BIOROB
EPFL Biorobotics Laboratory

The Jacobian

DON'T PANIC

Jesse van den Kieboom
jesse.vandenkieboom@epfl.ch

1 Introduction

The Jacobian is an important concept in robotics. Although the general concept of the Jacobian in robotics is not complicated, deriving and using the Jacobian appropriately is often not an entirely trivial task. The goal of this manual is to explain in clear terms, with concrete examples, everything you ever wanted to know about what the Jacobian is and how you can use it.

1.1 The Jacobian

Per definition, the Jacobian matrix is the matrix of all the first order partial derivatives of a vector function towards another vector. So if we have some vector $\mathbf{q}$ and a vector function $\mathbf{x}$ of $\mathbf{q}$ ($\mathbf{x}(\mathbf{q})$), then the Jacobian J of $\mathbf{x}$ is defined as:

$$J(\mathbf{x}(\mathbf{q})) = \begin{bmatrix} \frac{\partial x_1}{\partial q_1} & \cdots & \frac{\partial x_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_m}{\partial q_1} & \cdots & \frac{\partial x_m}{\partial q_n} \end{bmatrix} \tag{1}$$

In other words, each element J_{ij} relates a change of q_j to a change of x_i while keeping all $q_k, k \neq j$ stationary. Furthermore, recall that using the chain rule, the total derivative of $x_i(\mathbf{q})$ (i.e. the ith function in $\mathbf{x}$) towards time is given by:

$$\dot{x}_i = \sum_{j=1}^{n} \frac{\partial x_i}{\partial q_j} \cdot \dot{q}_j \tag{2}$$

i.e. the sum (over all j) of: the partial derivative of x_i towards q_j multiplied by the time derivative of q_j. In matrix notation (with $\mathbf{q}$ a column vector), this simply becomes:

$$\dot{\mathbf{x}} = J\dot{\mathbf{q}} \tag{3}$$

When people refer to "The Jacobian" in robotics, what they actually mean to say is "The Jacobian of a point in Cartesian space as a function of joint angles". They don't actually say that of course, but every time you read "The Jacobian" begin mentioned, you can just substitute it with the above sentence and things should become more clear.

More concretely though, the Jacobian referred to in robotics relates generalized coordinate velocities to Cartesian velocities. If we restrict ourselves to revolute and prismatic joints, then these generalized coordinates are simply the joint angle and joint translation respectively. It's important to note that this Jacobian relates velocities, and is thus involves only the kinematic model of a robot (no dynamics).

We have so far not yet talked about coordinate frames, however a Jacobian always relates joint velocities to a Cartesian velocity in a particular frame of reference. It's therefore important to keep in mind that a Jacobian is always associated with a specific coordinate frame. We will look into more detail about what this means exactly later.

Before we continue to show how to derive the Jacobian of a particular kinematic model of a robot, we first briefly introduce Spatial vector algebra, since we will use it to do all the calculations and derivations.

1.2 Spatial vector algebra

In this manual, we will use Spatial vector algebra to formulate the kinematic model. Here we only very briefly introduce the most important concepts, but for a more thorough and nice introduction tutorial please see Featherstone [2010a,b].

Spatial vector algebra is a mathematical tool which can be used to express physical properties of rigid body systems in a natural way. A spatial vector describing motion (using plücker coordinates) is a 6D vector composed of 3 angular velocities and 3 linear velocities, or:

$$\underline{\hat{\nu}} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ \nu_{O_x} \\ \nu_{O_y} \\ \nu_{O_z} \end{bmatrix} \tag{4}$$

Where $\omega_{x,y,z}$ are the Cartesian angular velocities around the unit axes x, y, z of the joint and $\nu_{O_{x,y,z}}$ are the Cartesian linear velocities. Note that we have ordered the angular velocities before the linear velocities.

