Lecture 2-1: 3D Geometry Basics

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor(s).*

This lecture introduces basic geometric concepts, including translations, rotations, and poses. In particular, we will cover:

- coordinate frames;
- positions and translations;
- attitude representation;
- pose representation.

2.1 Coordinate Frames

A *coordinate frame* is a set of orthogonal *axes* attached to a body that serves to describe position of points relative to that body. The axes meet at a single point, which is called the *origin* of the coordinate frame. In this course we mainly attach coordinate frames to:

- our robot (robot frame "r");
- each sensor on our robot (e.g, camera frame "c");
- a fixed location in the world (world frame "w");
- external bodies (e.g., other robots, objects in the world).

In other robotics applications (e.g., manipulation), the robot has multiple articulated parts, and one attaches a reference frame to each part.

In robotics, we use *right-handed* coordinate frames, where the *direction* of the axes is chosen according to the *right-hand* rule. Consider a 3D coordinate frame r with axis $\boldsymbol{x}_\mathrm{r}$, $\boldsymbol{y}_\mathrm{r}$, $\boldsymbol{z}_\mathrm{r}$; then different mnemonics that describe a *right-handed* coordinate frame are as follows:

- using your right hand, your thumb points along the $\boldsymbol{z}_\mathrm{r}$ axis in the positive direction and the curl of your fingers represents a motion from the $\boldsymbol{x}_\mathrm{r}$ axis to the $\boldsymbol{y}_\mathrm{r}$ axis.
- using your right hand, the positive $\boldsymbol{x}_\mathrm{r}$ axis points along your index finger, the positive $\boldsymbol{y}_\mathrm{r}$ axis points along your middle finger, and the positive $\boldsymbol{z}_\mathrm{r}$ points long the thumb.
- using your left hand (!), the positive $\boldsymbol{x}_\mathrm{r}$ axis points along your middle finger, the positive $\boldsymbol{y}_\mathrm{r}$ axis points along your index finger, and the positive $\boldsymbol{z}_\mathrm{r}$ points long the thumb.

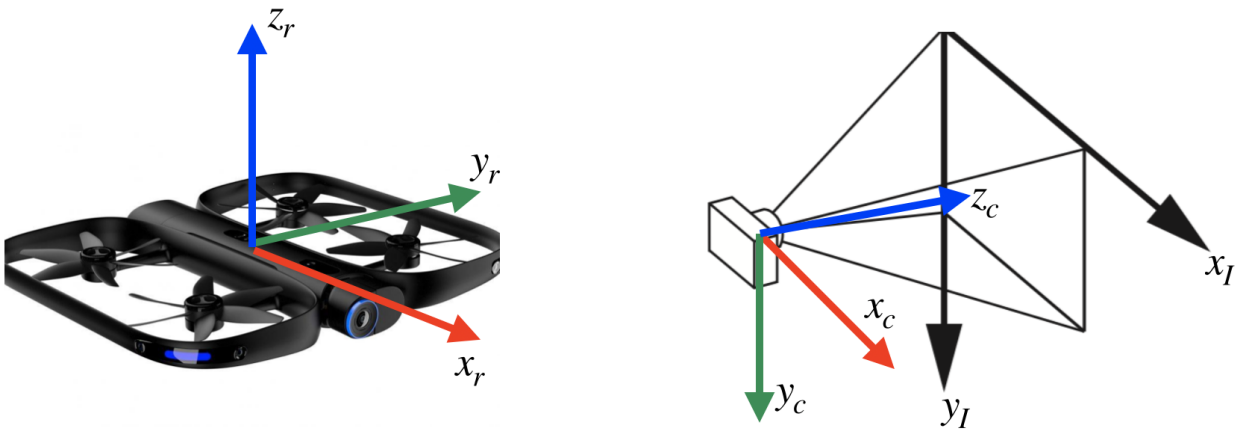

Figure 2.1: Coordinate frames. (left) Robot frame, (right) Camera frame and Image frame

Example 2.1.1 (Robot frame convention)**.** We use the following standard convention:

- Origin: the center of mass.
- Axes: $\boldsymbol{x}_{\mathrm{r}}$ forward, $\boldsymbol{y}_{\mathrm{r}}$ to the left, and $\boldsymbol{z}_{\mathrm{r}}$ up.

Example 2.1.2 (Camera and Image frame conventions)**.** We use the following standard conventions:

Camera frame (3D)

- Origin: center of the camera.
- Axes: $\boldsymbol{x}_{\mathrm{c}}$ to the right, $\boldsymbol{y}_{\mathrm{c}}$ down, $\boldsymbol{z}_{\mathrm{c}}$ looking at the scene.

Image frame (2D)

- Origin: top-left corner of the camera image.
- Axes: looking at the camera image, $\boldsymbol{x}_{\mathrm{c}}$ to the right, $\boldsymbol{y}_{\mathrm{c}}$ down.

2.2 Points, positions, and translations

The advantage of defining a reference frame is that it allows representing points using linear algebra constructs. For instance, we can represent the position of a 3D point $\boldsymbol{p}$ with respect to the world frame "w" using a 3D vector:

$$\boldsymbol{p}^{\mathrm{w}} = \begin{bmatrix} p_x^{\mathrm{w}} \\ p_y^{\mathrm{w}} \\ p_z^{\mathrm{w}} \end{bmatrix} \tag{2.1}$$

where $p_x^{\mathrm{w}}, p_y^{\mathrm{w}}, p_z^{\mathrm{w}} \in \mathbb{R}$ are scalars, called the *coordinates* of $\boldsymbol{p}$ in the coordinate frame w. The coordinates $p_x^{\mathrm{w}}, p_y^{\mathrm{w}}, p_z^{\mathrm{w}}$ are equal to the projections of the point p to the axes $\boldsymbol{x}_{\mathrm{w}}$, $\boldsymbol{y}_{\mathrm{w}}$, $\boldsymbol{z}_{\mathrm{w}}$ of the reference frame w.

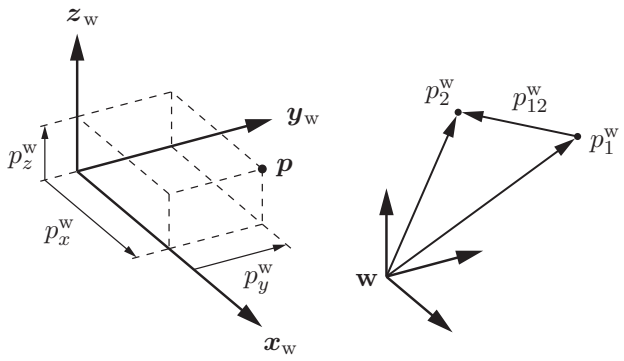

Figure 2.2: Point coordinates.

