Math for Computer Vision and Navigation

Ferdi van der Heijden ▸ University of Twente - RAM ▸ 1/9/2018

This document summarizes the mathematical background needed to study the principles for 3D computer vision and navigation technology.

Contents

A. Vector spaces

Vectors are mathematical objects that have some prescribed properties. Vectors are often denoted by bold faced, lower case characters, e.g. $\mathbf{p}$, $\mathbf{q}$ .and $\mathbf{r}$. By definition, a *vector space* is a set of vectors in which two operations are specified:

- The vector addition: $\mathbf{r} = \mathbf{p} + \mathbf{q}$
- The scalar product: $\mathbf{r} = \alpha\mathbf{p}$

The vector addition is an operation on two vectors that produces a third vector. The scalar product is an operation on one vector and a scalar. It produces another vector.

A vector space is only a vector space if it satisfies a number of conditions, called axioms. For instance, addition must be commutative: $\mathbf{p} + \mathbf{q} = \mathbf{q} + \mathbf{p}$. There are eight of such axioms.

We can think of vectors as arrows: entities that have a length and a direction. This enables easy visualization of vectors:

These graphs might suggest that vectors are entities in a 2D or 3D space, but vector spaces are not confined to that.

Example: polynomial functions

The set of all polynomial functions of degree N:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_N x^N$$

is a vector space. This follows readily by considering two vectors: $f(x) = a_0 + \cdots + a_N x^N$ and $g(x) = b_0 + \cdots + b_N x^N$, and showing that vector addition, $f(x) + g(x)$, and scalar product $\alpha f(x)$, both end up in polynomials of the same degree, and thus in vectors that are lying in the same vector space. Also, it should be verified that these two operations comply with the eight axioms.

Many vector spaces can be represented by an ordered set of elements. For the polynomial function example, these elements are the coefficient $a_0, \cdots, a_N$ .Usually, these elements are vertically aligned in a column, e.g.:

$$\mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}$$

in which the set a_n with $n = 0, \cdots, N$ are the elements. For an inline expression this takes too much space, and the transposed notation, $\mathbf{a} = \begin{bmatrix} a_0 & a_1 & \cdots & a_N \end{bmatrix}^{\mathrm{T}}$ is often used. Note that, without the transpose, $\mathbf{a} = \begin{bmatrix} a_0 & a_1 & \cdots & a_N \end{bmatrix}$, the result is a row vector.

The number of elements is the *dimension* of the vector. In the polynomial function example, the dimension was $N+1$. Thus, a polynomial function of degree N has a dimension of $N+1$. Of course, this is confusing. Therefore, unlike polynomial functions, the enumeration of the elements almost always starts at 1, thus: $\mathbf{a} = \begin{bmatrix} a_1 & a_2 & \cdots & a_N \end{bmatrix}^{\mathrm{T}}$, so that the dimension of this vector is just N.

The space of N-dimensional vectors, of which the elements are real numbers, is denoted by $\mathbb{R}^N$. If the elements are complex numbers, the space is denoted by $\mathbb{C}^N$.

A.1 Properties of vector spaces

A.1.1 The norm of a vector space

The *norm* of a vector space is an axiomatic defined assignment of a real number to a vector. If $\mathbf{f}$ is a vector with elements f_n, then its norm is denoted by $\|\mathbf{f}\|$. A norm can be defined freely as long as it complies with the following three axioms:

$$\begin{aligned} &1) \quad \|\mathbf{f}\| \geq 0, \text{ where } \|\mathbf{f}\| = 0 \text{ if and only if } \mathbf{f} = \mathbf{0} \\ &2) \quad \|\alpha\mathbf{f}\| = |\alpha|\|\mathbf{f}\| \\ &3) \quad \|\mathbf{f} + \mathbf{g}\| \leq \|\mathbf{f}\| + \|\mathbf{g}\| \end{aligned} \tag{1}$$

A well-known norm in $\mathbb{R}^N$ is the so-called L_p norm:

$$\|\mathbf{f}\|_p = \sqrt[p]{\sum_{n=1}^{N} |f_n|^p} \quad \text{with } p \geq 1 \tag{2}$$

With $p = 2$, we have the Euclidean norm which is often used because it measures the geometrical length of a vector in the 2- and 3-dimensional case. For this reason, $\|\mathbf{f}\|$ always denotes the Euclidean norm, unless it is specified otherwise.

If we are only interested in the direction of a vector, and not in its length, then we may want to normalize the vector:

$$\mathbf{n}_\mathbf{p} = \frac{\mathbf{p}}{\|\mathbf{p}\|} \tag{3}$$

The vector $\mathbf{n}_\mathbf{p}$ has unit length, and holds only information about the direction of $\mathbf{p}$.

A.1.2 Inner product

The *inner product* is an axiomatic defined assignment of a real number to a pair of vectors. Most commonly, this assignment is the so-called *vector dot product*. Let $\mathbf{p}$ and $\mathbf{q}$ be two vectors from $\mathbb{R}^N$, that is $\mathbf{p} = [p_1 \quad \cdots \quad p_N]^\mathrm{T}$ and $\mathbf{q} = [q_1 \quad \cdots \quad q_N]^\mathrm{T}$. The inner product, denoted by $(\mathbf{p},\mathbf{q})$, takes the form of the dot product:

$$(\mathbf{p},\mathbf{q}) = \mathbf{p}^\mathrm{T}\mathbf{q} = \mathbf{q}^\mathrm{T}\mathbf{p} = \sum_{n=1}^{N} p_n q_n \tag{4}$$

Note that the Euclidean norm of a vector $\mathbf{p}$ follows from:

$$\|\mathbf{p}\|^2 = \mathbf{p}^\mathrm{T}\mathbf{p} \tag{5}$$

Associated with the inner product is an *angle* φ between the two vectors, defined by:

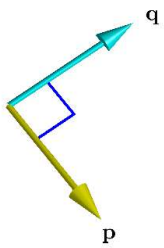
$$(\mathbf{p},\mathbf{q}) = \|\mathbf{p}\|\|\mathbf{q}\|\cos\varphi \tag{6}$$

In the 2 and 3-dimensional case, the angle φ indeed represents the angle between the arrows.

Matlab code for inner product:
If `p` and `q` are column vectors, then `p'*q` or `dot(p,q)` provides the inner product. If `p` and `q` are row vectors, then `p*q'` would be the inner product.

A.1.3 Orthogonality

Two vectors are said to be *orthogonal*, $\mathbf{p} \perp \mathbf{q}$, if $\varphi = 90^0$, which is equivalent to $\mathbf{p}^T\mathbf{q} = 0$. A graphical representation is given below.



The blue rectangle symbolizes the orthogonality of the two vectors.

In the case of orthogonality, Pythagoras' theorem applies:

$$\mathbf{p} \perp \mathbf{q} \quad \Leftrightarrow \quad \|\mathbf{p}+\mathbf{q}\|^2 = \|\mathbf{p}\|^2 + \|\mathbf{q}\|^2 \tag{7}$$

Example: Pythagoras in 2D

Suppose: $\mathbf{a} = [a_1 \quad 0]^\mathrm{T}$ and $\mathbf{b} = [0 \quad b_2]^\mathrm{T}$. Then clearly $\mathbf{a} \perp \mathbf{b}$. We form a third vector: $\mathbf{c} = \mathbf{a} + \mathbf{b}$. Then according to Pythagoras, the length $\|\mathbf{c}\|$ follows from: $\|\mathbf{c}\|^2 = a_1^2 + b_2^2$ which indeed is $\|\mathbf{a}\|^2 + \|\mathbf{b}\|^2$.

A.1.4 Projection

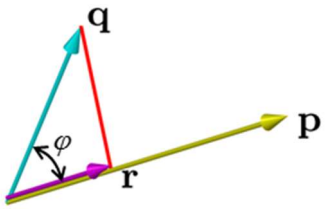
The *projection* $\mathbf{r}$ of $\mathbf{q}$ on $\mathbf{p}$ is defined by

$$\mathbf{r} = \alpha\mathbf{p} \quad \text{such that} \quad (\mathbf{p},\mathbf{r}) = (\mathbf{p},\mathbf{q}) \tag{8}$$

Since $(\mathbf{p},\mathbf{r}) = \alpha(\mathbf{p},\mathbf{p}) = \alpha\|\mathbf{p}\|^2$, we find

$$\alpha = \frac{(\mathbf{p},\mathbf{q})}{\|\mathbf{p}\|^2} = \frac{\mathbf{p}^\mathrm{T}\mathbf{q}}{\mathbf{p}^\mathrm{T}\mathbf{p}} = \|\mathbf{q}\|\cos\varphi \tag{9}$$

See figure, below.



A.1.5 Distance measures

A *distance measure* is a non-negative real number that is assigned to a pair of vectors, and that complies with three axioms. $\rho(\mathbf{x},\mathbf{y})$ is a distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ if the following three axioms are met:

$$\begin{aligned} &1) \quad \rho(\mathbf{x},\mathbf{y}) = 0 \ \text{ if and only if } \mathbf{x} = \mathbf{y} \\ &2) \quad \rho(\mathbf{x},\mathbf{y}) = \rho(\mathbf{y},\mathbf{x}) \\ &3) \quad \rho(\mathbf{x},\mathbf{y}) \le \rho(\mathbf{x},\mathbf{z}) + \rho(\mathbf{y},\mathbf{z}) \end{aligned} \tag{10}$$

Any norm $\|\ \|$ of a vector can be used to define a distance measure:

$$\rho(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\| \tag{11}$$

However, there are also distance measures that are not derived from a norm. For instance, with the definition:

$$\rho(\mathbf{x},\mathbf{y}) = \begin{cases} 0 & \text{if } \mathbf{x} = \mathbf{y} \\ 1 & \text{if } \mathbf{x} \neq \mathbf{y} \end{cases} \tag{12}$$

$\rho(.,.)$ complies with the three axioms, but $\rho(\mathbf{x},\mathbf{0})$ is not a norm of $\mathbf{x}$.

