Multibody System Dynamics (2023) 60:3–25
https://doi.org/10.1007/s11044-023-09938-0

RESEARCH

Human motion capture, reconstruction, and musculoskeletal analysis in real time

Urbano Lugrís[1] · Manuel Pérez-Soto[2] · Florian Michaud[1] · Javier Cuadrado[1]

Received: 17 April 2023 / Accepted: 12 September 2023 / Published online: 6 October 2023

Abstract
Optical motion capture is an essential tool for the study and analysis of human movement. Currently, most manufacturers of motion-capture systems provide software applications for reconstructing the movement in real time, thus allowing for on-the-fly visualization. The captured kinematics can be later used as input data for a further musculoskeletal analysis. However, in advanced biofeedback applications, the results of said analysis, such as joint torques, ground-reaction forces, muscle efforts, and joint-reaction forces, are also required in real time.

In this work, an extended Kalman filter (EKF) previously developed by the authors for real-time, whole-body motion capture and reconstruction is augmented with inverse dynamics and muscle-efforts optimization, enabling the calculation and visualization of the latter, along with joint-reaction forces, while capturing the motion.

A modified version of the existing motion-capture algorithm provides the positions, velocities, and accelerations at every time step. Then, the joint torques are calculated by solving the inverse-dynamics problem, using force-plate measurements along with previously estimated body-segment parameters. Once the joint torques are obtained, an optimization problem is solved, in order to obtain the muscle forces that provide said torques while minimizing an objective function. This is achieved by a very efficient quadratic programming algorithm, thoroughly tuned for this specific problem.

With this procedure, it is possible to capture and label the optical markers, reconstruct the motion of the model, solve the inverse dynamics, and estimate the individual muscle forces, all while providing real-time visualization of the results.

Keywords Biomechanics · Motion capture · Musculoskeletal analysis · Biofeedback · Kalman filter

✉ U. Lugrís
urbano.lugris@udc.es

M. Pérez-Soto
manuel.perez@unizar.es

1 Laboratory of Mechanical Engineering, University of La Coruña, Mendizábal s/n, 15403 Ferrol, Spain

2 IDERGO, University of Zaragoza, María de Luna 3, 50010 Zaragoza, Spain

1 Introduction

Motion capture, in its multiple forms, is possibly the most widely used technique in biomechanical analysis. The data it provides can be directly used in many cases to perform basic studies, focused only on kinematics. However, it is also possible to carry out more advanced analyses, based on additional information such as joint torques, which are in turn obtained by applying the acquired motion to skeletal models. Going even further, the use of complex musculoskeletal models also allows the estimation of joint-reaction forces and muscle efforts. There are currently multiple software packages, both open source and commercial, that provide such models, among which OpenSim [1] (NCSRR, Stanford CA, USA), AnyBody [2] (AnyBody Technology A/S, Aalborg, Denmark), or BoB [3] (BoB Biomechanics, Bromsgrove, UK) are well-known examples. However, they are essentially designed as post-processing tools, so they do not provide the results in real time.

Obtaining the musculoskeletal analysis results in real time can offer many benefits. On the one hand, it enables for early detection of motion-capture problems, thus avoiding the problem of finding that a take had errors when the subject is no longer available to repeat it. On the other hand, the availability of musculoskeletal data in real time can serve not only to improve the biofeedback-based methodologies currently used in sports and rehabilitation, but also to enable the creation of new and more advanced applications. Giggins et al. [4] conducted a literature review on the benefits of real-time biofeedback in rehabilitation, including many studies based on different technologies, such as electromyography (EMG), inertial measurement units (IMUs) and optical motion-capture systems. Some of the reviewed studies demonstrated that the use of real-time biofeedback improved rehabilitation outcomes, whereas others were less conclusive. There are specific applications, such as lower-back postural control, where the use of simple motion sensors to provide real-time feedback has been shown to be effective as a tool to reduce pain [5]. Later studies used IMUs to provide real-time feedback for more complex movements. For example, Carpinella et al. [6] used six IMUs attached to the lower limbs and the trunk to provide visual and acoustic feedback to Parkinson's patients during rehabilitation, which helped to improve their balance. Cerqueira et al. [7] used a similar setup for the upper extremities, this time providing vibrotactile feedback whenever the ranges of motion of the trunk, neck, and arm joints reached compromised values, effectively reducing ergonomic risk. Real-time biofeedback has also demonstrated its advantages in sports applications. For example, Chan et al. [8] used force plates to retrain the gait of novice long-distance runners, and the results showed that participants who trained with real-time feedback had a lower injury rate than those who received the same training without it.

Obtaining musculoskeletal analysis results in real time from optical motion capture is a challenging task. However, among the multiple technologies available for motion capture, systems based on IR cameras and reflective markers are currently the most accurate, which makes them remain the gold standard [9]. Inertial systems are known to enable simpler and faster reconstruction of kinematics [10], but at the cost of being less accurate, as well as suffering from translational drift, as they lack an absolute position reference. Nowadays, real-time motion reconstruction and visualization has become a standard feature of optical systems, although the motion-reconstruction algorithms developed by manufacturers are not usually published. The traditional approach for working with optical markers requires a rather complex manual postprocessing [11, 12], although there exist more automatic methods in the literature, such as algorithms based on nonlinear optimization [13], or on the Kalman filter [14–17], but none of them are intended for real-time use. Among the published motion-reconstruction methods targeted at real-time applications, very few use reflective markers [18].