We can also use linear algebra to compute the displacement or *translation* between 2 points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$:

$$\boldsymbol{p}_{12}^{\mathrm{w}} = \boldsymbol{p}_2^{\mathrm{w}} - \boldsymbol{p}_1^{\mathrm{w}} \tag{2.2}$$

or, equivalently, compute the position of the second $\boldsymbol{p}_2$ given point $\boldsymbol{p}_1$ and the displacement $\boldsymbol{p}_{12}^{\mathrm{w}}$:

$$\boldsymbol{p}_2^{\mathrm{w}} = \boldsymbol{p}_1^{\mathrm{w}} + \boldsymbol{p}_{12}^{\mathrm{w}} \qquad \text{("composition")} \tag{2.3}$$

Clearly, it holds:

$$\boldsymbol{p}_{12}^{\mathrm{w}} = -\boldsymbol{p}_{21}^{\mathrm{w}} \qquad \text{("inverse")} \tag{2.4}$$

Note that we use a slightly different **notation for positions and translations**. In particular, the subscript of a translation $\boldsymbol{p}_{12}^{\mathrm{w}}$ stresses the fact that the translation is between point 1 and 2 and it is expressed with respect to the world frame w; on the other hand, we use a single subscript for positions, e.g., $\boldsymbol{p}_1^{\mathrm{w}}$, where it is implicit that the position is with respect to the origin of the coordinate frame w.

In other words, we can rephrase geometric concepts (positions, displacement) into algebraic ones (vectors). Note that in all these expressions we keep the **superscript** w, since the vectors are meaningless if we do not specify a frame for the coordinates in the vector.

In general, we represent positions and translations using vectors in $\mathbb{R}^d$, where $d = 2$ for planar problems and $d = 3$ in three-dimensional problems.

2.3 Attitude and rotations

The tools described in the previous section allow using vectors to describe positions of points in a given coordinate frame. In robotics, however, we are interested in modeling objects that can assume arbitrary positions and **orientations**.

We assume to deal with *rigid bodies*, whose position and orientation is fully described by the corresponding coordinate frame. Therefore, the question we address in this section is: *how can we represent the orientation of a frame, e.g.,* r*, with respect to another frame, e.g.,* w*?*

In this section we assume that the two coordinate frames have the *same origin but potentially different orientations* and we discuss alternative representations for the orientation of a frame. Then, in Section 2.4 we reconcile positions and orientations in a unified representation.

Terminology: the terms "orientation", "attitude", and "rotation" are used interchangeably to define the intuitive notion of orientation of a 3D body (although the term attitude is more rarely found in 2D problems).

2.3.1 Rotation matrix representation

A very naive way (that indeed will prove to be very clever later on) to *represent* the attitude of a frame r with respect to a frame w is as follows: treat the tip of each axis $\boldsymbol{x}_\mathrm{r}$, $\boldsymbol{y}_\mathrm{r}$, $\boldsymbol{z}_\mathrm{r}$ as a point, and stack the coordinates of each point with respect to the frame w as columns of a matrix. Let us consider a couple of examples to clarify this matrix representation.

Example 2.3.1 (2D rotation matrix)**.** In the 2D example in Fig. 2.3, it is easy to see that the coordinates of the tip of the axis $\boldsymbol{x}_\mathrm{r}$ with respect to the frame w are:

$$\boldsymbol{R}_\mathrm{r}^\mathrm{w} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin\theta & \cos(\theta) \end{bmatrix} \tag{2.5}$$

where θ is the angle shown in the figure.

Example 2.3.2 (3D rotation matrix)**.** As in the previous example, we can form a matrix by filling in each column with the coordinates of the axes of r expressed in the coordinate frame w:

$$\boldsymbol{R}_\mathrm{r}^\mathrm{w} = \begin{bmatrix} \boldsymbol{x}_\mathrm{r}^\mathrm{w} & \boldsymbol{y}_\mathrm{r}^\mathrm{w} & \boldsymbol{z}_\mathrm{r}^\mathrm{w} \end{bmatrix} \tag{2.6}$$

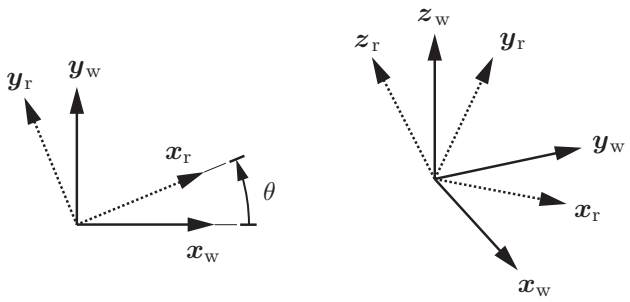

Figure 2.3: 2D and 3D Rotations.

The matrix $\boldsymbol{R}_\mathrm{r}^\mathrm{w}$ is called a *rotation matrix* (in aerospace, it is sometimes referred to as the *Direction Cosine Matrix*, DCM). Clearly, this is not a generic matrix, since its columns represent orthogonal unit-length axis that satisfy the right-hand rule. Therefore, any rotation matrix $\boldsymbol{R}_\mathrm{r}^\mathrm{w}$ has to satisfy:

- *orthogonality*: the axes $\boldsymbol{x}_\mathrm{r}^\mathrm{w}$, $\boldsymbol{y}_\mathrm{r}^\mathrm{w}$, $\boldsymbol{z}_\mathrm{r}^\mathrm{w}$ have unit length and are orthogonal to each other (independently on the reference frame they are expressed in), therefore:

$$\|\boldsymbol{x}_\mathrm{r}^\mathrm{w}\|_2^2 = 1 \qquad \|\boldsymbol{y}_\mathrm{r}^\mathrm{w}\|_2^2 = 1 \qquad \|\boldsymbol{z}_\mathrm{r}^\mathrm{w}\|_2^2 = 1 \qquad \text{(unit length)} \tag{2.7}$$

$$(\boldsymbol{x}_\mathrm{r}^\mathrm{w})^\mathsf{T}\boldsymbol{y}_\mathrm{r}^\mathrm{w} = 0 \qquad (\boldsymbol{x}_\mathrm{r}^\mathrm{w})^\mathsf{T}\boldsymbol{z}_\mathrm{r}^\mathrm{w} = 0 \qquad (\boldsymbol{y}_\mathrm{r}^\mathrm{w})^\mathsf{T}\boldsymbol{z}_\mathrm{r}^\mathrm{w} = 0 \qquad \text{(orthogonal vectors)} \tag{2.8}$$