B. Matrix algebra

A matrix is an arrangement of numbers on a 2D grid:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1M} \\ \vdots & & & \vdots \\ h_{N1} & h_{N2} & \cdots & h_{NM} \end{bmatrix} \tag{13}$$

The grid consists of N rows and M columns. As such the matrix consists of NM numbers, and the matrix is said to be $N \times M$ dimensional. Usually, a matrix is denoted by a bold-faced capital, such as $\mathbf{H}$. The elements are denoted by the same character, but now italic and in lower case, e.g. h_{ij}. The set of $N \times M$ matrices is denoted by $\mathbb{R}^{N \times M}$.

The matrix addition and scalar multiplication for matrices are defined as follows[1]:

$$\begin{aligned} \mathbf{F} &= \mathbf{G} + \mathbf{H} \quad \text{with elements } f_{ij} = g_{ij} + h_{ij} \\ \mathbf{F} &= \alpha \mathbf{H} \quad \text{with elements } f_{ij} = \alpha h_{ij} \end{aligned} \tag{14}$$

A third operation on a matrix is the scalar addition:

$$\mathbf{F} = \mathbf{G} + \alpha \quad \text{with elements } f_{ij} = g_{ij} + \alpha \tag{15}$$

Note, that a matrix with just one column, i.e. $M = 1$, is equivalent to an N-dimensional vector: $\mathbb{R}^{N \times 1} \sim \mathbb{R}^N$.

'Dimension' in Matlab

For matrices, the term *dimension* can have different connotations:

- *Number of array dimensions*. This is the dimension of the orthogonal grid, which in the case of matrices is always 2. Arrays can have higher dimensional grids. Full color images, and also volumetric medical data, have a dimension 3. Matlab function: **`ndims`**.
- *Array dimensions*. These are the sizes of each dimension of an array. For an $N \times M$ matrix, they are N and M. Matlab function: **`size`**.
- *Number of array elements*. As matrices fulfill the requirements for vector spaces, a matrix can also be regarded as a vector. In this connotation, the dimension of an $N \times M$ matrix is NM. Matlab function: **`numel`**.

The elements in Matlab arrays and matrices can be rearranged on different orthogonal grids. For instance, a 6×12 matrix can be rearranged on an 8×9 grid. Matlab function: **`reshape`**.
To rearrange an $N \times M$ matrix to a $NM \times 1$ array (i.e. to vectorize the data), use the semicolon operator, e.g. **`H(:)`**.

B.1 Matrix-vector and matrix-matrix products

B.1.1 Matrix-vector product

Matrices offer a very concise way to represent a system of linear equations. Consider, for instance, the following set of equations:

$$\begin{aligned} y_1 &= h_{11}x_1 + h_{12}x_2 + h_{13}x_3 \\ y_2 &= h_{21}x_1 + h_{22}x_2 + h_{23}x_3 \end{aligned}$$

By introducing:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \end{bmatrix}$$

the equation is fully defined by:

$$\mathbf{y} = \mathbf{H}\mathbf{x} \tag{16}$$

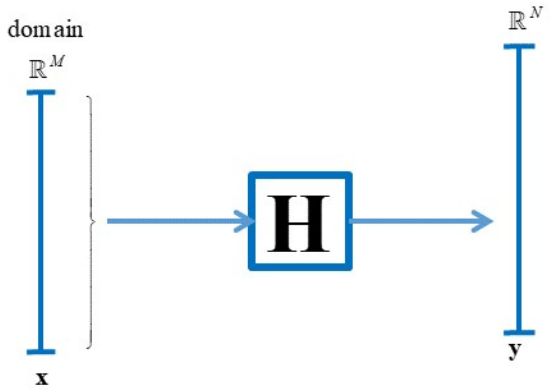
This equation represents the *matrix-vector product*. For the case in which $\mathbf{x}$ is M-dimensional and $\mathbf{y}$ is N-dimensional, the matrix $\mathbf{H}$ must be $N \times M$ dimensional. The multiplication is defined by:

$$y_i = \sum_{j=1}^{M} h_{ij} x_j \quad \text{for } i = 1, \cdots, N \tag{17}$$

B.1.2 Linear mappings

Matrix-vector products can be regarded as linear mappings of vectors. Consider, for instance, a vector $\mathbf{x} \in \mathbb{R}^M$ and an $N \times M$ dimensional matrix $\mathbf{H}$. Then the multiplication $\mathbf{H}\mathbf{x}$ will produce a vector $\mathbf{y} = \mathbf{H}\mathbf{x}$ in an N-dimensional space. The vector space $\mathbb{R}^M$ is the *domain* of the mapping $\mathbf{H}$. Each vector in this set is mapped to an other vector which is lying in the vector space $\mathbb{R}^N$. The procedure is illustrated in the figure below.

[1] These operations fulfill the eight axioms for linear vector spaces. Therefore, the set of all matrices of a given size form a vector space by itself.



Mathematically, this mapping is expressed as:

$$\mathbf{H}: \quad \mathbb{R}^M \to \mathbb{R}^N \tag{18}$$

The mapping is called *linear* because if we have two vectors in $\mathbb{R}^M$, $\mathbf{p}$ and $\mathbf{q}$, and two scalars, α and β, then the following property holds true:

$$\mathbf{H}(\alpha\mathbf{p}+\beta\mathbf{q}) = \alpha\mathbf{H}\mathbf{p}+\beta\mathbf{H}\mathbf{q} \tag{19}$$

B.1.3 Matrix-matrix products

The *matrix-matrix* product $\mathbf{F}=\mathbf{H}\mathbf{G}$ combines two matrices:

$$f_{ij} = \sum_{k=1}^{K} h_{ik} g_{kj} \tag{20}$$

Here, $\mathbf{H}$ is $N\times K$ dimensional, $\mathbf{G}$ is $K\times M$ dimensional, and the resulting product is $N\times M$ dimensional. Note, that the number of columns of $\mathbf{H}$ must equal the number of rows of $\mathbf{G}$.

B.2 Special matrices and properties of matrices

The *null-matrix* $\mathbf{0}$. This is a matrix fully filled with zeros. Matlab function: **`zeros`**. If regarded $\mathbf{0}$ as an operator, then this operator maps each vector to the zero vector: $\mathbf{0}\mathbf{x}=\mathbf{0}$.

A *square matrix* is a matrix with equal number of rows and columns: $N=M$. Such a matrix maps its domain $\mathbb{R}^M$ to itself.

$$\mathbf{H}: \quad \mathbb{R}^M \to \mathbb{R}^M$$

In this case, the mapping is called a *linear operator*.

A *diagonal matrix* is a square matrix fully consisting of zeros except at the diagonal:

$$\mathbf{H} = \begin{bmatrix} h_{11} & & 0 \\ & \ddots & \\ 0 & & h_{NN} \end{bmatrix} = \mathrm{diag}(\begin{bmatrix} h_{11} & \cdots & h_{NN} \end{bmatrix})$$

Matlab function: **`diag`**. Regarded as a mapping, the matrix $\mathbf{H}$ *scales* the elements of the vector $\mathbf{x}$. That is, $y_n = h_{nn}x_n$. Thus, a diagonal matrix *stretches* the space.

The *unit matrix* $\mathbf{I}$. This matrix is a square matrix, fully filled with zero, except for the diagonal elements which are unit:

$$\mathbf{I} = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}$$

Matlab function: **`eye`**. Regarded as an operator, $\mathbf{I}$ maps each vector $\mathbf{x}$ to itself: $\mathbf{I}\mathbf{x}=\mathbf{x}$.

The *transpose operator* swaps the rows and the columns of the matrix. An $N\times M$ matrix $\mathbf{H}$ with elements h_{ij} becomes an $M\times N$ matrix $\mathbf{H}^{\mathrm{T}}$ with elements h_{ji}. Matlab: **`H'`** or **`transpose(H)`**. Properties:

$$(\mathbf{A}\mathbf{B})^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$$

A *symmetric matrix* is a square matrix for which $h_{ij}=h_{ji}$ or $\mathbf{H}=\mathbf{H}^{\mathrm{T}}$.

The *trace* of a square matrix is the sum of its diagonal elements:

$$trace(\mathbf{H}) = \sum_{n=1}^{N} h_{nn} \tag{21}$$

Matlab function: **`trace`**.

The *determinant* $|\mathbf{H}|$ of a square matrix $\mathbf{H}$ is recursively defined with its co-matrices. The co-matrix $\mathbf{H}_{n,m}$ is an $(N-1)\times(N-1)$-matrix that is derived from $\mathbf{H}$ by exclusion of the n-th row and the m-th column. The following equations define the determinant:

$$\begin{aligned} &\text{If} \quad N=1: \quad |\mathbf{H}| = h_{1,1} \\ &\text{If} \quad N>1: \quad |\mathbf{H}| = \sum_{m=1}^{N} (-1)^{m-1} h_{1,m} |\mathbf{H}_{1,m}| \end{aligned} \tag{22}$$

Matlab function: **`det`**. Properties: $|\alpha\mathbf{H}| = \alpha^N |\mathbf{H}|$ and $|\mathbf{H}\mathbf{G}| = |\mathbf{H}||\mathbf{G}|$.

B.3 Matrix inversion

The *inverse of a square matrix* $\mathbf{H}$ is denoted $\mathbf{H}^{-1}$, and is defined as the matrix for which

$$\mathbf{H}\mathbf{H}^{-1} = \mathbf{I} \tag{23}$$

The inverse exists if, and only if, $\det(\mathbf{H}) \neq 0$. The matrix is said to be invertible then. If the inverse does not exist, the matrix is called *singular*.

Consider a set of N linear equations with N unknown variables. For instance:

$$\begin{aligned} y_1 &= h_{11}x_1 + h_{12}x_2 \\ y_2 &= h_{21}x_1 + h_{22}x_2 \end{aligned}$$

in which the variables y_n and coefficients h_{ij} are assumed to be known, and the variables x_n are unknown. The two equations represent two lines in $\mathbb{R}^2$. The intersection point of the two lines is the solution of the equations. If the two lines do not intersect (they are parallel), there is no (unique) solution, and the associated matrix is singular.

