmatrix} = \begin{bmatrix} -sin(\theta) \\ cos(\theta) \end{bmatrix} \tag{11}$$

This is the same result we got before, so this is good news! It means our matrix multiply approach actually works!

Ok, but why do we care? What does this buy us? This technique works for any vector in the A-B frame of reference. So if someone gives us a vector in the A-B frame of reference, we can multiply it by our 2D rotation matrix to get the vector in the X-Y frame of reference. This is the first part of learning how to convert from one frame of reference to another!

## 4 Extending to 3D Rotation Matrices

Now that we have 2D rotation matrices, let's extend them to 3D. We said before that all we need to do is add another component for the extra dimension. Our 2D vectors now have 3 components, and our 2D rotation matrices now have 3 rows and 3 columns, like this:

$$\vec{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{12}$$

$$\begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix} \tag{13}$$

### 4.1 Rotations About Each Axis

It is possible to create a rotation matrix that rotates about the $x$-axis, the $y$-axis, or the $z$-axis. These matrices are simple to construct, and look very similar to their 2D counterparts.

For instance, the rotation matrix around the $z$-axis is

$$R_z(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{14}$$

You can recognize this as just the 2D rotation matrix about X-Y with some extra elements. If you think about why this is, it actually makes sense. In the 2D example we were actually rotating about the z-axis, which faced out of the page. When we rotate about the z-axis the vectors only change in the X-Y plane. So it makes sense that the matrix would look this way in that the only components that change are the X-Y components. Another way to think about this is that the third column of the matrix is (0, 0, 1), which also happens to the be coordinates of the z-axis. Only the X-Y components change when whatever rotate about the z-axis.

The rotation matrix around the $y$-axis is

$$R_y(\gamma) = \begin{pmatrix} cos(\gamma) & 0 & sin(\gamma) \\ 0 & 1 & 0 \\ -sin(\gamma) & 0 & cos(\gamma) \end{pmatrix} \tag{15}$$

Here I have introduced another angle $\gamma$. Notice that the second column of the matrix is (0, 1, 0), which also happens to the be coordinates of the y-axis. When you rotate about the y-axis the vectors only change in the X-Z plane.

The rotation matrix around the $x$-axis is

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{pmatrix} \tag{16}$$

Here I have introduced another angle $\alpha$. Notice that the first column of the matrix is (1, 0, 0), which also happens to the be coordinates of the x-axis. When you rotate about the x-axis the vectors only change in the Y-Z plane.

These rotations are shown in figure 3.

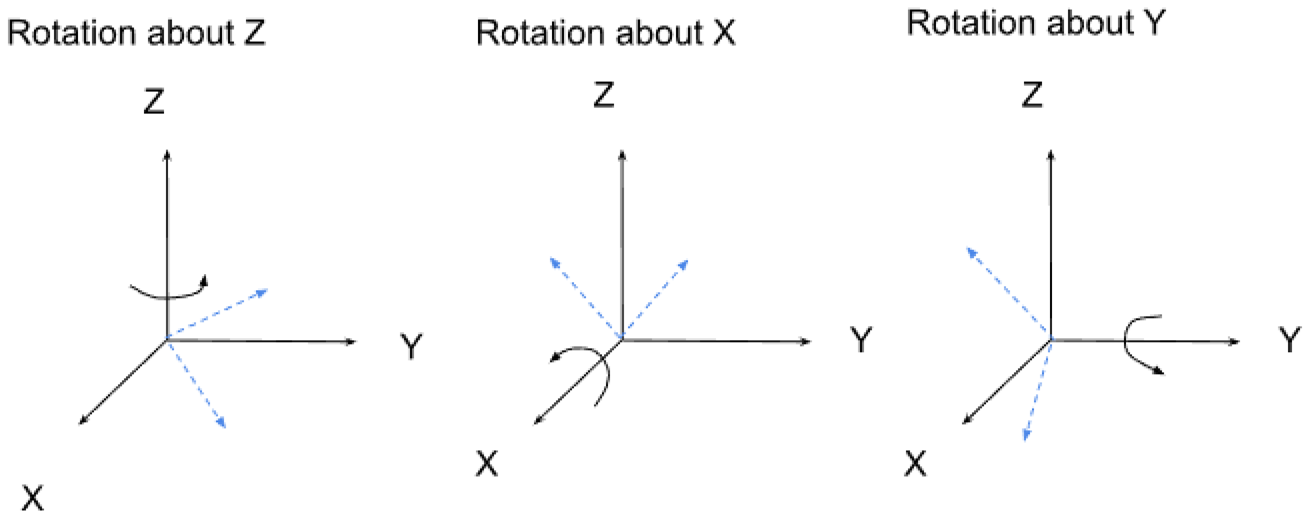

Figure 3: 3D Rotations about each axis.

## 4.2 Order of Rotations Matters A Lot

Rotation matrices are not commutative. That is, $R_z(\theta) * R_y(\gamma) \neq R_y(\gamma) * R_z(\theta)$. This is because the order of rotations matters! You can see this pictorially by making a 3D coordinate frame out of your thumb and fingers on your right hand. If you rotate about the z-axis first, then rotate about the y-axis, your hand will end up in a different orientation than if you rotated about the y-axis first, then rotate about the z-axis.