  These relations can be rewritten directly as:

$$(\boldsymbol{R}_\mathrm{r}^\mathrm{w})^\mathsf{T}\boldsymbol{R}_\mathrm{r}^\mathrm{w} = \mathbf{I}_d \qquad (\textbf{orthogonality}) \tag{2.9}$$

  where $\mathbf{I}_d$ is the identity matrix of size d (as before, $d = 2$ in 2D problems and $d = 3$ in 3D). A matrix satisfying (2.9) is said to be *orthogonal* and it's easy to see that such a matrix satisfies:

$$(\boldsymbol{R}_\mathrm{r}^\mathrm{w})^{-1} = (\boldsymbol{R}_\mathrm{r}^\mathrm{w})^\mathsf{T} \tag{2.10}$$

- *right-handedness*: the 3D axes $\boldsymbol{x}_{\mathrm{r}}^{\mathrm{w}}$, $\boldsymbol{y}_{\mathrm{r}}^{\mathrm{w}}$, $\boldsymbol{z}_{\mathrm{r}}^{\mathrm{w}}$ have to satisfy the right-hand rule, which implies that $\boldsymbol{x}_{\mathrm{r}}^{\mathrm{w}} \times \boldsymbol{y}_{\mathrm{r}}^{\mathrm{w}} = \boldsymbol{z}_{\mathrm{r}}^{\mathrm{w}}$ where $\times$ denotes the cross product between vectors. Since these are unit-length vectors, the following relations hold:

$$\boldsymbol{x}_{\mathrm{r}}^{\mathrm{w}} \times \boldsymbol{y}_{\mathrm{r}}^{\mathrm{w}} = \boldsymbol{z}_{\mathrm{r}}^{\mathrm{w}} \iff (\boldsymbol{z}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}} (\boldsymbol{x}_{\mathrm{r}}^{\mathrm{w}} \times \boldsymbol{y}_{\mathrm{r}}^{\mathrm{w}}) = +1 \iff \det(\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}) = +1 \tag{2.11}$$

  where in the last equality we noticed that the determinant of a 3×3 matrix can be computed as a *triple product* [https://en.wikipedia.org/wiki/Triple_product#Scalar_triple_product] of the columns. In, particular given three vectors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^3$:

$$\boldsymbol{a} \cdot (\boldsymbol{b} \times \boldsymbol{c}) = \det\left(\begin{bmatrix} \boldsymbol{a} & \boldsymbol{b} & \boldsymbol{c} \end{bmatrix}\right) \qquad \text{and} \tag{2.12}$$

$$\boldsymbol{a} \cdot (\boldsymbol{b} \times \boldsymbol{c}) = -\boldsymbol{c} \cdot (\boldsymbol{b} \times \boldsymbol{a}) = \boldsymbol{c} \cdot (\boldsymbol{a} \times \boldsymbol{b}) \tag{2.13}$$

  Hence a rotation matrix $\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}$ has to satisfy:

$$\det(\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}) = +1 \qquad (\textbf{determinant } +1) \tag{2.14}$$

2.3.1.1 Operations involving rotations

Now that we have (at least one) way to represent the orientation of a frame, we would like to ask few natural questions that will be crucial to relate quantities measured in different coordinate frames:

- *rotating points*: given the coordinates of a point in a frame r and the attitude of r with respect to a frame w, how can we compute the coordinates of the point in the frame w?
- *composition*: given the attitude of a frame r with respect to a frame w and the attitude of a frame c with respect to a frame r, how can we compute the attitude of the frame c with respect to the frame w? Intuitively, this is "similar" to (2.3), where we computed the position of a point given the position of another point and the displacement between the two points;
- *inverse*: given the attitude of a frame r with respect to a frame w, how can we compute the attitude of w with respect to frame r? Intuitively, this is "similar" to (2.2);

We address each question in the following subsections.

Expressing points in a rotated frame.

Let us assume we are given the coordinates of a point in the frame r, namely $\boldsymbol{p}^{\mathrm{r}}$, and the attitude of r with respect to a frame w, namely $\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}$, where the frames w and r share the same origin. This section addresses the following question: how can we compute the coordinates $\boldsymbol{p}^{\mathrm{w}}$ of the point in the frame w?

It is possible to show that the following relation holds:

$$\boldsymbol{p}^{\mathrm{w}} = \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{p}^{\mathrm{r}} \tag{2.15}$$

Proof. From Fig. 2.4 we can see that the coordinates of $\boldsymbol{p}^{\mathrm{w}}$ can be obtained as a sum of vectors vectors. In the general 3D case, $\boldsymbol{p}^{\mathrm{w}}$ is given by:

$$\boldsymbol{p}^{\mathrm{w}} = \boldsymbol{x}_{\mathrm{r}}^{\mathrm{w}}\, p_x^{\mathrm{r}} + \boldsymbol{y}_{\mathrm{r}}^{\mathrm{w}}\, p_y^{\mathrm{r}} + \boldsymbol{z}_{\mathrm{r}}^{\mathrm{w}}\, p_z^{\mathrm{r}} = \begin{bmatrix} \boldsymbol{x}_{\mathrm{r}}^{\mathrm{w}} & \boldsymbol{y}_{\mathrm{r}}^{\mathrm{w}} & \boldsymbol{z}_{\mathrm{r}}^{\mathrm{w}} \end{bmatrix} \begin{bmatrix} p_x^{\mathrm{r}} \\ p_y^{\mathrm{r}} \\ p_z^{\mathrm{r}} \end{bmatrix} = \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{p}^{\mathrm{r}} \tag{2.16}$$

□

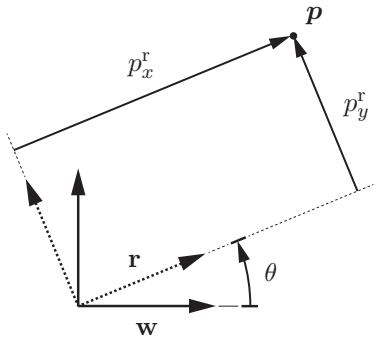

Figure 2.4: Expressing points in rotated frame.