The equations are written concisely as $\mathbf{y} = \mathbf{H}\mathbf{x}$. This equation is solved by:

$$\mathbf{x} = \mathbf{H}^{-1}\mathbf{y} \tag{24}$$

If the inverse of the matrix $\mathbf{H}$ exists, then this matrix maps each vector $\mathbf{x}$ from $\mathbb{R}^M$ to a unique vector $\mathbf{y}$, In addition, the operation $\mathbf{H}\mathbf{x}$ can 'hit' then any vector in $\mathbb{R}^M$. That is, for every $\mathbf{y} \in \mathbb{R}^N$, there is a unique corresponding $\mathbf{x}$. In other words, there is a one-to-one relation between $\mathbf{x}$ and $\mathbf{y}$. This is called a *surjection*.

Matlab function for matrix inversion: **`inv`**. Matlab expression for $\mathbf{H}^{-1}\mathbf{y}$ is **`H\y`**.

Some inversion properties:

$$(\mathbf{H}^{-1})^{\mathrm{T}} = (\mathbf{H}^{\mathrm{T}})^{-1}$$

$$(\mathbf{A}\mathbf{B})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$$

$$(\mathbf{A}^{-1} + \mathbf{H}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{H})^{-1} = \mathbf{A} - \mathbf{A}\mathbf{H}^{\mathrm{T}}\left(\mathbf{H}\mathbf{A}\mathbf{H}^{\mathrm{T}} + \mathbf{B}\right)^{-1}\mathbf{H}\mathbf{A}$$

The latter is called the *matrix inversion lemma*.

B.4 Eigenvalues

Sometimes it is very useful to decompose a matrix into other matrices. For instance, singular value decomposition rewrites a given matrix $\mathbf{H}$ into $\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$. The matrices $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{S}$ have nice properties as will be shown in Section B.5.

A widely used tool to get such a decomposition is eigenvalue analysis of a matrix. This concept applies to square matrices only. Suppose $\mathbf{H}$ is an $N \times N$ matrix, then $\mathbf{v}$ is an *eigenvector* of $\mathbf{H}$ if the operation $\mathbf{H}\mathbf{v}$ does not change the direction of $\mathbf{v}$. It only alters its length:

$$\mathbf{H}\mathbf{v} = \lambda\mathbf{v} \tag{25}$$

The scaling factor λ is scalar which is called the *eigenvalue* associated with $\mathbf{v}$. Note that if $\mathbf{v}$ is an eigenvector, then so is $\alpha\mathbf{v}$ with $\alpha \neq 0$. Therefore, we require eigenvectors to have unit length: $\|\mathbf{v}\| = 1$.

Suppose that $\mathbf{H}$ has N linearly independent eigenvectors $\{\mathbf{v}_1, \cdots, \mathbf{v}_N\}$ with N eigenvalues $\{\lambda_1, \cdots, \lambda_N\}$. We put the eigenvectors side by side to form an $N \times N$ matrix:

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_N \end{bmatrix} \tag{26}$$

Next, we form a $N \times N$ diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_N)$. We then have:

$$\mathbf{V}\Lambda = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_N \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_N \end{bmatrix} = \begin{bmatrix} \lambda_1\mathbf{v}_1 & \cdots & \lambda_N\mathbf{v}_N \end{bmatrix} \tag{27}$$

Since $\mathbf{H}\mathbf{V} = \begin{bmatrix} \mathbf{H}\mathbf{v}_1 & \cdots & \mathbf{H}\mathbf{v}_N \end{bmatrix}$, it follows that:

$$\mathbf{H}\mathbf{V} = \mathbf{V}\Lambda \qquad \text{or} \qquad \mathbf{H} = \mathbf{V}\Lambda\mathbf{V}^{-1} \tag{28}$$

Hence, we have decomposed $\mathbf{H}$ in $\mathbf{V}$, Λ, and $\mathbf{V}^{-1}$.

An even more interesting decomposition is achieved if the matrix $\mathbf{H}$ is symmetric: $\mathbf{H} = \mathbf{H}^{\mathrm{T}}$. In that case, it can be proven that:

a) All eigenvalues are real, i.e. imaginary part is zero.
b) Any pair of eigenvectors is orthogonal: $\mathbf{v}_n \perp \mathbf{v}_m$.

The consequence of b), together with $\|\mathbf{v}_n\| = 1$, for the matrix $\mathbf{V}$ is that:

$$\mathbf{V}\mathbf{V}^{\mathrm{T}} = \mathbf{I} \qquad \text{or} \qquad \mathbf{V}^{-1} = \mathbf{V}^{\mathrm{T}} \tag{29}$$

Matrices with this property are said to be *orthonormal.* For these matrices, the following properties hold true:

$$\begin{aligned}(\mathbf{V}\mathbf{y})^{\mathrm{T}}(\mathbf{V}\mathbf{x}) &= \mathbf{y}^{\mathrm{T}}\mathbf{x} \\ \|\mathbf{V}\mathbf{x}\| &= \mathbf{V}\|\mathbf{x}\|\end{aligned} \tag{30}$$

In other words, the operator $\mathbf{V}$ does not change the length of a vector, nor does it change the angles between pairs of vectors. The operator merely *rotates* the whole vector space.

The decomposition of the symmetric matrices becomes:

$$\mathbf{H} = \mathbf{V}\Lambda\mathbf{V}^{\mathrm{T}} \tag{31}$$

The interpretation is as follows. The operation $\mathbf{H}\mathbf{x}$ can be regarded as a sequence of operations:

a) A rotation $\mathbf{V}^{\mathrm{T}}$ acting on the vector $\mathbf{x}$.
b) An elementwise multiplication (stretching) Λ acting on the elements of $\mathbf{V}^{\mathrm{T}}\mathbf{x}$.
c) A back rotation $\mathbf{V}$ acting on the vector $\Lambda\mathbf{V}^{\mathrm{T}}\mathbf{x}$.

The attentive reader sees an analogy with convolution, which is equivalent to a) Fourier transform, b) elementwise multiplication, and c) inverse Fourier transform.

B.5 Singular value decomposition

Singular value decomposition (SVD) is a mathematical theorem that states that any matrix $\mathbf{H}$ can be decomposed into two orthonormal matrices $\mathbf{U}$ and $\mathbf{V}$ and a third matrix $\mathbf{S}$ which is a diagonal matrix:

$$\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^{T} \tag{32}$$

Suppose that $\mathbf{H}$ is a $N \times M$ matrix. Then:

- $\mathbf{U}$ is an orthonormal matrix, i.e. $\mathbf{U}^{T}\mathbf{U} = \mathbf{I}$ with size $N \times N$.
- $\mathbf{V}$ is an orthonormal matrix, i.e. $\mathbf{V}^{T}\mathbf{V} = \mathbf{I}$ with size $M \times M$.
- $\mathbf{S}$ is a diagonal matrix with size $N \times M$. The diagonal elements s_{nn} are called the singular values.

Equation (32) can be written in an alternative form. If we define: $\mathbf{U} = \begin{bmatrix}\mathbf{u}_1 & \cdots & \mathbf{u}_N\end{bmatrix}$ and $\mathbf{V} = \begin{bmatrix}\mathbf{v}_1 & \cdots & \mathbf{v}_M\end{bmatrix}$, where $\mathbf{u}_i$ and $\mathbf{v}_j$ are the columns in the $\mathbf{U}$ and $\mathbf{V}$ matrices, then:

$$\mathbf{H} = \sum_{i=1}^{K} s_{ii}\mathbf{u}_i\mathbf{v}_i^{T} \quad \text{with} \quad K = \min(N, M) \tag{33}$$

Note that some of the singular values might be zero.

Corollary 1

From $\mathbf{H}\mathbf{H}^{T} = \mathbf{U}\mathbf{S}\mathbf{V}^{T}(\mathbf{U}\mathbf{S}\mathbf{V}^{T})^{T} = \mathbf{U}\mathbf{S}\mathbf{V}^{T}\mathbf{V}\mathbf{S}\mathbf{U}^{T} = \mathbf{U}\mathbf{S}^{2}\mathbf{U}^{T}$ it follows that $\mathbf{u}_i$ are the eigenvectors of $\mathbf{H}\mathbf{H}^{T}$ with associated eigenvalues s_{ii}^{2}.

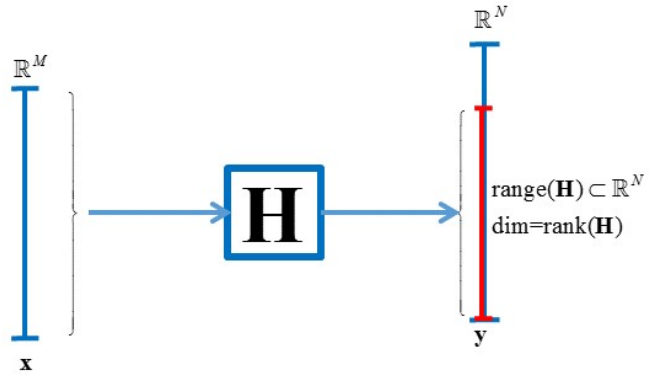
Corollary 2

From $\mathbf{H}^{T}\mathbf{H} = (\mathbf{U}\mathbf{S}\mathbf{V}^{T})^{T}\mathbf{U}\mathbf{S}\mathbf{V}^{T} = \mathbf{V}\mathbf{S}\mathbf{U}^{T}\mathbf{U}\mathbf{S}\mathbf{V}^{T} = \mathbf{V}\mathbf{S}^{2}\mathbf{V}^{T}$ it follows that $\mathbf{v}_i$ are the eigenvectors of $\mathbf{H}^{T}\mathbf{H}$ with associated eigenvalues s_{ii}^{2}.

B.6 The range and the rank of a matrix

Consider a $N \times M$ matrix $\mathbf{H}$. The operation $\mathbf{y} = \mathbf{H}\mathbf{x}$ maps the M dimensional space $\mathbb{R}^{M}$ of $\mathbf{x}$ to the N dimensional space $\mathbb{R}^{N}$ of $\mathbf{y}$.

The *range* of the matrix is the set of all possible outcomes $\mathbf{y}$ when $\mathbf{x}$ is varied over the whole M dimensional space $\mathbb{R}^{M}$. The range is a linear subspace of $\mathbb{R}^{N}$. This is because if $\mathbf{y}_1$ and $\mathbf{y}_2$ are in the range of $\mathbf{H}$, then there exists an $\mathbf{x}_1$ and an $\mathbf{x}_2$ such that $\mathbf{y}_1 = \mathbf{H}\mathbf{x}_1$ and $\mathbf{y}_2 = \mathbf{H}\mathbf{x}_2$. For any linear combination $\alpha\mathbf{x}_1 + \beta\mathbf{x}_2$ we have $\alpha\mathbf{y}_1 + \beta\mathbf{y}_2 = \mathbf{H}(\alpha\mathbf{x}_1 + \beta\mathbf{x}_2)$. Thus, $\alpha\mathbf{y}_1 + \beta\mathbf{y}_2$ is also in the range of $\mathbf{H}$.