1.2.1 Spatial transformations

A general transformation (rotations and translations) of a spatial motion vector can be done using a spatial transformation matrix. If we define two coordinate frames A and B, then the transformation ${}^{B}X_{A}$ of a spatial motion vector ${}^{A}\mathbf{m}$ (in A coordinates) from frame A to frame B is given by:

$${}^{B}\mathbf{m} = {}^{B}X_{A}{}^{A}\mathbf{m} \tag{5}$$

$${}^{B}X_{A} = \begin{bmatrix} \mathbf{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{r}\times & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{E} & \mathbf{0} \\ -\mathbf{E}\mathbf{r}\times & \mathbf{E} \end{bmatrix} \tag{6}$$

Here, $\mathbf{E}$ is a 3-by-3 rotation matrix, $\mathbf{I}$ is a 3-by-3 identity matrix, $\mathbf{0}$ is a 3-by-3 zero matrix and $\mathbf{r}$ is a 3-by-1 translation vector. $\mathbf{r}\times$ denotes the skew symmetric matrix of $\mathbf{r}$, which is defined as:

$$\mathbf{r}\times = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \tag{7}$$

It is useful to understand the meaning of the skew symmetric matrix as the matrix form of the cross product. Additionally, recall that the linear velocity $\frac{\mathbf{dr}}{dt}$ of a vector $\mathbf{r}$ resulting from an angular velocity $\boldsymbol{\omega}$ is given by:

$$\frac{\mathbf{dr}}{dt} = \boldsymbol{\omega} \times \mathbf{r} \tag{8}$$

It should now be clear how the spatial motion transformation ${}^{B}X_{A}$ is derived and that it can be used to transform spatial motion vectors from one frame to the other. Note that using this formulation, it becomes trivial to transform motion vectors (velocities) from one frame to another which will be very useful when deriving the Jacobian.

1.2.2 Useful operations

There are some other useful operations on spatial vectors and spatial transforms. The following table lists a few that will be relevant later.

Note that the spatial motion transform inverse can be obtained by simply transposing each of the 3-by-3 block matrices. For the pure rotations, the $\mathbf{E}_{x,y,z}(\theta)$ definitions are standard 3D rotation matrices.

Although we have only touched upon the subject of spatial vector algebra here, the concepts defined above are sufficient to understand the derivation and use of the Jacobian.

Quantity	Expression	
Spatial inverse	${}^{A}X_{B}$	$= {}^{B}X_{A}^{-1} = \begin{bmatrix} \mathbf{E}^{T} & \mathbf{0}_{3\times3} \\ (-\mathbf{Er}\times)^{T} & \mathbf{E}^{T} \end{bmatrix}$
Spatial translation	$\mathrm{Xtr}(r)$	$= \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ -\mathbf{r}\times & \mathbf{I}_{3\times3} \end{bmatrix}$
Spatial rotation	$\mathrm{Xrot}_{x,y,z}(\theta)$	$= \begin{bmatrix} \mathbf{E}_{x,y,z}(\theta) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{E}_{x,y,z}(\theta) \end{bmatrix}$
$\mathbf{E}_{x}(\theta)$		$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix}$
$\mathbf{E}_{y}(\theta)$		$= \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$
$\mathbf{E}_{z}(\theta)$		$= \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Spatial quaternion	$\mathrm{Xquat}(\mathbf{q})$	$= \begin{bmatrix} \mathbf{E}_{q}(q) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{E}_{q}(q) \end{bmatrix}$
$\mathbf{E}_{q}(q)$		$= 2\begin{bmatrix} q_1^2+q_2^2-0.5 & q_2q_3+q_4q_1 & q_2q_4-q_3q_1 \\ q_2q_3-q_4q_1 & q_1^2+q_3^2-0.5 & q_3q_4+q_2q_1 \\ q_2q_4+q_3q_1 & q_3q_4-q_2q_1 & q_1^2+q_4^2-0.5 \end{bmatrix}$

2 Derivation

In this section we will derive the Jacobian from a kinematic model description in a generic way. We will see that using spatial motion vectors, the Jacobian can be derived trivially in a manner often referred to as the geometrical derivation. Before we can derive the Jacobian however, we will first show how to describe the kinematics of the robot.