Note: our *notation* helps us transform points using the correct rotation, and one is only "allowed" multiplying a rotation matrix by a vector when the subscript of the rotation matches the superscript (coordinate frame) of the vector.

Rotation composition.

Let us assume we are given the attitude of a frame r with respect to a frame w, namely $\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}$, and the attitude of a frame c with respect to a frame r, namely $\boldsymbol{R}_{\mathrm{c}}^{\mathrm{r}}$. This section addresses the following question: how can we compute the attitude of the frame c with respect to the frame w, i.e., $\boldsymbol{R}_{\mathrm{c}}^{\mathrm{w}}$?

It is possible to show that we can *compose* rotations as follows:

$$\boldsymbol{R}_{\mathrm{c}}^{\mathrm{w}} = \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\, \boldsymbol{R}_{\mathrm{c}}^{\mathrm{r}} \tag{2.17}$$

Proof.

$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\, \boldsymbol{R}_{\mathrm{c}}^{\mathrm{r}} = \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \begin{bmatrix} \boldsymbol{x}_{\mathrm{c}}^{\mathrm{r}} & \boldsymbol{y}_{\mathrm{c}}^{\mathrm{r}} & \boldsymbol{z}_{\mathrm{c}}^{\mathrm{r}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\boldsymbol{x}_{\mathrm{c}}^{\mathrm{r}} & \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\boldsymbol{y}_{\mathrm{c}}^{\mathrm{r}} & \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\boldsymbol{z}_{\mathrm{c}}^{\mathrm{r}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_{\mathrm{c}}^{\mathrm{w}} & \boldsymbol{y}_{\mathrm{c}}^{\mathrm{w}} & \boldsymbol{z}_{\mathrm{c}}^{\mathrm{w}} \end{bmatrix} = \boldsymbol{R}_{\mathrm{c}}^{\mathrm{w}} \tag{2.18}$$

□

Note 1: our *notation* helps us to compose rotations in the correct order, and one is only "allowed" composing two rotations only if the subscript of the first rotation matches the superscript of the subsequent one.

Note 2: rotation composition is not commutative in general, hence in general: $\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\, \boldsymbol{R}_{\mathrm{c}}^{\mathrm{r}} \neq \boldsymbol{R}_{\mathrm{c}}^{\mathrm{r}}\, \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}$.

Inverse of a rotation.

Let us assume we are given the attitude of a frame r with respect to a frame w, namely $\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}$. This section addresses the following question: how can we compute the attitude of w with respect to frame r, i.e., $\boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}}$?

It is possible to show that $\boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}}$ can be computed as follows:

$$\boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} = (\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{-1} = (\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}} \tag{2.19}$$

where the second equality follows from the orthogonality of the rotation matrix.

Proof. By definition the attitude of a frame with respect to itself is the identity matrix: $\boldsymbol{R}_{\mathrm{w}}^{\mathrm{w}} = \mathbf{I}_d$. Moreover, using rotation composition we get:

$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} = \boldsymbol{R}_{\mathrm{w}}^{\mathrm{w}} = \mathbf{I}_d \implies \boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} = (\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{-1}\mathbf{I}_d = (\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{-1} \tag{2.20}$$

□

Note: in Example 2.3.2 we observed that the columns $\boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}}$ describes the coordinates of the unit vectors defining the frame r with respect to the frame w. The relation $\boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} = (\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}}$ above suggests that the *rows* of $\boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}}$ represent the unit vectors defining the frame w with respect to to the frame r.

2.3.2 Elementary rotations and Euler angles representation

A representation has to serve different purposes: (i) it allows *storing* information efficiently (e.g., keep track of the rotation of a frame), (ii) it can be used to perform calculations without too much overhead, and (iii) in many cases it must be understandable for a human (e.g., for debugging or monitoring).

The rotation matrix $\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}$ is an excellent representation to perform calculations (Section 2.3.1.1), but is not very "readable" for a human. Moreover, the representation is clearly redundant: in 2D we intuitively need a single angle to represent the orientation of a frame, while a 2D rotation matrix stores 4 scalars. Similarly, one may wonder if we really need 9 scalars to represent a 3D rotation. The following result tells us that we only need 3 parameters to represent an arbitrary 3D rotation.

Theorem 1 (Euler's rotation theorem, 1775)**.** *Any rotation can be written as the product of no more than three elementary rotations (no two consecutive rotations along the same axis), where the elementary rotations are defined as follows:*

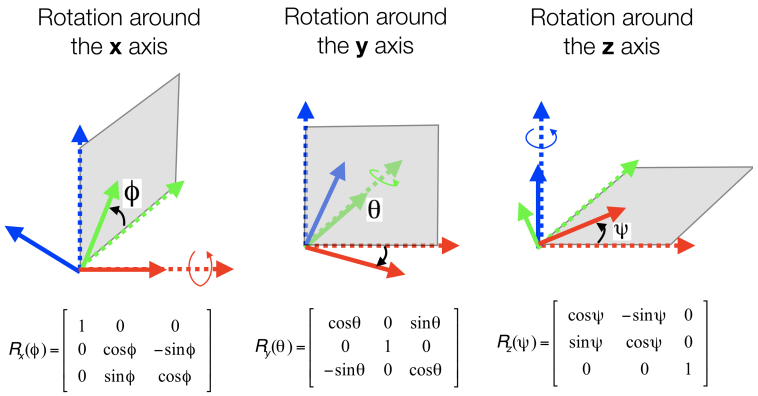

Figure 2.5: Elementary rotations.

Theorem 1 ensures that we only need three parameters, generally called the *Euler angles* to represent a 3D rotation. However, the theorem leaves freedom on the order of the elementary rotations. For instance, one can parametrize the same rotation $\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}$ as:

$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} = \boldsymbol{R}_x(\phi_1)\, \boldsymbol{R}_y(\theta_1)\, \boldsymbol{R}_x(\psi_1) \tag{2.21}$$

or

$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} = \boldsymbol{R}_x(\phi_2)\, \boldsymbol{R}_y(\theta_2)\, \boldsymbol{R}_z(\psi_2) \tag{2.22}$$

resulting in 2 different sets of angles describing the same rotation. This created a plethora of different convention, where each convention adopts a different order.

A particularly popular choice of Euler angles adopt the following order:

$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} = \boldsymbol{R}_z(\gamma)\, \boldsymbol{R}_y(\beta)\, \boldsymbol{R}_x(\alpha) \tag{2.23}$$

where γ is called the *yaw* angle, β is called the *pitch* angle, and α is called the *roll* angle.