The range can be found by considering the singular value decomposition of the matrix $\mathbf{H}$:

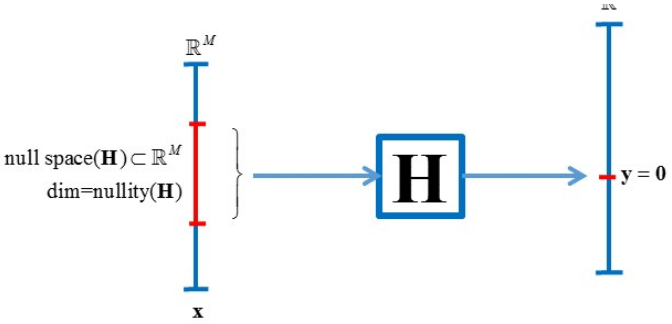
$$\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^{T} == \sum_{i-1}^{K} s_{ii}\mathbf{u}_i\mathbf{v}_i^{T} \quad \text{with} \quad K = \min(N, M) \tag{34}$$

Each $\mathbf{u}_i$ for which the associated singular value s_{ii} is not zero is in the range of $\mathbf{H}$. Since the matrix $\mathbf{U}$ is orthonormal, all these vectors $\mathbf{u}_i$ with non-zero s_{ii} form an orthonormal basis for the range. The dimension of the range equals the number of those vectors. This number is called the *rank* of the matrix, and is often denoted by $\mathrm{rank}(\mathbf{H})$.

B.7 The null space of a matrix

Consider a $N \times M$ matrix $\mathbf{H}$. The operation $\mathbf{y} = \mathbf{H}\mathbf{x}$ maps the M dimensional space $\mathbb{R}^M$ of $\mathbf{x}$ to the N dimensional space $\mathbb{R}^N$ of $\mathbf{y}$.

The *null space* is the set of vectors $\mathbf{x}$ for which $\mathbf{H}\mathbf{x} = \mathbf{0}$. If $\mathbf{x}_1$ and $\mathbf{x}_2$ are vectors from the null space, then so is any linear combination $\alpha\mathbf{x}_1 + \beta\mathbf{x}_2$. Thus, the null space is a linear subspace of $\mathbb{R}^M$.



The null space can be found by considering the singular value decomposition of the matrix $\mathbf{H}$. See above.

$$\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^T == \sum_{i-1}^{K} s_{ii}\mathbf{u}_i\mathbf{v}_i^T \quad \text{with} \quad K = \min(N, M) \tag{35}$$

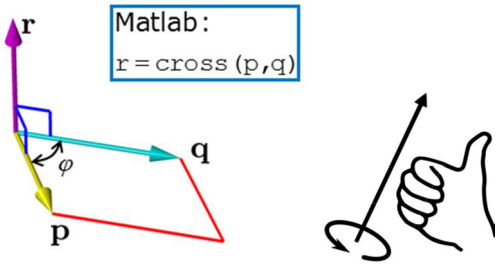
Each $\mathbf{v}_i$ for which the associated singular value s_{ii} is not zero is not lying in the null space. All the other vectors are within the null space. Since the matrix $\mathbf{V}$ is orthonormal, all these vectors $\mathbf{v}_i$ in the null space form an orthonormal basis for the null space, and the dimension of this null space equals the number of those vectors. This dimension is sometimes called the *nullity* of the matrix.

B.8 The cross product

The cross product is only applicable to 3D spaces. In $\mathbb{R}^3$ the cross product of two vectors $\mathbf{p}$ and $\mathbf{q}$ is defined as a third vector $\mathbf{r} = \mathbf{p} \times \mathbf{q}$ with the following properties (see figure below):

$$\begin{aligned} &\|\mathbf{r}\| = \|\mathbf{p}\|\|\mathbf{q}\|\sin\varphi \\ &\mathbf{r} \perp \mathbf{p} \quad \text{and} \quad \mathbf{r} \perp \mathbf{q} \end{aligned} \tag{36}$$

The direction of $\mathbf{r}$ follows the right hand rule. That is, if the fingers of the right hand follows the rotation from $\mathbf{p}$ to $\mathbf{q}$, then $\mathbf{r}$ is in the direction of the thumb.



Note that $\|\mathbf{p}\|\|\mathbf{q}\|\sin\varphi$ is the area of the parallelogram. Consequently, we have:

$$\begin{aligned} &\mathbf{r} = \mathbf{0} && \text{if } \sin\varphi = 0; \text{ that is if } \mathbf{p} = \alpha\mathbf{q} \\ &\|\mathbf{r}\| = \|\mathbf{p}\|\|\mathbf{q}\| && \text{if } \mathbf{p} \perp \mathbf{q} \end{aligned} \tag{37}$$

The cross product is calculated as follows:

$$\mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} p_y q_z - p_z q_y \\ -p_x q_z + p_z q_x \\ p_x q_y - p_y q_x \end{bmatrix} \tag{38}$$

The cross product can also be regarded as a matrix-vector multiplication. The vector $\mathbf{p}$ is then cast into a 3×3 skew symmetric (antisymmetric) matrix, as follows:

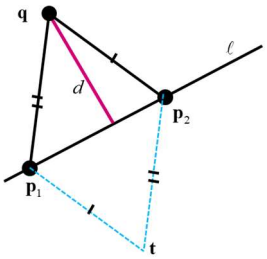
$$[\mathbf{p}]_\times \overset{def}{=} \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix} \tag{39}$$

so that the cross product can be written as a matrix-vector multiplication:

$$\mathbf{p} \times \mathbf{q} = [\mathbf{p}]_\times \mathbf{q} \tag{40}$$

Example: distance between a point and a line

The distance between a point $\mathbf{q}$ and a line ℓ defined by two points $\mathbf{p}_1$ and $\mathbf{p}_2$ is easily obtained using the cross product. For that purpose, consider the geometry in the figure below:



We ask for the shortest distance d. This distance follow readily from considering the parallelogram that is spanned

by $\mathbf{p}_1 - \mathbf{q}$ and $\mathbf{p}_2 - \mathbf{q}$. The area of this parallelogram follows from the cross product:

$$area = \left\| (\mathbf{p}_1 - \mathbf{q}) \times (\mathbf{p}_2 - \mathbf{q}) \right\|$$

However, the same area also equals the base and the height of the parallelogram:

$$area = d \left\| \mathbf{p}_2 - \mathbf{p}_1 \right\|$$

The distance equals:

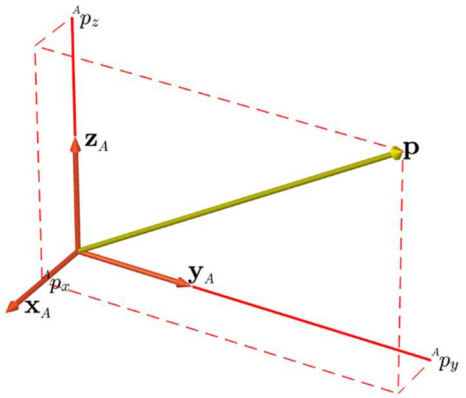
$$d = \frac{\left\| (\mathbf{p}_1 - \mathbf{q}) \times (\mathbf{p}_1 - \mathbf{q}) \right\|}{\left\| \mathbf{p}_2 - \mathbf{p}_1 \right\|} \tag{41}$$

C. 3D pose

We consider the position and angular position of rigid objects in 3D spaces. The angular position is often called the *orientation* or the *attitude*.

C.1 A position of a point in 3D

Given a coordinate system A that consists of three orthogonal basis vectors $(\mathbf{x}_A, \mathbf{y}_A, \mathbf{z}_A)$. They are orthogonal, e.g. $\mathbf{x}_A \perp \mathbf{x}_B$, and have unit length, e.g. $\left\| \mathbf{x}_A \right\| = 1$. These basis vectors span a 3D space. See figure.



A point $\mathbf{p}$ is represented in A as a vector:

$$\mathbf{p} = {}^A p_x \mathbf{x}_A + {}^A p_y \mathbf{y}_A + {}^A p_z \mathbf{z}_A$$

Such a representation is conveniently denoted by its coordinates, stacked into a 3D vector in $\mathbb{R}^3$:

$${}^A\mathbf{p} = \begin{bmatrix} {}^A p_x \\ {}^A p_y \\ {}^A p_z \end{bmatrix}$$

Note that ${}^A\mathbf{p}$ is just a possible representation of the point $\mathbf{p}$. The same point can equally well be represented in some other coordinate system B, such that ${}^B\mathbf{p}$ refers to the same point $\mathbf{p}$.

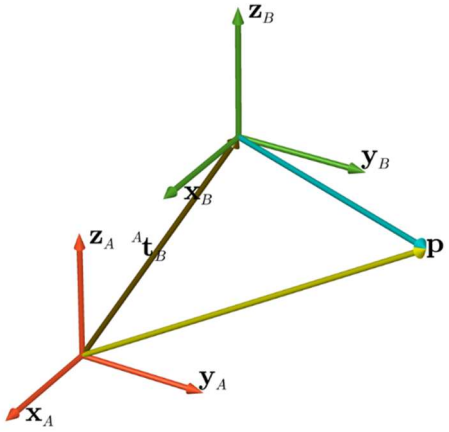
C.2 Two coordinate systems

We consider two coordinate systems and describe the relations between the two.

C.2.1 Translated coordinate systems

Given two coordinate systems A and B. We assume that one is a translated version of the other. That is, B is

shifted with respect to A, but it is not rotated. See figure below.



The origin of B is at a position $\mathbf{t}_B$ that is represented in A by ${}^A\mathbf{t}_B = \begin{bmatrix} {}^A t_x & {}^A t_y & {}^A t_z \end{bmatrix}^{\mathrm{T}}$. Likewise, ${}^B\mathbf{t}_A$ is the position of the origin of A expressed in B. With the logic of this notation, ${}^B\mathbf{t}_B = \mathbf{0}$ and ${}^A\mathbf{t}_A = \mathbf{0}$.