2.1 Kinematic Description

For the kinematic description of the robot we will adopt the methodology from Featherstone [2008]. We will need to define several quantities to describe the robot kinematics in a general sense. Here we limit ourselves to revolute and prismatic joints, but the method can be extended for other joint types easily (see Featherstone [2008]). The kinematic description of the robot consists of joint descriptions and the way in which joints are connected (the kinematic tree). Joints are described by two quantities, 1) a motion subspace $\mathbf{S}_i$ (or free-modes matrix) and 2) a coordinate transform ${}^{i}\mathbf{X}_{\lambda(i)}$, transforming parent coordinates to joint coordinates. The motion subspace $\mathbf{S}$ essentially describes the mapping from generalized coordinate velocities ($\boldsymbol{\nu}$) to spatial velocities, i.e.:

$$\boldsymbol{\nu}_i = \mathbf{S}_i \dot{\mathbf{q}}_i \tag{9}$$

and is defined for the revolute joint as $\mathbf{S}_i = \begin{bmatrix} \mathbf{a}_{1\times 3} & 0 & 0 & 0 \end{bmatrix}^T$, with $\mathbf{a}$ the joint axis (i.e. $\mathbf{a} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ for a revolute joint on the x-axis). Similarly, for prismatic joints $\mathbf{S}_i = \begin{bmatrix} 0 & 0 & 0 & \mathbf{a}_{1\times 3} \end{bmatrix}^T$.

The second quantity that needs to be described is the transformation from the parent frame to the joint frame. It is useful to separate define this transformation as the multiplication of two separate transformations. The first transformation is the transformation from the origin of the parent joint to the origin of the child joint (${}^{B}\mathbf{X}_A^T$), and is often simply a translation for most robotic systems. We can use the previously defined pure spatial translation ${}^{B}\mathbf{X}_A^T = \text{Xtr}(\mathbf{r}_i)$, with $\mathbf{r}_i$ being the translation vector. The second transformation is the transformation induced by the joint (${}^{B}\mathbf{X}_A^J$), depending on the value of the joint coordinates $\mathbf{q}$. For a revolute joint on the X-axis, we can use the earlier defined spatial rotation function ${}^{B}\mathbf{X}_A^J = \text{Xrotx}(q)$. The total transformation from the parent $\lambda(i)$ to body i is then given by:

$${}^{i}\mathbf{X}_{\lambda(i)} = {}^{B}\mathbf{X}_A^J \; {}^{B}\mathbf{X}_A^T \tag{10}$$

To define the kinematic tree, we will describe joint connectivity by a function $\lambda(i)$, meaning the index of the parent of joint i, and $\lambda(0) = -1$ (i.e. the robot base does not have a parent). We also define $\kappa(i)$ as the set of joints which are part of the kinematic chain from body i to the base.

2.2 Example Kinematic Model

Using the definitions of the previous section, we will now describe a simple kinematic model that we will use throughout the rest of the document. The kinematic model that we use is a basic lower extremities humanoid robot, with 6 degrees of freedom in each leg and a floating base (i.e. 6 DOFs). Table 1 describes general joint quantities for the various joint types used in our model. Note that the joint type quantities are independent of however the model is constructed. Table 2 describes all the specific model quantities that make up the final kinematic model. Note that the first joint ($i = 0$) is the floating base, which can be described by a 6-by-6 identity matrix for $\mathbf{S}_i$. Following the base are the two legs (which are symmetric) each containing six joints. Between the two legs, the only difference is the transformation of the first two joints (which displace the legs on the right and left side respectively). Note that we define the X-axis pointing forward, the Y-axis pointing to the left and the Z-axis pointing upward. The joints of each leg thus describe, in order, the hip pitch, hip roll, hip yaw, knee pitch, ankle pitch and ankle roll. Note that the numerical values of the transformations are chosen somewhat arbitrarily and are given in meters. Fig. 1 shows a schematic representation of the kinematics.

Tab. 1: Joint Type Description

Type	Motion Subspace ($\mathbf{S}_i$)	Joint Transform (${}^i\mathbf{X}^J_{\lambda(i)}$)
Floating	$\mathbf{1}_{6\times 6}$	Xtr($\mathbf{q}_{[5,6,7]}$) Xquat($\mathbf{q}_{[1,2,3,4]}$)
Revolute X	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$	Xrotx(q_1)
Revolute Y	$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$	Xroty(q_1)
Revolute Z	$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T$	Xrotz(q_1)