The popularity of the roll-pitch-yaw (RPY) angle representation stems from the fact that:

- when the frame w is chosen wisely, these angles assume a very intuitive meaning.
- they provide a *minimal representation* for a rotation: we use 3 parameters to represent 3 rotational degrees of freedom (a rotation matrix uses 9 parameters).

On the downside, RPY and Euler angles in general have 3 main drawbacks:

- *conventions*: there exist multiple conventions and specifying the angles without being precise about the convention may create problems.
- *operations*: while we saw that operations with rotations matrices (rotating points, composition, inverse) reduce to matrix products and transposes, operations with Euler angles typically involve trigonometric functions, which may be slower to compute and more difficult to analyze.
- *singularities*: Euler angles suffer from singularities: some attitudes do not have a unique Euler angle representation. For instance, there is an infinite number of RPY angles representing the attitude of a body with $\pi/2$ pitch angle. In other words, the following product produces the same rotation matrix for any choice of the angle δ:

$$\boldsymbol{R} = \boldsymbol{R}_z(\delta)\,\boldsymbol{R}_y(\pi/2)\,\boldsymbol{R}_x(\alpha+\delta) \tag{2.24}$$

Proof. Let us proceed by inspection and compute the product $\boldsymbol{R}_z(\delta)\,\boldsymbol{R}_y(\pi/2)$:

$$\boldsymbol{R}_z(\delta)\,\boldsymbol{R}_y(\pi/2) = \begin{bmatrix} \cos(\delta) & -\sin(\delta) & 0 \\ \sin(\delta) & \cos(\delta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\sin(\delta) & \cos(\delta) \\ 0 & \cos(\delta) & \sin(\delta) \\ -1 & 0 & 0 \end{bmatrix} \tag{2.25}$$

Now doing the overall multiplication:

$$\boldsymbol{R}_z(\delta)\,\boldsymbol{R}_y(\pi/2)\,\boldsymbol{R}_x(\alpha+\delta) = \begin{bmatrix} 0 & -\sin(\delta) & \cos(\delta) \\ 0 & \cos(\delta) & \sin(\delta) \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha+\delta) & -\sin(\alpha+\delta) \\ 0 & \sin(\alpha+\delta) & \cos(\alpha+\delta) \end{bmatrix} =$$

$$\begin{bmatrix} 0 & -\sin(\delta)\cos(\delta+\alpha)+\cos(\delta)\sin(\delta+\alpha) & \sin(\delta)\sin(\delta+\alpha)+\cos(\delta)\cos(\delta+\alpha) \\ 0 & \sin(\delta)\sin(\delta+\alpha)+\cos(\delta)\cos(\delta+\alpha) & \sin(\delta)\cos(\delta+\alpha)-\cos(\delta)\sin(\delta+\alpha) \\ -1 & 0 & 0 \end{bmatrix} \overset{\text{(from trigonometry)}}{=}$$

$$\begin{bmatrix} 0 & \sin((\delta+\alpha)-\delta) & \cos((\delta+\alpha)-\delta) \\ 0 & \cos((\delta+\alpha)-\delta) & -\sin((\delta+\alpha)-\delta) \\ -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & \sin(\alpha) & \cos(\alpha) \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ -1 & 0 & 0 \end{bmatrix}$$

which is independent from δ, proving the claim.

□

Therefore we say that the RPY has a singularity for pitch angle equal to $\pi/2$ (same happens for $-\pi/2$). This singularity is also called *Gimbal lock*.

An excellent interactive tool to visualize and understand RPY angles is the "tripleangle" script in Peter Corke's robotics toolbox (tested: release 10.2, Matlab 2016b). In particular, one can use the tool to gain a better understanding of:

- the meaning of the RPY angles
- the notion of singularity

It is known that any 3 parameter representation cannot be free of singularities [4]. Singularities occur when the second Euler angle aligns the first and the third rotation axes which causes the loss of a degree of freedom. Depending on the order of the Euler angles, singularities occur at $\{0, \pi\}$ or $\{+\pi/2, -\pi/2\}$ angles for the second rotation.

2.3.3 Axis-angle representation

Another theorem from Euler implicitly suggests a different parametrization for rotations:

Theorem 2 (Euler's rotation theorem, 1776)**.** *Every rotation can be described as a rotation of an angle θ around an axis $\boldsymbol{u}$ (with $\|\boldsymbol{u}\| = 1$), not necessarily aligned with the Cartesian axes.*

This suggests the *axis-angle* representations, which encodes a 3D rotation using the pair $(\boldsymbol{u}, \theta)$. This representation is very important since

- can be easily visualized and understood
- only stores 4 parameters (3 if we use $\|\boldsymbol{u}\| = 1$)
- has interesting relations with Lie group theory (next lecture).

However, it has few drawbacks (the notation $R(\boldsymbol{u}, \theta)$ denotes a rotation of an angle θ around the axis $\boldsymbol{u}$):

- operations (composition, rotating points) with the axis-angle representation typically involve trigonometric functions, which may be slower to compute and more difficult to analyze. The inverse of a rotation, however, is easy to compute: $R(\boldsymbol{u}, \theta)^{-1} = R(\boldsymbol{u}, -\theta) = R(-\boldsymbol{u}, \theta)$
- the representation is not unique, e.g., $R(\boldsymbol{u}, \theta) = R(\boldsymbol{u}, \theta + 2k\pi)$ for any integer k.

Conversions.

- From axis-angle to rotation matrix (*Rodrigues' rotation formula*): given a rotation angle θ and a rotation axis $\boldsymbol{u}$ that describe the attitude of the frame r with respect to the frame w (with $\boldsymbol{u}$ expressed in the frame w), the rotation matrix $\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}$ can be computed as:

$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} = \cos(\theta)\mathbf{I}_3 + \sin(\theta)[\boldsymbol{u}]_\times + (1 - \cos(\theta))\boldsymbol{u}\boldsymbol{u}^\mathsf{T} \tag{2.26}$$

  where

$$[\boldsymbol{u}]_\times \doteq \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \tag{2.27}$$

  is a skew symmetric matrix (the *cross product matrix*). Note that the cross product matrix allows writing the cross product between 2 vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$:

$$\boldsymbol{v}_1 \times \boldsymbol{v}_2 = [\boldsymbol{v}_1]_\times \boldsymbol{v}_2 \tag{2.28}$$

  Eq. (2.26) is also known as the *Rodrigues' rotation formula*.