The translation of B with respect to A is opposite to the translation of A with respect to B, so that we have:

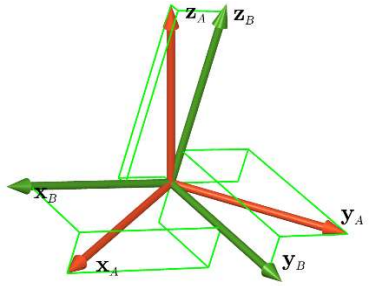
$$ {}^B\mathbf{t}_A = -{}^A\mathbf{t}_B $$

A point $\mathbf{p}$ can be represented both in A and in B. These representations are related to each other according to:

$$ {}^A\mathbf{p} = {}^B\mathbf{p} - {}^B\mathbf{t}_A $$
$$ {}^B\mathbf{p} = {}^A\mathbf{p} - {}^A\mathbf{t}_B $$

C.2.2 Rotated coordinate systems

We consider two coordinate systems that are rotated versions of each other. See figure.



The system A is described in terms of system B by defining the basis vectors of A expressed in those of B:

$$ \mathbf{x}_A = R_{xx}\mathbf{x}_B + R_{xy}\mathbf{y}_B + R_{xz}\mathbf{z}_B $$
$$ \mathbf{y}_A = R_{yx}\mathbf{x}_B + R_{yy}\mathbf{y}_B + R_{yz}\mathbf{z}_B $$
$$ \mathbf{z}_A = R_{zx}\mathbf{x}_B + R_{zy}\mathbf{y}_B + R_{zz}\mathbf{z}_B $$

This is conveniently expressed in matrix-vector notation:

$$ \begin{bmatrix} \mathbf{x}_A \\ \mathbf{y}_A \\ \mathbf{z}_A \end{bmatrix} = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{bmatrix} \begin{bmatrix} \mathbf{x}_B \\ \mathbf{y}_B \\ \mathbf{z}_B \end{bmatrix} \quad \text{or:} \quad \begin{bmatrix} \mathbf{x}_A \\ \mathbf{y}_A \\ \mathbf{z}_A \end{bmatrix} = {}^A\mathbf{R}_B \begin{bmatrix} \mathbf{x}_B \\ \mathbf{y}_B \\ \mathbf{z}_B \end{bmatrix} $$

The matrix:

$$ {}^A\mathbf{R}_B = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{bmatrix} $$

is called the *rotation matrix*.

For $\mathbf{x}_A$ we find:

$$ \mathbf{x}_A = R_{xx}\mathbf{x}_B + R_{xy}\mathbf{y}_B + R_{xz}\mathbf{z}_B \quad \text{thus} \quad {}^B\mathbf{x}_A = \begin{bmatrix} R_{xx} \\ R_{xy} \\ R_{xz} \end{bmatrix} $$

Applying this to $\mathbf{y}_A$ $\mathbf{z}_A$ as well, we conclude:

$$ {}^A\mathbf{R}_B = \begin{bmatrix} {}^B\mathbf{x}_A & {}^B\mathbf{y}_A & {}^B\mathbf{z}_A \end{bmatrix}^{\mathrm{T}} \tag{42} $$

Note that, for instance, $R_{xx}\mathbf{x}_B$ is the projection of $\mathbf{x}_A$ on $\mathbf{x}_B$. Therefore,

$$ R_{xx} = \frac{(\mathbf{x}_A, \mathbf{x}_B)}{\|\mathbf{x}_B\|^2} = \cos\varphi_{xx} $$

where φ_{xx} is the angle between $\mathbf{x}_A$ and $\mathbf{x}_B$. Likewise, $R_{xy} = \cos\varphi_{xy}$ with φ_{xy} the angle between $\mathbf{x}_A$ and $\mathbf{y}_B$, and so on. Apparently:

$$ {}^A\mathbf{R}_B = \begin{bmatrix} \cos(\varphi_{xx}) & \cos(\varphi_{xy}) & \cos(\varphi_{xz}) \\ \cos(\varphi_{yx}) & \cos(\varphi_{yy}) & \cos(\varphi_{yz}) \\ \cos(\varphi_{zx}) & \cos(\varphi_{zy}) & \cos(\varphi_{zz}) \end{bmatrix} \tag{43} $$

For this reason, the rotation matrix is also called the *direct cosine matrix*.

To express the basis vectors of B in A we define the matrix ${}^B\mathbf{R}_A$. Inspection of eq (43) reveals that ${}^B\mathbf{R}_A = {}^A\mathbf{R}_B^{\mathrm{T}}$. Furthermore, we have:

$$ \begin{bmatrix} \mathbf{x}_A \\ \mathbf{y}_A \\ \mathbf{z}_A \end{bmatrix} = {}^A\mathbf{R}_B \begin{bmatrix} \mathbf{x}_B \\ \mathbf{y}_B \\ \mathbf{z}_B \end{bmatrix} = {}^A\mathbf{R}_B\,{}^B\mathbf{R}_A \begin{bmatrix} \mathbf{x}_A \\ \mathbf{y}_A \\ \mathbf{z}_A \end{bmatrix} $$

Considering the properties of (see Section B.4), the conclusion is:

$$ {}^{A}\mathbf{R}_{B}^{-1} = {}^{A}\mathbf{R}_{B}^{\mathrm{T}} = {}^{B}\mathbf{R}_{A} \tag{44} $$

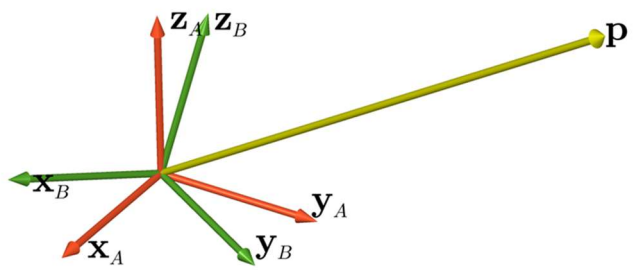
Corollary 3

Rotation matrices have 3 degrees of freedom. This latter follows from the following construction:

1. Consider the first column $\begin{bmatrix} R_{xx} & R_{xy} & R_{xz} \end{bmatrix}^{\mathrm{T}}$ of a rotation matrix. We can choose two elements freely, but within the constraint of unit length $R_{xx}^2 + R_{xy}^2 + R_{xz}^2 = 1$. The third element follows from this constraint.
2. Consider the second column $\begin{bmatrix} R_{yx} & R_{yy} & R_{yz} \end{bmatrix}^{\mathrm{T}}$. We can choose one element freely, but within the constraint of unit length $R_{yx}^2 + R_{yy}^2 + R_{yz}^2 = 1$. The other two elements follow from this constraint, but also from the orthogonality constraint $R_{xx}R_{yx} + R_{xy}R_{yy} + R_{xz}R_{yz} = 0$.
3. The third column follows directly from the unit length constraint $R_{zx}^2 + R_{zy}^2 + R_{zz}^2 = 1$, and the two orthogonality constraints: $R_{xx}R_{zx} + R_{xy}R_{zy} + R_{xz}R_{zz} = 0$ and $R_{yx}R_{zx} + R_{yy}R_{zy} + R_{yz}R_{zz} = 0$.

C.2.3 Point representation in two rotated systems

Consider a point $\mathbf{p}$ that is represented in two rotated coordinate systems A and B



The point $\mathbf{p}$ has two representation:

$$ \begin{aligned} \mathbf{p} &= {}^{A}p_x\mathbf{x}_A + {}^{A}p_y\mathbf{y}_A + {}^{A}p_z\mathbf{z}_A \\ &= {}^{B}p_x\mathbf{x}_B + {}^{B}p_y\mathbf{y}_B + {}^{B}p_z\mathbf{z}_B \end{aligned} $$

which can be written as:

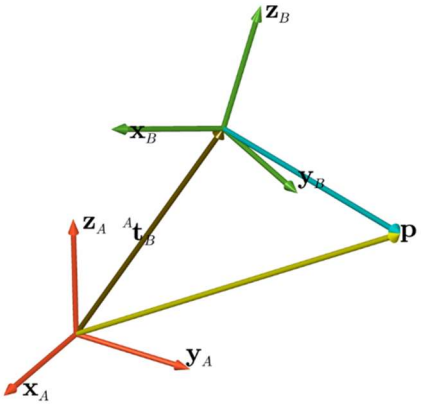
$$ \mathbf{p} = {}^{A}\mathbf{p}^{\mathrm{T}} \begin{bmatrix} \mathbf{x}_A \\ \mathbf{y}_A \\ \mathbf{z}_A \end{bmatrix} = {}^{A}\mathbf{p}^{\mathrm{T}}\,{}^{A}\mathbf{R}_{B} \begin{bmatrix} \mathbf{x}_B \\ \mathbf{y}_B \\ \mathbf{z}_B \end{bmatrix} = \left({}^{B}\mathbf{R}_{A}\,{}^{A}\mathbf{p}\right)^{\mathrm{T}} \begin{bmatrix} \mathbf{x}_B \\ \mathbf{y}_B \\ \mathbf{z}_B \end{bmatrix} = {}^{B}\mathbf{p}^{\mathrm{T}} \begin{bmatrix} \mathbf{x}_B \\ \mathbf{y}_B \\ \mathbf{z}_B \end{bmatrix} $$

From which follows:

$$ \begin{aligned} {}^{A}\mathbf{p} &= {}^{A}\mathbf{R}_{B}\,{}^{B}\mathbf{p} \\ {}^{B}\mathbf{p} &= {}^{B}\mathbf{R}_{A}\,{}^{A}\mathbf{p} \end{aligned} \tag{45} $$

C.2.4 Rotated and translated frames

Lastly, we consider two coordinate systems that are rotated and translated versions of each other. Such coordinate systems are called *frames*. We have frame A which we consider as the *reference frame*. We have frame B that is defined with respect to A by means of $\left\{ {}^{A}\mathbf{t}_{B}, {}^{A}\mathbf{R}_{B} \right\}$:



A point $\mathbf{p}$ is represented in B by ${}^{B}\mathbf{p}$. The same point is represented in A by:

$$ {}^{A}\mathbf{p} = {}^{A}\mathbf{R}_{B}\,{}^{B}\mathbf{p} + {}^{A}\mathbf{t}_{B} \tag{46} $$

Note, that the notation works out intuitively. The subscript ${}_{B}$ and the superscript ^{B} are situated next to each other ${}_{B}^{B}$ indicating that the variables ${}^{A}\mathbf{R}_{B}$ and ${}^{B}\mathbf{p}$ "fit". The two scripts neutralize each other, The final result on the right side is an expression in A, which is consistent with the left side.