You should be able to convince yourself mathematically that $R_z(\theta) * R_y(\gamma) \neq R_y(\gamma) * R_z(\theta)$ by multiplying the matrices together both ways and comparing the results.

## 4.3 Roll Pitch Yaw

Another way to think about rotations is in terms of roll, pitch, and yaw. Roll is rotation around the x-axis, pitch is rotation around the y-axis, and yaw is rotation around the z-axis. This is often used in aeronautical applications to describe the orientation of an airplane.

### 4.4 Combining the Rotation Matrices

In you know the individual rotation angles about each of the 3 axes, you can construct each rotation matrix. You can then combine those matrices to get a single rotation matrix by multiplying them together as follows:

$$R(\theta, \gamma, \alpha) = R_x(\alpha) * R_y(\gamma) * R_z(\theta) \tag{17}$$

But you do have to be careful about the order of multiplication. If we include a vector in the equation as follows:

$$v' = R(\theta, \gamma, \alpha) * \vec{v} = R_x(\alpha) * R_y(\gamma) * R_z(\theta) * \vec{v} \tag{18}$$

hopefully you can see that the first rotation is around the z-axis, followed by a rotation about the y-axis, followed by a rotation about the x-axis. The equation reads right to left, and the first thing that gets multiplied by the vector is the z-axis rotation.

## 5 Coordinate Transformations

Now we have enough background to tackle a coordinate transformation on a robot. Let's say that have a robot with a camera mounted on it. The coordinate frame of the robot is denoted by $X_R$, $Y_R$, and $Z_R$. The coordinate frame of the camera is denoted by $X_C$, $Y_C$, and $Z_C$. The robot coorinate frame is positioned at the origin of the robot such that $Z_R$ points upwards and $X_R$ points forwards on the robot.

The camera coordinate frame follows the coordinate frame of OpenCV, such that $Z_C$ points outwards from the center of the camera, $X_C$ points to the right, and $Y_C$ points down. The camera coordinate frame is shwon in figure 4.

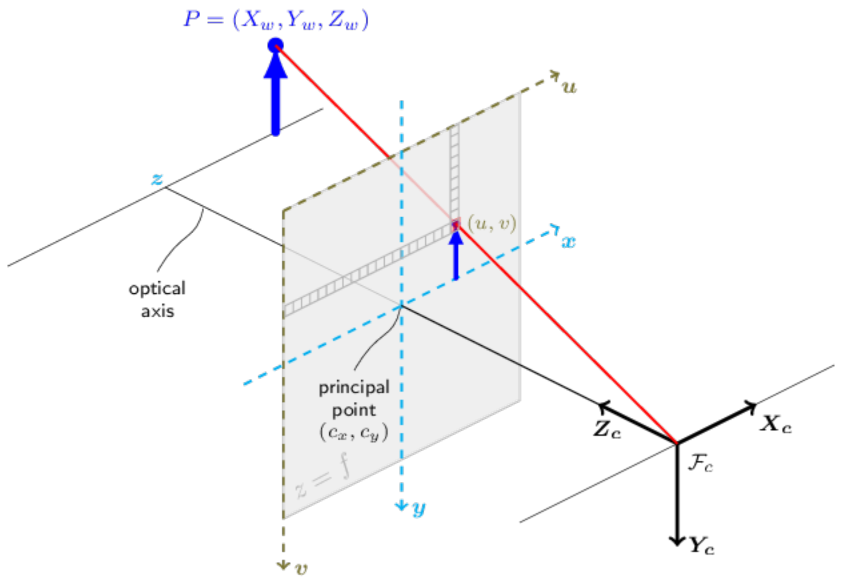

Figure 4: The camera coordinate frame.

The camera is placed on a location on the robot such that the camera $Z_C$ points towards the front of the robot. The camera is looking at something of interest, such as the apriltag, which is placed somewhere in the camera's field of view. This setup is shown in figure 5.

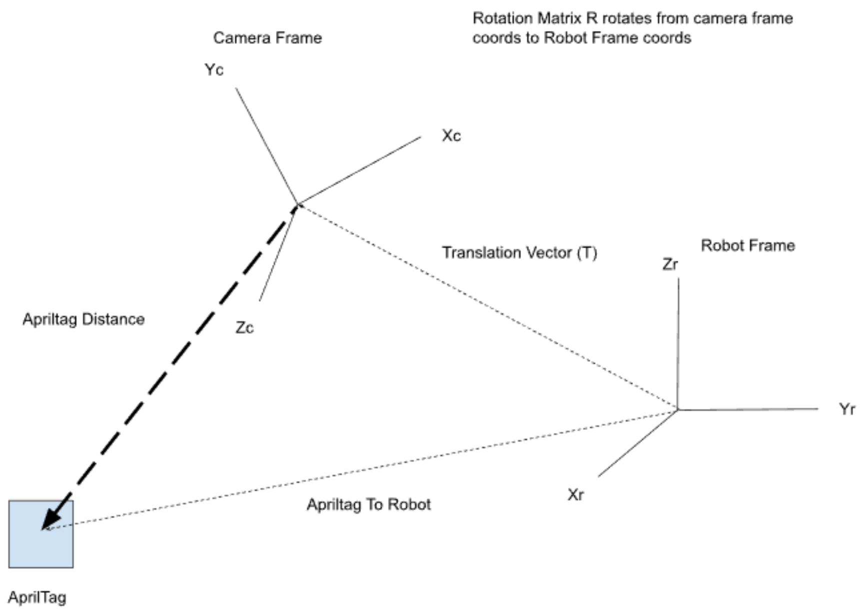

Figure 5: The robot and camera coordinate frames.