Tab. 2: Kinematic Description

Joint (i)	Parent ($\lambda(i)$)	Joint Type ($\mathbf{S}_i$)	Transform (${}^i\mathbf{X}^T_{\lambda(i)}$)
0	-1	Floating	$\mathbf{I}_{6\times 6}$
		Right Leg	
1	0	Revolute Y	Xtr($\begin{bmatrix} 0 & -0.10 & 0 \end{bmatrix}^T$)
2	1	Revolute X	Xtr($\begin{bmatrix} 0 & -0.10 & 0 \end{bmatrix}^T$)
3	2	Revolute Z	Xtr($\begin{bmatrix} 0 & 0 & -0.15 \end{bmatrix}^T$)
4	3	Revolute Y	Xtr($\begin{bmatrix} 0 & 0 & -0.15 \end{bmatrix}^T$)
5	4	Revolute Y	Xtr($\begin{bmatrix} 0 & 0 & -0.3 \end{bmatrix}^T$)
6	5	Revolute X	$\mathbf{I}_{6\times 6}$
		Left Leg	
7	0	Revolute Y	Xtr($\begin{bmatrix} 0 & 0.10 & 0 \end{bmatrix}^T$)
8	7	Revolute X	Xtr($\begin{bmatrix} 0 & 0.10 & 0 \end{bmatrix}^T$)
9	8	Revolute Z	Xtr($\begin{bmatrix} 0 & 0 & -0.15 \end{bmatrix}^T$)
10	9	Revolute Y	Xtr($\begin{bmatrix} 0 & 0 & -0.15 \end{bmatrix}^T$)
11	10	Revolute Y	Xtr($\begin{bmatrix} 0 & 0 & -0.30 \end{bmatrix}^T$)
12	11	Revolute X	$\mathbf{I}_{6\times 6}$

2.3 Jacobian

Now that we have everything defined and in order, lets derivate that juicy Jacobian already. Recall that the Jacobian relates joint velocities to Cartesian velocities in a particular frame A:

$$ {}^A\dot{\mathbf{x}} = {}^A J \dot{\mathbf{q}} \tag{11}$$

We are first going to derive the full body Jacobian with respect to the base (${}^0 J$), and then later show how to use this Jacobian to get the Jacobian of a particular body frame. It is useful to understand the contents of the Jacobian semantically. Each $i\text{th}_{6\times 1}$ column of the Jacobian represents the change of rate of the Cartesian velocity with respect to the generalized coordinate velocity $\dot{\mathbf{q}}_i$.

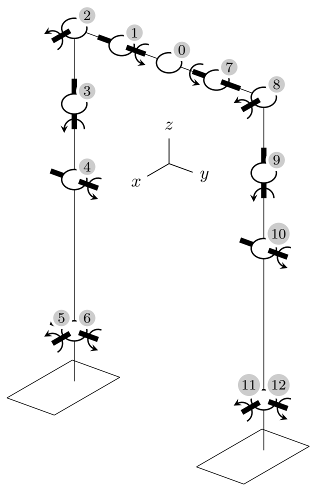

Fig. 1: Schematic representation of example kinematics.

Therefore, to calculate the ith column of the Jacobian, we need to calculate the velocity factor ${}^0\bar{\boldsymbol{\nu}}_i$ (as observed in the base) caused by $\dot{\mathbf{q}}_i$. Thus:

$$
{}^0J = \left[\begin{array}{cccc} {}^0\bar{\boldsymbol{\nu}}_0 & {}^0\bar{\boldsymbol{\nu}}_1 & \dots & {}^0\bar{\boldsymbol{\nu}}_n \end{array} \right] \tag{12}
$$

To calculate ${}^0\bar{\boldsymbol{\nu}}_i$, recall from Eq. 9 the calculation of the spatial velocity for a particular joint:

$$
\boldsymbol{\nu}_i = \mathbf{S}_i \dot{\mathbf{q}}_i \tag{13}
$$

i.e. $\bar{\boldsymbol{\nu}}_i = \mathbf{S}_i$ (in the local frame).