Going back, from a representation in A to one in B is likewise: ${}^{B}\mathbf{p} = {}^{B}\mathbf{R}_{A}\,{}^{A}\mathbf{p} + {}^{B}\mathbf{t}_{A}$. Substitution of (46) yields ${}^{B}\mathbf{p} = {}^{B}\mathbf{R}_{A}\left({}^{A}\mathbf{R}_{B}\,{}^{B}\mathbf{p} + {}^{A}\mathbf{t}_{B}\right) + {}^{B}\mathbf{t}_{A}$, from which:

$$ \begin{aligned} {}^{B}\mathbf{t}_{A} &= -{}^{B}\mathbf{R}_{A}\,{}^{A}\mathbf{t}_{B} \\ {}^{A}\mathbf{t}_{B} &= -{}^{A}\mathbf{R}_{B}\,{}^{B}\mathbf{t}_{A} \end{aligned} \tag{47} $$

This brings an alternative expression for (46):

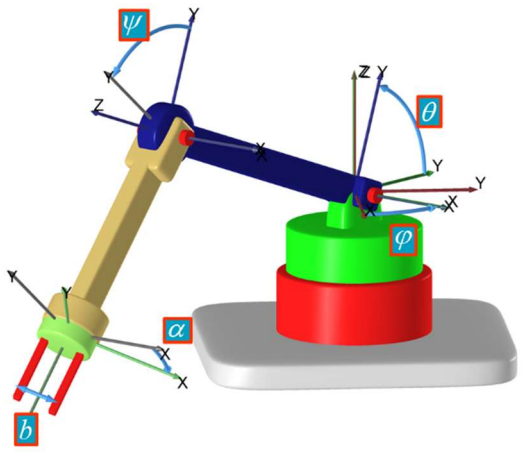
$$ {}^{A}\mathbf{p} = {}^{A}\mathbf{R}_{B}\left({}^{B}\mathbf{p} - {}^{B}\mathbf{t}_{A}\right) \tag{48} $$

Here too, the notation is consistent: subtraction of vectors, i.e. translation, is only meaningful if the vectors are represented in the same frame.

C.3 Homogeneous transformations

In stereo vision, two frames may suffice to describe the geometry, but in other computer vision applications, more frames are needed. This occurs, for example, in *multi-view geometry* and also the processing of an image sequence from a moving camera. The situation also occurs in surgical navigation, and in robotics,

To describe, for instance, the kinematics of the robotic device, shown below, each rigid part of the construction has its own frame.



To describe the frame $\{{}^5\mathbf{R}_1, {}^5\mathbf{t}_1\}$ attached to the red fingers relative to frame attached to the red robotic base, we have to go through a sequence of five frames $1,\cdots,5$ attached to the different parts. Each transformation is described by a parameter that represents the state of the corresponding actuator, i.e. the angles φ, θ, ψ, α and the shift b.

It is cumbersome to apply eq (46) for calculating the representation ${}^5\mathbf{p}$ of a point given its representation ${}^1\mathbf{p}$ in the robotic reference frame:

$$\begin{aligned}{}^5\mathbf{p} &= {}^5\mathbf{R}_4\,{}^4\mathbf{p} + {}^5\mathbf{t}_4 \\ &= {}^5\mathbf{R}_4\left({}^4\mathbf{R}_3\,{}^3\mathbf{p} + {}^4\mathbf{t}_3\right) + {}^5\mathbf{t}_4 \\ &= {}^5\mathbf{R}_4\left({}^4\mathbf{R}_3(\cdots) + {}^4\mathbf{t}_3\right) + {}^5\mathbf{t}_4 \qquad \text{and so on}\end{aligned}$$

Things become even more complicated if we want to inverse the relation, i.e. finding ${}^1\mathbf{p}$ given ${}^5\mathbf{p}$.

Homogeneous coordinates are invented to by-pass this complexity. The homogeneous representation of a 3D point ${}^1\mathbf{p}$ is a 4D vector ${}^1\underline{\mathbf{p}}$:

$${}^1\underline{\mathbf{p}} = \begin{bmatrix} {}^1p_x \\ {}^1p_y \\ {}^1p_z \\ 1 \end{bmatrix} = \begin{bmatrix} {}^1\mathbf{p} \\ 1 \end{bmatrix} \tag{49}$$

Definition of the homogeneous transformation matrix:

$${}^2\mathbf{T}_1 = \left[\begin{array}{ccc|c} & {}^2\mathbf{R}_1 & & {}^2\mathbf{t}_1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{50}$$

converts the *inhomogeneous* equation (46) into a *homogeneous* equation:

$${}^2\underline{\mathbf{p}} = {}^2\mathbf{T}_1\,{}^1\underline{\mathbf{p}} \tag{51}$$

Indeed, substitution of (49) and (51) in (51) shows that:

$$\begin{bmatrix} {}^2\mathbf{p} \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & {}^2\mathbf{R}_1 & & {}^2\mathbf{t}_1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^1\mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^2\mathbf{R}_1\,{}^1\mathbf{p} + {}^2\mathbf{t}_1 \\ 1 \end{bmatrix}$$

The following two properties make the homogenous representations very useful:

$$\begin{aligned} \text{cascading:} \quad & {}^3\mathbf{T}_1 = {}^3\mathbf{T}_2\,{}^2\mathbf{T}_1 \\ \text{inversion:} \quad & {}^2\mathbf{T}_1 = {}^1\mathbf{T}_2^{-1} \end{aligned} \tag{52}$$

In the robotic example, the forward transformation becomes simply: ${}^5\underline{\mathbf{p}} = {}^5\mathbf{T}_4\,{}^4\mathbf{T}_3\,{}^3\mathbf{T}_2\,{}^2\mathbf{T}_1\,{}^1\underline{\mathbf{p}}$. The backward transformation is: ${}^1\underline{\mathbf{p}} = \left({}^5\mathbf{T}_4\,{}^4\mathbf{T}_3\,{}^3\mathbf{T}_2\,{}^2\mathbf{T}_1\right)^{-1}\,{}^5\underline{\mathbf{p}}$.

C.4 Representation of pose

3D objects not only have a position, but also an orientation. Position and orientation, taken together, is the *pose* of an object. To define the pose, first a frame must be defined that is attached to some point of the object. The pose of that object is given then by the position and orientation of that frame with respect to some reference frame. The representation of pose is either explicitly by the rotation matrix and origin position: $\{{}^{ref}\mathbf{R}_{obj}, {}^{ref}\mathbf{t}_{obj}\}$, or more conveniently by a transformation matrix: ${}^{ref}\mathbf{T}_{obj}$

C.5 Representations and functions in Matlab

Representation of points in Matlab

Matrix-vector products are found in two forms:

pre-mulitiplication: $\mathbf{y} = \mathbf{A}\mathbf{x}$ This applies to column vectors

post-multiplication: $\mathbf{y} = \mathbf{x}\mathbf{B}$ This applies to row vectors

The two forms are easily converted into each other:

$$\text{if } \mathbf{y} = \mathbf{A}\mathbf{x} \text{ then: } \quad \mathbf{y}^{\mathrm{T}} = \mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$$

Expressions can be given in one or the other form. In most textbooks, column vectors are used. Consequently all expressions appear in the pre-multiplication form. The problem with Matlab is that it is not consistent with this. In many toolboxes, row vectors with post-multiplication are assumed. This is sometimes indicated by the 'transposed form' as opposed to the 'non-transposed form'.

If a series of points are to be represented, Matlab often assumes row vectors. A set of N points are represented by a $N\times 3$ or $N\times 4$ (homogeneous) array. For instance, two points $\mathbf{p}_1 = \begin{bmatrix} p_{11} & p_{12} & p_{13} & 1 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{p}_2 = \begin{bmatrix} p_{21} & p_{22} & p_{23} & 1 \end{bmatrix}^{\mathrm{T}}$ are stored in a 2×4 array:

```
P = [p11 p12 p13 1;
     p21 p22 p23 1];
```

If these points are to be tansformed by a 4×4 transformation matrix $\mathbf{T}$, e.g. $\mathbf{T}\mathbf{p}_1$, Matlab applies post-multiplication `P*T`. The consequence is that the matlab matrix is the transposed version of $\mathbf{T}$. That is: `T`$=\mathbf{T}^{\mathrm{T}}$.

Useful Functions in Matlab

- **`cart2hom`** and **`hom2cart`** – Conversion between Cartesian coordinates and homogeneous coordinates. These functions are from the Robotics System Toolbox. The functions assume row vector representation of points.
- **`makehgtform`** – Create 4-by-4 transform matrix. This function is part of the set standard graphics functions of Matlab. The resulting matrix is in column vector representation, i.e non-transposed form.

D. Four representations of orientation and rotation

D.1 Rotation matrices

In section C.2.2, the rotation matrix ${}^{A}\mathbf{R}_{B}$, also called DCM, was introduced to specify the orientation of frame B relative to frame A. The interpretation is simple. Consider how the basis vectors

$$\mathbf{x}_B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{y}_B = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{z}_A = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{53}$$

of frame B are represented in frame A:

$${}^{A}\mathbf{R}_{B}\begin{bmatrix} \mathbf{x}_B & \mathbf{y}_B & \mathbf{z}_B \end{bmatrix} = {}^{A}\mathbf{R}_{B} = \begin{bmatrix} {}^{A}\mathbf{r}_1 & {}^{A}\mathbf{r}_2 & {}^{A}\mathbf{r}_3 \end{bmatrix} \tag{54}$$

Thus, the columns of ${}^{A}\mathbf{R}_{B}$ are the basis vectors of frame B, expressed in coordinates of frame A. Likewise, the columns of ${}^{B}\mathbf{R}_{A}$ are the basis vectors of frame A, expressed in coordinates of frame B.