- From rotation matrix to axis-angle: given a rotation matrix $\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}$ that describes the attitude of the frame r with respect to the frame w, the corresponding axis-angle representation $(\boldsymbol{u}, \theta)$ can be computed as:
  - angle: using (2.26), we note that

$$\mathrm{tr}\left(\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\right) = 1 + 2\cos(\theta) \tag{2.29}$$

    which can be used to compute the rotation angle:

$$\theta = \arccos\left(\frac{\mathrm{tr}\left(\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\right) - 1}{2}\right) \tag{2.30}$$

- axis: it we try to rotate a vector equal to the axis $\boldsymbol{u}$, then this vector does not change. In mathematical terms:
$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\boldsymbol{u} = \boldsymbol{u} \tag{2.31}$$
which means that $\boldsymbol{u}$ can be computed as the eigenvector corresponding to the eigenvalue 1 of the rotation matrix. Therefore, we can compute the axis by computing the eigenvectors of the matrix $\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}$.
Interestingly, it turns out that the three eigenvalues of a 3D rotation matrix are always $\{1, \cos\theta \pm i \sin\theta\}$. By inspection (i.e., looking at the *characteristic polynomial*), one can also see that a 2D rotation matrix has always eigenvalues $\{\cos\theta \pm i \sin\theta\}$.

2.3.4 Quaternion representation

If 3-parameter representations are ideal for storage (but have singularities and requires trigonometry) and rotation matrices are singularity-free (but largely over-parametrize the attitude), the natural question is: is there a representation that is singularity free and uses less parameters than a rotation matrix?

W.R. Hamilton provided a positive answer to this question in 1843, by introducing the *quaternion* representation. Depending on the context a quaternion is denoted as a column vector

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{2.32}$$

or as an ipercomplex number, i.e., $\boldsymbol{q} = iq_1 + jq_2 + kq_3 + q_4$, with i, j, k satisfying:

$$i^2 = j^2 = k^2 = ijk = -1$$
$$ij = -ji = k \qquad jk = -kj = i \qquad ki = -ik = j$$

A quaternion can be also written as $\boldsymbol{q} = [\boldsymbol{v}^{\mathsf{T}}\ s]^{\mathsf{T}}$, where the first element $\boldsymbol{v} \in \mathbb{R}^3$ is the *vector part* of the quaternion, while the last element s is the *scalar part*.[1]

Unit quaternions, i.e., quaternions having unit norm, are compact representations for rotations and have been used in several applications of computer vision and 3D navigation (examples of use in navigation problems are [1–3, 5]). The basic insight behind the quaternion representation is to "rearrange" the axis-angle parametrization $(\boldsymbol{u}, \theta)$ as follows:

$$\boldsymbol{q} = \begin{bmatrix} \sin(\theta/2)\boldsymbol{u} \\ \cos(\theta/2) \end{bmatrix} \tag{2.33}$$

It is possible to show that the rotation matrix corresponding to a unit quaternion $\boldsymbol{q}$ is given by:

$$\boldsymbol{R}(\boldsymbol{q}) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_2 + q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{2.34}$$

Clearly, (2.33) stores all the information we need to represent a 3D attitude (e.g., we can easily extract an axis-angle representation from (2.33)). The remaining questions are then: *does this representation exhibit*

[1] Also in the case of quaternions, multiple conventions exist. We adopt the somehow unusual convention of putting the scalar part as last entry in the quaternion (most commonly it is put first) to keep some similarity with the homogeneous coordinates discussed in below.

singularities? does this representation make computation easy? Surprisingly, quaternions do not have singularities. The only ambiguity follows from the fact that, if we denote with $\boldsymbol{R}(\boldsymbol{q})$ the rotation matrix corresponding to the quaternion $\boldsymbol{q}$, then $\boldsymbol{q}$ and $-\boldsymbol{q}$ result in the same rotation matrix:

$$\boldsymbol{R}(\boldsymbol{q}) = \boldsymbol{R}(-\boldsymbol{q}) \tag{2.35}$$

hence each attitude can be represented by both a quaternion $\boldsymbol{q}$ and its negation $-\boldsymbol{q}$.

Proof. Clearly a rotation of an angle θ and of an angle $\theta+2\pi$ produce the same attitude, hence, the following quaternion describes the same attitude of $\boldsymbol{q} \doteq [\sin(\theta/2)\boldsymbol{u}^{\mathsf{T}} \ \cos(\theta/2)]^{\mathsf{T}}$:

$$\begin{bmatrix} \sin((\theta+2\pi)/2)\boldsymbol{u} \\ \cos(\theta+2\pi) \end{bmatrix} = \begin{bmatrix} \sin(\theta/2+\pi)\boldsymbol{u} \\ \cos(\theta/2+\pi) \end{bmatrix} \overset{\text{(some trigonometry)}}{=} \begin{bmatrix} -\sin(\theta/2)\boldsymbol{u} \\ -\cos(\theta/2) \end{bmatrix} = -\boldsymbol{q} \tag{2.36}$$

□

Quaternion composition. Given two quaternions $\boldsymbol{q}_a = [q_{a,1} \ q_{a,2} \ q_{a,3} \ q_{a,4}]^{\mathsf{T}}$ and $\boldsymbol{q}_b = [q_{b,1} \ q_{b,2} \ q_{b,3} \ q_{b,4}]^{\mathsf{T}}$, we define the *quaternion product* $\boldsymbol{q}_c = \boldsymbol{q}_a \otimes \boldsymbol{q}_b$, which can be computed as:

$$\boldsymbol{q}_c = \begin{bmatrix} q_{a,4} & -q_{a,3} & q_{a,2} & q_{a,1} \\ q_{a,3} & q_{a,4} & -q_{a,1} & q_{a,2} \\ -q_{a,2} & q_{a,1} & q_{a,4} & q_{a,3} \\ -q_{a,1} & -q_{a,2} & -q_{a,3} & q_{a,4} \end{bmatrix} \begin{bmatrix} q_{b,1} \\ q_{b,2} \\ q_{b,3} \\ q_{b,4} \end{bmatrix} = \Upsilon(\boldsymbol{q}_a)\, \boldsymbol{q}_b$$

or, equivalently, as:

$$\boldsymbol{q}_c = \begin{bmatrix} q_{b,4} & q_{b,3} & -q_{b,2} & q_{b,1} \\ -q_{b,3} & q_{b,4} & q_{b,1} & q_{b,2} \\ q_{b,2} & -q_{b,1} & q_{b,4} & q_{b,3} \\ -q_{b,1} & -q_{b,2} & -q_{b,3} & q_{b,4} \end{bmatrix} \begin{bmatrix} q_{a,1} \\ q_{a,2} \\ q_{a,3} \\ q_{a,4} \end{bmatrix} = \bar{\Upsilon}(\boldsymbol{q}_b)\, \boldsymbol{q}_a$$