In this case the camera measures the location of the apriltag in the camera coordinate frame. That is denoted by the vector called "Apriltag Distance" on the diagram. In order to get the coordinates of the apriltag in the robot coordinate frame, we need to transform the vector called "Aptriltag Distance" by the rotation matrix of the robot, and apply the translation vector denoted by "T" in the diagram. The resulting vector is the location of the apriltag in the robot coordinate frame, denoted by "Apriltag to Robot".

The translation vector is just the mounting point of the camera in the robot coordinate frame. We generally try to make it so that the camera is rigidly mounted on the robot, and doesn't move around. If the camera is not rigidly mounted on the robot, then the translation vector will be off and our results will be off. The translation vector is simply a 3-component vector as follows:

$$\vec{T} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{19}$$

Generally this can be derived fairly accurately from CAD Drawings. The rotation matrix may also be derived from CAD drawings, or it can be measured directly. We will work out how to measure the rotation matrix in the next section. In this section let's work out how to build a rotation matrix from a CAD drawing.

The first thing to notice is that the camera coordinate frame is rotated from the robot frame. The $Z_C$ axis aligns with the robot $X_R$ axis. So we need a rotation about the Y axis by 90 degrees to get those aligned. After that we can rotate by -90 degrees about the X axis to get the camera coordinate frame aligned with the robot coordinate frame. That is the rotation matrix we need to build. This is shown in figure 6.

Putting this together we would get the following rotation matrix:

$$R_x(-90) * R_y(90) * \vec{v} \tag{20}$$

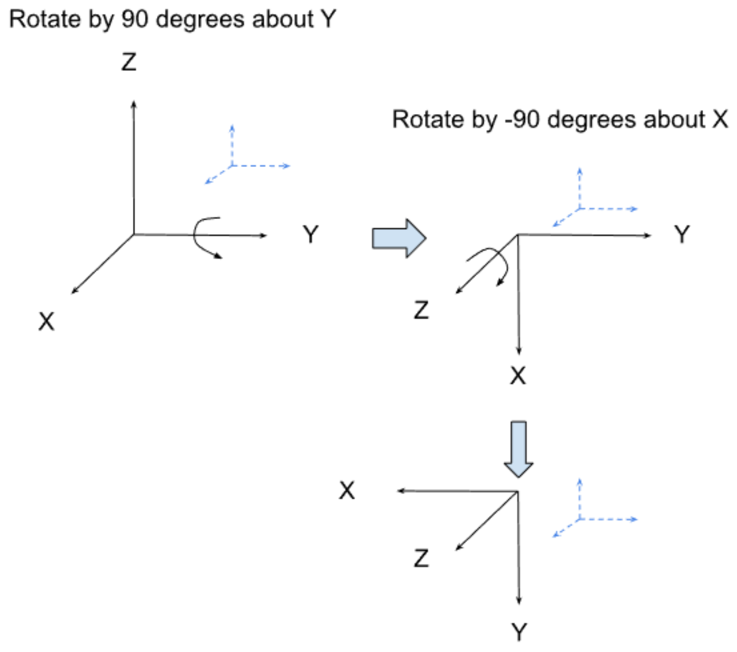

Figure 6: Two rotations to align the camera coordinate frame with the robot coordinate frame.

which expands to:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{pmatrix} * \begin{pmatrix} cos(\gamma) & 0 & sin(\gamma) \\ 0 & 1 & 0 \\ -sin(\gamma) & 0 & cos(\gamma) \end{pmatrix} * \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \tag{21}$$

where $\gamma$ is the angle of rotation about the Y axis equal to 90 degrees ($\pi/2$ radians), and $\alpha$ is the angle of rotation about the X axis equal to -90 degrees ($-\pi/2$ radians).

Putting this all together and including the translation vector we get:

$$v_{robot\vec{-}frame} = R * v_{camer\vec{a}-frame} + \vec{T} \tag{22}$$

or

$$v_{robot\vec{-}frame} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{pmatrix} * \begin{pmatrix} cos(\gamma) & 0 & sin(\gamma) \\ 0 & 1 & 0 \\ -sin(\gamma) & 0 & cos(\gamma) \end{pmatrix} * \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \tag{23}$$

## 6 Conclusion

Summarize key points. You may also mention limitations and open problems or suggest further reading.

## References

[1] Author Name, *Title of Book or Paper*, Journal/Publisher, Year.

[2] Another Author, “Interesting Insights Into 3D Rotations,” Conference Proceedings, Year.