A rotation matrix can also be regarded as a rotation. ${}^{B}\mathbf{R}_{A}$ is the rotation applied to frame A to bring it to an orientation of frame B.

Adding a second rotation applied to frame B such that it is brought to a new frame C with orientation ${}^{A}\mathbf{R}_{C}$ is defined by a rotation matrix ${}^{B}\mathbf{R}_{C}$ (expressed in frame B). The addition of this new rotation to frame B is reflected in the rotation matrices by means of a matrix multiplication:

$${}^{A}\mathbf{R}_{C} = {}^{A}\mathbf{R}_{B}\,{}^{B}\mathbf{R}_{C} \tag{55}$$

See eq (52). It is clear that adding a zero rotation is the same as multiplication by a unit rotation matrix ${}^{B}\mathbf{R}_{C} = \mathbf{I}$.

D.1.1 Pros and cons

Rotation matrices have a nice property: the representation is unique. For a given rotation, there is only one associated rotation matrix. For a given rotation matrix, there is only one associated rotation.

Yet, for navigation, rotation matrices are not always most appropriate. The matrices are redundant: 9 parameters to represent a quantity with 3 degrees of freedom. In other words, after each manipulation of a rotation matrix, we have to assure that in the resulting rotation matrix, 6 constraints are satisfied. This can be cumbersome and

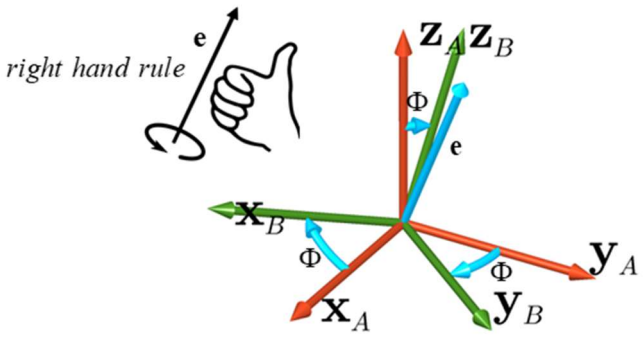
unstable. For instance, an application could require an interpolation from ${}^{A}\mathbf{R}_{B}$ to ${}^{A}\mathbf{R}_{C}$ consisting of a sequence of small rotations of equal size to obtain a smooth rotational movement. This is difficult to achieve with rotation matrices.

Another disadvantage of the rotation matrix occurs if we want to know whether the orientation of frame B is almost the same as the one of frame A. Is frame B almost aligned with frame A? To answer this, the angle between frame A and B must be quantified. This angle is not directly available, but follows from the trace of the rotation matrix as will be shown in eq (58) in section D.2.

D.2 Axis-angle representation

Euler's principal rotation theorem:

> *Each frame can brought from an arbitrary initial orientation to another final orientation by means of a single rotation about an axis. The representation of this axis is the same in both the initial and final frame.*



The figure above shows frame A and frame B. There is a rotation axis $\mathbf{e}$ which has equal representations in the two frames: ${}^{A}\mathbf{e} = {}^{B}\mathbf{e}$. Since only the direction of the axis matters, we require unit length: $\left\| {}^{A}\mathbf{e} \right\| = 1$. The angle of rotation is denoted by Φ. Its sign is defined using the right hand rule. Its absolute value $|\Phi|$ is the magnitude of rotation. A small value means that A and B are almost aligned.

D.2.1 Non-uniqueness

Each additional multiple of 360^0 to the angle Φ does not change the orientation of frame B. It only changes the amount of rotation to get there, starting from frame A. Thus, Φ can be replaced by $\Phi + 2\pi N$ with N any integer. The same orientation can also be reached by rotating the other way round. That is, Φ can also be replaced by $\Phi - 2\pi$. However, the two rotations Φ and $\Phi - 2\pi$ differ. In the current example, Φ is the short way to rotate, whereas $\Phi - 2\pi$ is the long way.

The rotation axis is not unique either. If the opposite direction $-\mathbf{e}$ was chosen, the same rotations can be described by opposite angles: $-\Phi$ and $2\pi - \Phi$. Hence, there are four different principal rotations that all describe two different rotations, but that end up in just one orientation. Note also that if the rotation is zero, the axis $\mathbf{e}$ is undefined.

D.2.2 Conversion to and from rotation matrix

The rotation does not effect the rotation axis:

$$ {}^{A}\mathbf{e} = {}^{A}\mathbf{R}_{B}\,{}^{B}\mathbf{e} = {}^{B}\mathbf{e} \tag{56} $$

From this, it follows that ${}^{B}\mathbf{e}$ is the eigenvector of the matrix ${}^{A}\mathbf{R}_{B}$ with eigenvalue 1. This could be used to convert a rotation matrix into an axis-angle representation. An easier way to find the conversion between rotation matrix and axis-angle representation is the *Rodrigues' rotation formula*. Suppose that ${}^{B}\Phi_{A}$ is the angle with which frame A is rotated around the axis $\mathbf{e}$ to get frame B. Rodriques showed that if $\mathbf{e} = \begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix}^{\mathrm{T}}$ is the rotation axis, then:

$$ {}^{B}\mathbf{R}_{A} = \mathbf{I} - \sin\left({}^{B}\Phi_{A}\right)\mathbf{E} + \left(1 - \cos\left({}^{B}\Phi_{A}\right)\right)\mathbf{E}^2 $$
$$ \text{with } \mathbf{E} = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \tag{57} $$

By expanding this equation, the inverse conversion is found. Suppose that the elements of ${}^{B}\mathbf{R}_{A}$ are denoted by r_{nm}, then:

$$ {}^{B}\Phi_{A} = \arccos\tfrac{1}{2}\left(r_{11} + r_{22} + r_{33} - 1\right) $$
$$ {}^{A}\mathbf{e} = {}^{B}\mathbf{e} = \frac{1}{2\sin\left({}^{B}\Phi_{A}\right)}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{58} $$

D.2.3 Pros and Cons

The advantage of the axis/angle representation is twofold. First, it has a simple interpretation. Second, the magnitude of rotations is easily expressed as just $|\Phi|$. Therefore, a smooth interpolation between two different orientations is easily accomplished.

A disadvantage is the already mentioned four-fold ambiguities. A second disadvantage is the complexity to

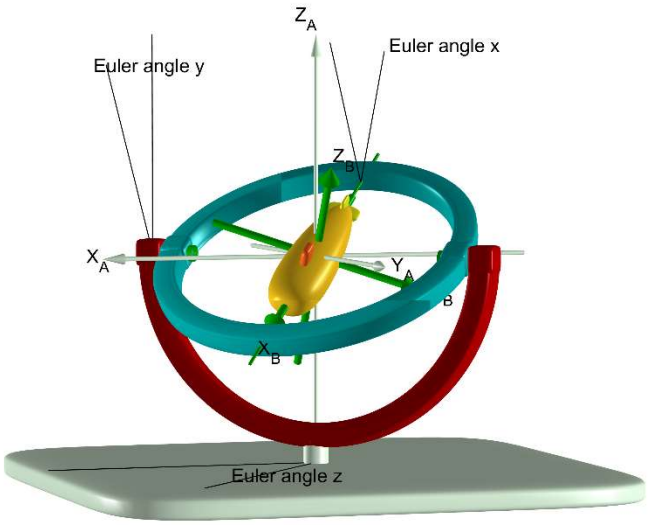
concatenate two consecutive rotations with different rotational axes. To do so, first the representations must be converted to rotation matrices, so that eq (55) applies, and then it has to be back transformed to axis-angle.

D.3 Euler angles

Euler angles are the most commonly used representation. The orientation is described by three sequential rotations about the axes of the coordinate system. The order in this sequence is important. Suppose that the three axes are denoted by i, j, and k. The i-j-k Euler angles means that we rotate first about the i-th axis, then about the j-th axis, and finally about the k-th axis. As an example, (3-2-1), or 'zyx' means:

- First rotating about the z-axis. This creates a new frame in which the z-is unaltered.
- Then rotating about the (new) y-axis, creating a second frame.
- Finally, rotating about the (newest) x-axis. This creates the final frame.

This convention is mostly used in navigation applications. n astronomy the 'zxz' convention is popular to describe orbit planes.



In full, 12 conventions exist, of which 6 are symmetric, e.g. 'zxz', 'zyz' and 6 are asymmetric: 'zyx', 'yzx', and so on. Whatever convention is chosen, there is always a so-called singularity. This always occurs for typical values of the second rotation angle, in which the values of the first and third angles are not unique[2]. For asymmetric conventions, this happens when the angle +90⁰ and -90⁰. For symmetric conventions the singularities are at 0⁰ and 180⁰.

The 'zxy' convention is illustrated below. The three different angles have the following names:

- Euler angle around z axis: *yaw*.
- Euler angle around y axis: *pitch*.
- Euler angle around x axis: *roll*.

D.3.1 Conversion to and from rotation matrix

The rotation matrices that are associated with rotations around the basic vectors, called principal rotations, are:

$$\mathbf{R}_{x,\varphi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix}$$
$$\mathbf{R}_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \qquad (59)$$
$$\mathbf{R}_{z,\psi} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

With these principal rotations, each set of Euler angles can be mapped to the corresponding rotation matrix. For instance, the rotation matrix associated with the Euler angles with 'zyx' convention are:

$${}^{B}\mathbf{R}_{A}(\psi,\theta,\varphi) = \mathbf{R}_{x,\varphi}\mathbf{R}_{y,\theta}\mathbf{R}_{z,\psi} \qquad (60)$$

Substitution of (59) in (60) shows how the rotation matrix is calculated from the Euler angles:

$${}^{B}\mathbf{R}_{A}(\psi,\theta,\varphi) = \begin{bmatrix} c_\psi c_\theta & c_\theta s_\psi & -s_\theta \\ c_\psi s_\theta s_\varphi - c_\varphi s_\psi & c_\psi c_\varphi + s_\psi s_\theta s_\varphi & c_\theta s_\varphi \\ s_\psi s_\varphi + c_\psi c_\varphi s_\theta & c_\varphi s_\psi s_\theta - c_\psi s_\varphi & c_\theta c_\varphi \end{bmatrix} \qquad (61)$$

For brevity, the shortcuts $c_\varphi = \cos\varphi$ and $s_\varphi = \sin\varphi$, and so on, have been made. The inverse conversion follows directly from (61):

$$\psi = \operatorname{atan}\left(\frac{r_{12}}{r_{11}}\right) \qquad \theta = \operatorname{asin}(-r_{13}) \qquad \varphi = \operatorname{atan}\left(\frac{r_{23}}{r_{33}}\right) \qquad (62)$$

[2] Compare this, for instance, with the latitudinal and longitudinal earth coordinates. At a pole, (latitude = 90⁰), the longitudes are not defined.