We can now discuss how to compose rotations using unit quaternions. Let us first go back to our original notation and denote with $\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}}$ the quaternion describing the attitude of frame r with respect to frame w. Also, let us assume we are given the attitude of a frame r with respect to a frame w as a quaternion, namely $\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}}$, and the attitude of a frame c with respect to a frame r, namely $\boldsymbol{q}_{\mathrm{c}}^{\mathrm{r}}$. How can we compute the attitude of the frame c with respect to the frame w, i.e., $\boldsymbol{q}_{\mathrm{c}}^{\mathrm{w}}$?

It turns out that we can compose rotations using the quaternion product introduced above:

$$\boldsymbol{q}_{\mathrm{c}}^{\mathrm{w}} = \boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}} \otimes \boldsymbol{q}_{\mathrm{c}}^{\mathrm{r}} \tag{2.37}$$

Inverse of a quaternion. Let us assume we are given the attitude of a frame r with respect to a frame w in quaternion notation, namely $\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}}$. This section addresses the following question: how can we compute the attitude of w with respect to frame r, i.e., $\boldsymbol{q}_{\mathrm{w}}^{\mathrm{r}}$?

It turns out that computing the inverse notation in quaternion form is straightforward and can be computed as (proof hint, think about inverse rotation in axis-angle representation):

$$\boldsymbol{q}_{\mathrm{w}}^{\mathrm{r}} = \begin{bmatrix} -\boldsymbol{v}_{\mathrm{r}}^{\mathrm{w}} \\ s_{\mathrm{r}}^{\mathrm{w}} \end{bmatrix} \doteq (\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}})^{-1} \tag{2.38}$$

where $\boldsymbol{v}_{\mathrm{r}}^{\mathrm{w}}$ and $s_{\mathrm{r}}^{\mathrm{w}}$ are the vector and the scalar part of $\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}}$, respectively. In the last equality in (2.38) we defined the "inverse" of a quaternion to be the operation that switches the sign of the vector part of a quaternion.

It also holds:

$$(\boldsymbol{q}_a \otimes \boldsymbol{q}_b \otimes \ldots \otimes \boldsymbol{q}_N)^{-1} = \boldsymbol{q}_N^{-1} \otimes \ldots \otimes \boldsymbol{q}_b^{-1} \otimes \boldsymbol{q}_a^{-1}$$

Expressing points in a rotated frame. Let us assume we are given the coordinates of a point in the frame r, namely $\boldsymbol{p}^{\mathrm{r}}$, and the attitude of r with respect to a frame w in quaternion notation, namely $\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}}$, where the frames w and r share the same origin. This section addresses the following question: how can we compute the coordinates $\boldsymbol{p}^{\mathrm{w}}$ of the point in the frame w?

The following expression expresses point $\boldsymbol{p}^{\mathrm{r}}$ in the reference frame w:

$$\boldsymbol{p}^{\mathrm{w}} = (\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}}) \otimes \begin{bmatrix} \boldsymbol{p}^{\mathrm{r}} \\ 1 \end{bmatrix} \otimes (\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}})^{-1} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{p}^{\mathrm{r}} \\ 1 \end{bmatrix} \tag{2.39}$$

An alternative expression is given as follows:

$$\boldsymbol{p}^{\mathrm{w}} = (\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}}) \otimes \begin{bmatrix} \boldsymbol{p}^{\mathrm{r}} \\ 0 \end{bmatrix} \otimes (\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}})^{-1} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{p}^{\mathrm{r}} \\ 0 \end{bmatrix} \tag{2.40}$$

Interestingly, we can augment a 3D position with an extra entry (1 or 0) and use quaternion product and inverse to express points in a different reference frame. As we will see in Section (2.4), stacking an entry equal to 1 to a position or translation produces a representation known as *homogeneous coordinates*.

Are quaternions the "best" representation for rotations? Quaternions are the minimal representation of a rotation that does not have singularities (we already observed that all 3-parameter representations are affected by singularities), hence they are excellent to store data. Moreover, the allow to perform computation without resorting to trigonometry. In particular, while rotation matrix multiplication (composition) involves 27 products of the entries of the matrices, quaternion multiplication (composition) only involves 16 products of entries; this computational advantage makes quaternion an excellent representation for computer graphics, where one has to transform many points quickly for 3D rendering.

A minor drawback is that they are slightly harder to parse (for a human) with respect to Euler angles. Then, are quaternions the "best" representation for rotations?

In the next lecture, we will see that a very subtle issue due to the sign ambiguity in eq. (2.35) may create some difficulties in certain estimation problems, hence, while being a very powerful estimation tool, we will mostly prefer to use rotation matrices.

2.4 Poses and rigid-body transformations

In Section 2.2 we observed that we can describe the position of a point using a 3D vector, while in Section 2.3.1 we observed that we can describe the orientation of a frame using a rotation matrix. Therefore, we are now ready to fully characterize the position and attitude of a frame r (attached to a body) with respect to another frame w. In particular, if we call $\boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}}$ the position of the origin of r with respect to w and $\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}$ the attitude of r with respect to w, then the pair:

$$(\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}, \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}}) \tag{2.41}$$

fully characterizes the geometry of r with respect to w, or, more precisely, the *pose* (i.e., position and orientation) of r with respect to w. A pose is fully defined by 6 parameters (3 for the translation and 3 for the attitude). In the following section, we will see that it is actually convenient to assemble the rotation and the translation defining a pose into a suitable matrix:

$$\boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} & \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}} \\ \boldsymbol{0}_d^{\mathsf{T}} & 1 \end{bmatrix} \tag{2.42}$$

where $d = 2$ in 2D problems and $d = 3$ in 3D.

Rigid-body transformations.