Secondly, remember that we can use spatial transformations to transform spatial motion vectors, such that we can calculate the velocity in the base frame (i.e. ${}^0\bar{\boldsymbol{\nu}}_i$) with:

$$
{}^0\bar{\boldsymbol{\nu}}_i = {}^0X_i \mathbf{S}_i \tag{14}
$$

$$
{}^0X_i = {}^0X_{\lambda(i)}{}^{\lambda(i)}X_i \tag{15}
$$

Note the recursive definition of the transformation ${}^0X_i$, which allows for efficient calculation of the base transformations of all bodies. With ${}^0\bar{\boldsymbol{\nu}}_i$ calculated, we have obtained the full body Jacobian ${}^0J$.

To obtain the Jacobian of a particular body b, with respect to the base, we can simply select the relevant columns of the full body Jacobian based on the kinematic chain leading up from body i to the base (i.e. $i \in \kappa(b)$)

$$
{}^0J_b = K_b{}^0J \tag{16}
$$

With K_b being a $6n \times 6n$ diagonal block matrix with $\mathbf{I}_{6\times 6}$ for each body $i \in \kappa(b)$ (recall the definition of $\kappa(i)$ in Section 2.1).

If you were expecting something more spectacular then you will be disappointed, this is really all there is to it. Due to the spatial vector algebra formulation, the calculation of the Jacobian becomes both intuitive and trivially simple.

There is one important caveat with this formulation of ${}^0J_b$ in that the coordinate frame refers to the **origin** of the frame of joint b. However, usually you will want to use the Jacobian of a body b with respect a particular point in body b (other than the origin) corresponding to for example the end-effector point of a chain. The intuitive way to look at this is to take the example of a revolute joint. At the origin of the revolute joint, the joint velocity does not induce any linear velocities. However, at a different point in the joint frame, you *do* observe a linear velocity. It is therefore important to transform the body Jacobian to take into account the end-effector point. To get the Jacobian with respect to a particular point ${}^b\mathbf{r}$ in body b, we can simply translate the body Jacobian using a spatial transformation:

$$ {}^0J_{b_r} = \mathrm{Xtr}({}^0\mathbf{r})\,{}^0J_b \tag{17}$$

Note that ${}^0J_b$ needs to be translated with $\mathbf{r}$ in base coordinates. We will show more about transformations on the Jacobian in Section 4.

3 Usage

The Jacobian calculated in the previous section can be used in a variety of different ways. The following sections briefly discuss some of the uses.

3.1 Inverse Kinematics

The Jacobian can be used to solve iterative inverse kinematics problems. The main idea is that by inverting the Jacobian we can write:

$$\dot{\mathbf{q}} = J^{-1}\dot{\mathbf{x}} \tag{18}$$

In other words, we can use the Jacobian to compute the required joint velocities given a desired Cartesian velocity. Given a current Cartesian position $\mathbf{p}$ and a desired Cartesian position $\mathbf{x}$, we can use the following control law to perform inverse kinematics, by discretization:

$$\mathbf{q}_{t+1} = \mathbf{q}_t + kJ^{-1}(\mathbf{x} - \mathbf{p}) \tag{19}$$

with k a control gain. There are several issues with using the Jacobian for inverse kinematics. The first issue is that J cannot be inverted in the general case (for example in the case of redundancy). This problem is usually resolved by instead using the pseudoinverse of the Jacobian, $J^\dagger$. Often the Moore-Penrose pseudoinverse is used, i.e. $J^\dagger = J^T(JJ^T)^{-1}$. The pseudoinverse will give a least-squares solution to $\dot{\mathbf{q}}$.

A second issue is that singularities can occur (e.g. aligned joints or unreachable targets) in which case the pseudoinverse becomes ill-posed. The classical example of such a singularity occurs when a serial manipulator is trying to reach an unreachable point.

Some of these issues are addressed by the damped least squares method to inverse kinematics. Instead of using the minimum norm solution to $J\dot{\mathbf{q}} = \dot{\mathbf{x}}$, the following quantity is minimized:

$$||J\dot{\mathbf{q}} - \dot{\mathbf{x}}||^2 + \lambda^2||\dot{\mathbf{q}}||^2 \tag{20}$$

with λ being a non-zero damping constant. The solution to this minimization is given by:

$$J^T(JJ^T + \lambda^2 I)\dot{\mathbf{q}} = \dot{\mathbf{x}} \tag{21}$$

The term $(JJ^T + \lambda^2 I)$ is square and thus a vector $\mathbf{f}$ can be solved in $(JJ^T + \lambda^2 I)\mathbf{f} = \dot{\mathbf{x}}$ by means of solving the linear system. After having solved for $\mathbf{f}$, the solution for $\dot{\mathbf{q}}$ is given by $J^T\mathbf{f}$. The choice of λ is important and should be sufficiently large to avoid singularities. However too large values of λ lead to slow convergence to the desired target position.