In the atan function, the quadrants must be checked. If, for instance, r_{12} and r_{11} are both negative, 180⁰ should be added to ψ.

D.3.2 Pros and Cons

Euler angles are easy to interpret, but there are some serious disadvantages:

- The singularities which occur in the second rotation. This is often called the *gimbal lock*.
- It is difficult to describe a sequence of consecutive rotations by means of Euler angles. One could convert the sets of Euler angles to rotation matrices by using (60). Then to multiply the found matrices, and finally convert the rotation matrix back to Euler angles using (62).
- The magnitude of the overall rotation cannot easily be deduced directly by Euler angles. If all three angles are small, then the difference in orientations of A and B is also small. But a small difference in orientation does not guarantee that the Euler angles will be small.

D.4 Quaternions

A complex number with unit length, e.g. $\mathbf{z} = \cos\phi + \mathbf{j}\sin\phi$ can be used to represent orientations in 2D. A possible advantage of doing so is when describing two consecutive rotations $\mathbf{z}_1 = \cos\phi_1 + \mathbf{j}\sin\phi_1$ and $\mathbf{z}_2 = \cos\phi_2 + \mathbf{j}\sin\phi_2$. The result is just $\mathbf{z}_1\mathbf{z}_2$.

Quaternions is a number system that generalizes complex numbers from 2D to a 4D system. The basis of a quaternion consists of a scalar (the real part in a complex number), and 3 so-called quaternion units: $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$. A quaternion is a quadruple of real numbers that are combined as follows:

$$\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \tag{63}$$

q_0 is the *scalar* part. The triple $[q_1, q_2, q_3]$ is the vectorial part[3].

The complex number system has been set up such that a consistent framework for addition, subtraction, multiplication and division is established. This has been obtained from the axiom that $\mathbf{j}^2 = -1$. This results in the following multiplication table:

	1	**j**
1	1	**j**
j	**j**	−1

In 1843, the mathematician Hamilton invented a consistent framework for quaternions. It is based on the axioms:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \tag{64}$$

This leads to the following multiplication table:

	1	**i**	**j**	**k**
1	1	**i**	**j**	**k**
i	**i**	−1	**k**	−**j**
j	**j**	−**k**	−1	**i**
k	**k**	**j**	−**i**	−1

This system is not commutative, e.g. $\mathbf{ij} \neq \mathbf{ji}$.

Quaternions represent the orientation and rotation by coding the axis-angle representation. Suppose that $\mathbf{e} = [e_1 \quad e_2 \quad e_3]^T$ is the principal rotation axis, and Φ is the rotation angle, then the associated quaternion is defined as:

$$\begin{aligned} q_0 &= \cos\tfrac{1}{2}\Phi \\ q_1 &= e_1 \sin\tfrac{1}{2}\Phi \\ q_2 &= e_2 \sin\tfrac{1}{2}\Phi \\ q_3 &= e_3 \sin\tfrac{1}{2}\Phi \end{aligned} \tag{65}$$

Since quaternions have 4 parameters, there must be one constraint. This constraint follows from the requirement that the rotation axis has unit length: $\|\mathbf{e}\|^2 = e_1^2 + e_2^2 + e_3^2 = 1$. Only its direction matters. In general $\cos^2\alpha + \sin^2\alpha = 1$. Thus, the quaternions must also have unit length:

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{66}$$

The requirement that $e_1^2 + e_2^2 + e_3^2 = 1$ means that the vector $\mathbf{e}$ must be situated on the surface of sphere with unit radius. Equation (66) states that the quaternions must be situated on the 3D surface of a 4D hypersphere with unit radius.

Interpretation:
Since $q_0 = \cos\frac{1}{2}\Phi$, it is a direct measure of the magnitude of rotation:

[3] Beware: some authors use the notation $\mathbf{q} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} + q_4$, so that q_4 is the scalar part.

$q_0 = 1$: no rotation at all.
$q_0 \approx 1$: small rotation.
$q_0 = 0$: rotation of 180º.
$q_0 < 0$: rotation larger than 180º.

The direction of the vectorial part $\begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^T$ is the rotation axis. Its length equals $\sin\frac{1}{2}\Phi$. Therefore, quaternions are well defined if $\Phi = 0$. This is unlike the axis-angle representation, in which the vector $\mathbf{e}$ is undefined in case of zero rotations.

D.4.1 Non-uniqueness

The pair $(\mathbf{e}, \Phi)$ is equivalent with $(-\mathbf{e}, -\Phi)$. However, $\cos(-\frac{1}{2}\Phi) = \cos(\frac{1}{2}\Phi)$ and $-\sin(-\frac{1}{2}\Phi) = \sin(\frac{1}{2}\Phi)$. Therefore, the two equivalent axis-angle representations have just one quaternion representation.

The pair $(\mathbf{e}, \Phi)$ represents the same orientation as $(\mathbf{e}, \Phi - 2\pi)$. Yet, they differ in the sense that $(\mathbf{e}, \Phi)$ could be the short way to rotate frame A to the orientation of frame B. If this is true, and $\Phi \geq 0$, then $(\mathbf{e}, \Phi - 2\pi)$ is the long way around. If the short angle is negative, $\Phi < 0$, then $(\mathbf{e}, \Phi + 2\pi)$ is the long way. This property is reflected in the quaternions by realizing that: $\sin(\frac{1}{2}\Phi) = -\sin\frac{1}{2}(\Phi - 2\pi) = = -\sin\frac{1}{2}(\Phi + 2\pi)$ and $\cos(\frac{1}{2}\Phi) = -\cos\frac{1}{2}(\Phi - 2\pi) = = -\cos\frac{1}{2}(\Phi + 2\pi)$. The consequence of this is that if the quaternion $\mathbf{q}$ represents the short angle rotation, then the long way rotation is represented by $-\mathbf{q}$. This is consistent with the earlier observation that $q_0 < 0$ represents a rotation larger than 180º, and that $q_0 > 0$ is associated with a rotation smaller than 180º.

D.4.2 Conversions

The rotation matrix that is associated with quaternions follows from their connection between the axis-angle representation, and from Rodriques formula, eq (57):

$$\mathbf{R} = \begin{bmatrix} -2q_2^2 - 2q_3^2 + 1 & 2q_0 q_3 + 2q_1 q_2 & 2q_1 q_3 - 2q_0 q_2 \\ 2q_1 q_2 - 2q_0 q_3 & -2q_1^2 - 2q_3^2 + 1 & 2q_0 q_1 + 2q_2 q_3 \\ 2q_0 q_2 + 2q_1 q_3 & 2q_2 q_3 - 2q_0 q_1 & -2q_1^2 - 2q_2^2 + 1 \end{bmatrix} \quad (67)$$

The inverse relation follows from inspecting the trace of this matrix. This yields:

$$q_0 = \pm\tfrac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + 1} \qquad q_2 = \frac{r_{31} - r_{13}}{4q_0}$$
$$q_1 = \frac{r_{23} - r_{32}}{4q_0} \qquad q_3 = \frac{r_{12} - r_{21}}{4q_0} \quad (68)$$

This is singular if $q_0 = 0$. A variant of eq (68), called *Sheppard's method*, avoids this singularity.

D.4.3 Pros and Cons

The major benefits of quaternions are:

- There is no singularity. This is in contrast with Euler angles (gimbal lock) and with the axis-angle representation (axis undefined if the rotation angle is zero).
- There is only one constraint as opposed to rotation matrices that have six constraints.
- A sequence of two rotations, i.e. $\mathbf{q}^A$ and $\mathbf{q}^B$ is accomplished by quaternion multiplication:

$$\mathbf{q}^C = \mathbf{q}^B \otimes \mathbf{q}^A = \begin{bmatrix} q_0^B & -q_1^B & -q_2^B & -q_3^B \\ q_1^B & q_0^B & q_3^B & -q_2^B \\ q_2^B & -q_3^B & q_0^B & q_1^B \\ q_3^B & q_2^B & -q_1^B & q_0^B \end{bmatrix} \begin{bmatrix} q_0^A \\ q_0^A \\ q_0^A \\ q_0^A \end{bmatrix} \quad (69)$$

  Note that the matrix, which accomplishes the quaternion multiplication is orthonormal, and thus can easily be inverted to implemented subtraction of rotations.

A disadvantage is that the quaternions are difficult to visualize.

Useful Functions in Matlab (Robotic system toolbox)

`axang2quat`	axis-angle rotation to quaternion
`axang2rotm`	axis-angle rotation to rotation matrix
`axang2tform`	axis-angle rotation to homogeneous transformation
`eul2quat`	Euler angles to quaternion
`eul2rotm`	Euler angles to rotation matrix
`eul2tform`	Euler angles to homogeneous transformation
`quat2axang`	quaternion to axis-angle rotation
`quat2eul`	quaternion to Euler angles
`quat2rotm`	quaternion to rotation matrix
`quat2tform`	quaternion to homogeneous transformation
`rotm2axang`	rotation matrix to axis-angle rotation
`rotm2eul`	rotation matrix to Euler angles
`rotm2quat`	rotation matrix to quaternion
`rotm2tform`	rotation matrix to homogeneous transformation
`tform2axang`	homogeneous transformation to axis-angle rotation
`tform2eul`	Extract Euler angles from homogeneous transformation
`tform2quat`	Extract quaternion from homogeneous transformation
`tform2rotm`	Extract rotation matrix from homogeneous transformation
`tform2trvec`	Extract translation vector from homogeneous transformation
`angdiff`	Difference between two angles
`cart2hom`	Cartesian coordinates to homogeneous coordinates
`hom2cart`	homogeneous coordinates to Cartesian coordinates
`trvec2tform`	translation vector to homogeneous transformation