While in Section 2.3.1 we discussed how to transform points between rotated coordinate frames (sharing the same origin), we now discuss expressing points in reference frames with arbitrary pose (i.e., not necessarily sharing the same origin). Assume we are given the coordinates $\boldsymbol{p}^{\mathrm{r}}$ of a point $\boldsymbol{p}$, expressed in the reference frame r and that we know the relative pose of the reference frame r with respect to the frame w, namely $\boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}}$. Then the position of point $\boldsymbol{p}$ with respect to frame w is given by:

$$\boldsymbol{p}^{\mathrm{w}} = \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{p}^{\mathrm{r}} + \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}} \tag{2.43}$$

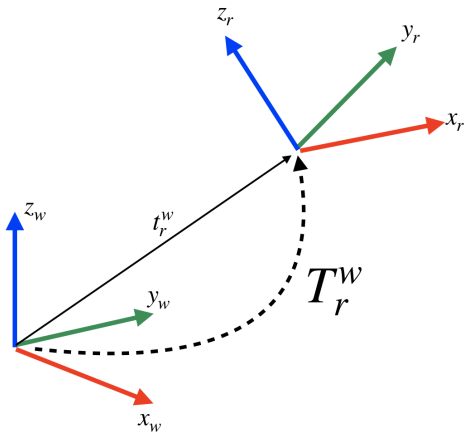

Figure 2.6: Rigid-body transformations.

We can write (2.43) more succinctly by introducing the *homogeneous coordinate* representation for a 3D point $\boldsymbol{p}^{\mathrm{r}}$, which is simply obtained by concatenating a "1" to the vector containing the coordinates of the point:

$$\tilde{\boldsymbol{p}}^{\mathrm{r}} = \begin{bmatrix} \boldsymbol{p}^{\mathrm{r}} \\ 1 \end{bmatrix} \tag{2.44}$$

Using homogeneous coordinates we can write (2.43) in matrix form as:

$$\tilde{\boldsymbol{p}}^{\mathrm{w}} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} & \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}} \\ \boldsymbol{0}_d^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}^{\mathrm{r}} \\ 1 \end{bmatrix} = \boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}} \tilde{\boldsymbol{p}}^{\mathrm{r}} \tag{2.45}$$

hence generalizing the pure rotation case (2.15).

Pose composition. Let us assume we are given the pose of a frame r with respect to a frame w, namely $\boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}}$, and the pose of a frame c with respect to a frame r, namely $\boldsymbol{T}_{\mathrm{c}}^{\mathrm{r}}$. This section addresses the following question: how can we compute the pose of the frame c with respect to the frame w, i.e., $\boldsymbol{T}_{\mathrm{c}}^{\mathrm{w}}$?

Similarly to the rotation case, we can compose poses by matrix multiplication:

$$\boldsymbol{T}_{\mathrm{c}}^{\mathrm{w}} = \boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}} \, \boldsymbol{T}_{\mathrm{c}}^{\mathrm{r}} \tag{2.46}$$

Proof.

$$\boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}} \, \boldsymbol{T}_{\mathrm{c}}^{\mathrm{r}} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} & \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}} \\ \boldsymbol{0}_d^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_{\mathrm{c}}^{\mathrm{r}} & \boldsymbol{t}_{\mathrm{c}}^{\mathrm{r}} \\ \boldsymbol{0}_d^{\mathsf{T}} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{R}_{\mathrm{c}}^{\mathrm{r}} & \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{t}_{\mathrm{c}}^{\mathrm{r}} \\ \boldsymbol{0}_d^{\mathsf{T}} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{c}}^{\mathrm{w}} & \boldsymbol{t}_{\mathrm{c}}^{\mathrm{w}} \\ \boldsymbol{0}_d^{\mathsf{T}} & 1 \end{bmatrix} = \boldsymbol{T}_{\mathrm{c}}^{\mathrm{w}} \tag{2.47}$$

□

Inverse of a pose.

Let us assume we are given the pose of a frame r with respect to a frame w, namely $\boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}}$. This section addresses the following question: how can we compute the pose of w with respect to frame r, i.e., $\boldsymbol{T}_{\mathrm{w}}^{\mathrm{r}}$?

It is possible to show that $\boldsymbol{T}_{\mathrm{w}}^{\mathrm{r}}$ can be computed as follows:

$$\boldsymbol{T}_{\mathrm{w}}^{\mathrm{r}} = (\boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}})^{-1} = \begin{bmatrix} (\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}} & -(\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}}\boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}} \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix} \tag{2.48}$$

Note that unlike the rotation-only case, when dealing with poses, the inverse is no longer the transpose, since the matrix $\boldsymbol{T}_{\mathrm{w}}^{\mathrm{r}}$ is not an orthogonal matrix in general.

Proof. Instead of using the compact notation $\boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}}$, let us use $\boldsymbol{t}_{\mathrm{wr}}^{\mathrm{w}}$ to stress the fact that this vector represents the translation between the origin of frame w and r expressed with respect to frame w. Then we can develop $(\boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}})^{-1}$ as:

$$\begin{bmatrix} (\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}} & -(\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}}\boldsymbol{t}_{\mathrm{wr}}^{\mathrm{w}} \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} & -\boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}}\boldsymbol{t}_{\mathrm{wr}}^{\mathrm{w}} \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} & -\boldsymbol{t}_{\mathrm{wr}}^{\mathrm{r}} \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} & \boldsymbol{t}_{\mathrm{rw}}^{\mathrm{r}} \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix} \tag{2.49}$$

where in the last equality we used eq. (2.4). Noting that $\boldsymbol{t}_{\mathrm{rw}}^{\mathrm{r}}$ is what we typically call $\boldsymbol{t}_{\mathrm{w}}^{\mathrm{r}}$, we can recognize the last matrix to be $\boldsymbol{T}_{\mathrm{w}}^{\mathrm{r}}$, concluding the proof. □

References

[1] W.G. Breckenridge. Quaternions - proposed standard conventions. In *JPL, Tech. Rep. INTEROFFICE MEMORANDUM IOM 343-79-1199*, 1999.

[2] E.J. Lefferts, F.L. Markley, and M. D. Shuster. Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 5(5):417–429, 1982.

[3] M.D. Shuster. A survey of attitude representations. *Journal of the Astronautical Sciences*, 41(4):439–517, 1993.

[4] J. Stuelpnagel. On the Parametrization of the Three-Dimensional Rotation Group. *SIAM Review*, 6(4):422–430, 1964.

[5] N. Trawny and S.I. Roumeliotis. Indirect kalman filter for 3D attitude estimation. *Mars Lab, Technical Report Number 2005-002, Rev. 57*, 2005.