3.2 Virtual Force Control

The Jacobian is not only a mapping of velocities. By using the principle of virtual work, we can show that the Jacobian also maps forces. The virtual work $\delta\mathbf{U}$ done in a system as a result of an infinitesimal change δ of some coordinates $(\mathbf{c})$ is defined as:

$$\delta\mathbf{U} = \mathbf{F}_c^T \delta\mathbf{c} \tag{22}$$

The virtual work in a system is always the same, regardless of the coordinate system. Therefore, we can express the following two equations for the virtual work $\delta\mathbf{U}$ in the generalized coordinates ($\mathbf{q}$ with forces $\mathbf{F}_q$) and the coordinates of a particular body ($\mathbf{x}$ with forces $\mathbf{F}_x$).

$$\begin{aligned}
\delta\mathbf{U} &= \mathbf{F}_q^T \delta\mathbf{q} \\
\delta\mathbf{U} &= \mathbf{F}_x^T \delta\mathbf{x} \\
\mathbf{F}_q^T \delta\mathbf{q} &= \mathbf{F}_x^T \delta\mathbf{x} \\
\mathbf{F}_q^T &= \mathbf{F}_x^T \frac{\delta\mathbf{x}}{\delta\mathbf{q}} \qquad (23)
\end{aligned}$$

Since δ signifies an infinitesimal change, we can write:

$$\frac{\delta\mathbf{x}}{\delta\mathbf{q}} = \frac{\partial\mathbf{x}}{\partial\mathbf{q}} = J \tag{24}$$

Substituting and using $(AB)^T = B^T A^T$ we can finally write:

$$\mathbf{F}_q = J^T \mathbf{F}_x \tag{25}$$

This means that the effect of having a **virtual** Cartesian force can be achieved by simply using the transposed Jacobian. This type of virtual force control is therefore sometimes called Jacobian transposed control. Note that for revolute joints, $\mathbf{F}_q$ is often written as $\boldsymbol{\tau}$ (i.e. $\boldsymbol{\tau} = J^T\mathbf{F}$).

3.3 Null Space Projection

Another useful aspect of the Jacobian is related to the null space projection of the Jacobian. Recall that the Jacobian maps joint velocities to Cartesian velocities and that the null space of a matrix A (null(A)) is the matrix which project any vector $\mathbf{x}$ such that $A\mathbf{x} = 0$, and can be derived as:

$$\text{null}(A) = \mathbf{I} - A^\dagger A \tag{26}$$

Therefore, the null space of the Jacobian can be used to project joint velocities such that the resulting Cartesian velocities are $\mathbf{0}$. Thus, the point for which the Jacobian is derived does not move. One application of this can be found in Prioritized Task Control, in which several tasks are cascaded by projecting each task in the null space of the previous task's Jacobian.

The null space of the Jacobian can also be used in combination with the previously described inverse kinematics to introduce secondary objectives to the end-effector tracking. Secondary objectives include avoiding joint limits or singular configuration.

3.4 End Effector Force Estimation

Another use of the Jacobian comes from the mapping of forces using J^T. When joint force sensing is available, but end effector force sensing is not, then the Jacobian can be used to estimate (at least partially, depending on the configuration) the force exerted at the end effector. Recall that the Jacobian maps forces by:

$$\mathbf{F_q} = J^T \mathbf{F}_x \tag{27}$$

Knowing $\mathbf{F}_q$, the effective forces at the end effector can be obtained by:

$$\mathbf{F}_x = J^{-T} \mathbf{F_q} \tag{28}$$

Again, since J^T cannot be inverted in general, the pseudoinverse can be used instead to give a minimum norm solution to F_x.

3.5 Augmented Jacobian

If you have several Jacobi, for example for several tasks, then an Augmented Jacobian can be constructed by simply stacking the individual Jacobi under each other. The new, augmented Jacobian then contains both tasks which can then be controlled for simultaneously.

4 Transformations

Since a Jacobian is simply a linear mapping of velocities, we can use spatial motion transforms to transform a Jacobian to a different frame. In the previous sections we have always used the Jacobian defined for some point in the base frame. This Jacobian can be trivially converted to another frame, for example a body frame ${}^{i}X$:

$$ {}^{i}J = {}^{i}X_0 {}^{0}J \tag{29} $$

Here we have used the spatial motion transform ${}^{i}X_0$ transforming base coordinates to body coordinates. This is only a transformation of the coordinate frame of the Jacobian, but the base towards which the Jacobian is defined is still the robot base. We can use a similar transformation of the Jacobian to *rebase* the base Jacobian to another body. This effectively allows to calculate a Jacobian from any point on the robot body to any other point. An example of when this is useful is for legged robots. Whenever a leg (or limb) is in contact when the environment, a Jacobian assuming the limb in contact as a fixed base can be used to control the rest of the body.

Consider a body b representing the new, fictive fixed base and another body i being the end effector. Furthermore, assume that we have successfully calculated the Jacobi ${}^{0}J_b$ and ${}^{0}J_i$ using the methods described in previous sections (i.e. the Jacobi for each of the bodies b and i in the base). The new Jacobian ${}^{b}J_i$ (i.e. the Jacobian of i in b) can be obtained from:

$$ {}^{b}J_i = {}^{0}J_i - \mathrm{Xtr}({}^{0}\mathbf{x}_i - {}^{0}\mathbf{x}_b){}^{0}J_b \tag{30} $$

where ${}^{0}\mathbf{x}_b$ is the end effector location of b (in base coordinates) and ${}^{0}\mathbf{x}_i$ is the end effector location of i (in base coordinates).

5 Center of Mass Jacobian

The Jacobian of the Center of Mass (CoM) relates joint velocities to center of mass velocities and can be used to control the Center of Mass using any of the techniques previously described. The center of mass in base coordinates (${}^0\mathbf{x}^{\mathrm{CoM}}$) is calculated using:

$$ {}^0\mathbf{x}^{\mathrm{CoM}} = \frac{\sum_i m_i {}^0\mathbf{x}_i^{\mathrm{CoM}}}{M} \tag{31} $$

$$ M = \sum_i m_i \tag{32} $$

where m_i is the mass of body i, ${}^0\mathbf{x}_i^{\mathrm{CoM}}$ is the center of mass of body i in base coordinates and M is the total system mass. To derive the center of mass Jacobian ${}^0J^{\mathrm{CoM}}$, we can use exactly the same method as described in Section 2.3. The only difference is that we will use the transformation of the center of mass location to the base instead of the joint origin transformation and scale by the mass proportion.

$$ {}^0\bar{\boldsymbol{\nu}}_i^{\mathrm{CoM}} = \frac{m_i}{M} {}^0X_i^{\mathrm{CoM}} \mathbf{S}_i \tag{33} $$

$$ {}^0X_i^{\mathrm{CoM}} = {}^0X_i \ \mathrm{Xtr}(\mathbf{x}_i^{\mathrm{CoM}}) \tag{34} $$

Finally, the center of mass Jacobian is obtained by:

$$ {}^0J^{\mathrm{CoM}} = \begin{bmatrix} {}^0\bar{\boldsymbol{\nu}}_0^{\mathrm{CoM}} & {}^0\bar{\boldsymbol{\nu}}_1^{\mathrm{CoM}} & \dots & {}^0\bar{\boldsymbol{\nu}}_n^{\mathrm{CoM}} \end{bmatrix} \tag{35} $$

References

Roy Featherstone. *Rigid body dynamics algorithms*, volume 49. Springer New York, 2008.

Roy Featherstone. A beginner's guide to 6-d vectors (part 1). *Robotics & Automation Magazine, IEEE*, 17(3):83–94, 2010a.

Roy Featherstone. A beginner's guide to 6-d vectors (part 2). *Robotics & Automation Magazine, IEEE*, 17(4):88–99, 2010b.