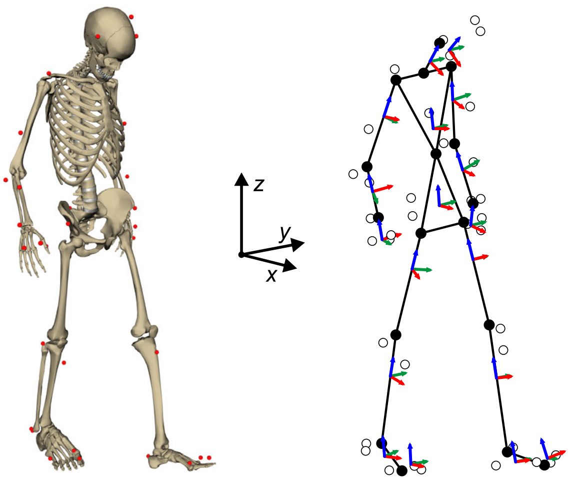
Methods already exist that provide real-time inverse dynamics at the skeletal level. Pizzolato et al. [19] adapted OpenSim to be able to estimate joint torques while receiving motion-capture data in real time. They achieved processing times as low as 0.5 ms per frame on a 23-DoF model, by solving the kinematics with a multithreaded implementation of the nonlinear optimization algorithm described by Lu and O'Connor [13]. In addition, the paper presents a comprehensive study of the effects on the joint torques caused by the delay induced by real-time low-pass filtering of kinematic data. Giarmatzis et al. [20] resorted to a totally different approach, by using machine-learning techniques to achieve real-time estimation of the joint torques produced at the lower limbs during gait, regardless of the technology used for the motion capture. The main drawback of the method stems from its data-driven nature, since the analysis is limited to the specific activity used to train the algorithm.

Most implementations of real-time musculoskeletal dynamics require the use of EMG. For instance, Sartori et al. [21] used an EMG-driven muscle model to estimate muscle activations in real-time at several degrees of freedom. Murai et al. [22] achieved real-time musculoskeletal analysis and visualization in a large-scale model by using EMG measurements where possible, and assuming certain activation correlations to estimate the forces of the remaining nonmeasured muscles. There are also real-time musculoskeletal dynamics implementations that do not require EMG. Van den Bogert et al. [23] used the same nonlinear least-squares method as Pizzolato et al. to solve the kinematics at every time step, followed by inverse dynamics and static optimization for the muscle-redundancy problem. This work also used a large-scale musculoskeletal model, with 44 degrees of freedom and 300 muscles, achieving real-time performance at 120 Hz. In order to obtain such performance, the authors limited the iterations of both the kinematics solver and the muscle optimizer, and used an approximated method to compute the moment arms of the muscles. More recently, Stanev et al. [24] also implemented real-time musculoskeletal analysis and visualization, in this case using OpenSim. The authors focused again on the filtering delay, and proposed a new method to minimize it. The described motion capture and musculoskeletal analysis algorithm is theoretically able to work with either optical or inertial-based motion-capture systems. However, its real-time capabilities were only tested using the latter, with a reduced model including only the lower limbs, and using approximated muscle-moment arms.

In this work, the objective is to develop and implement a highly efficient algorithm for whole-body motion capture using IR cameras and optical markers, along with force-plate measurements, with the ability to estimate and visualize muscle efforts in real time. This is achieved by combining a Kalman filter to estimate the skeletal kinematics, with a multibody formulation to calculate the joint torques, and an optimization algorithm for obtaining the muscle efforts producing such torques. The Kalman filter is derived from an existing EKF developed for real-time motion capture [18], with some modifications to include the estimation of accelerations.

The resulting algorithm is highly robust and efficient, thus leaving enough computational headroom to increase the complexity of the musculoskeletal model, either by including muscle wrapping [25–27], more sophisticated physiological muscle models [28–30], or improved joint kinematics [25], including closed-loop models [31, 32] that can be solved iteratively or using lookup tables. It is also possible to incorporate additional sensors into the Kalman filter, such as IMUs, to reinforce the robustness and accuracy of the motion-capture process, or to integrate the equations of motion into the plant model [33–35], thus allowing to also add the force plates as sensors, which would be beneficial for both motion acquisition and joint-torques estimation.

Fig. 1 Multibody model showing joints and optical markers



2 Real-time motion capture with musculoskeletal analysis

This section describes in detail all the elements involved in the motion-capture system with real-time musculoskeletal analysis developed in this paper: the definition of the biomechanical model, the Kalman filter used for reconstructing the motion, the procedure for solving the inverse dynamics at every time step, and the optimization approach for estimating the muscular forces.

2.1 Biomechanical model

The biomechanical model used here, which is shown in Fig. 1, is the same one employed in the previously developed EKF [18] that, in turn, derives from the work by Silva and Ambrósio [36]. It consists of $n_b = 18$ rigid bodies: pelvis, trunk, neck, head, arms, forearms, hands, thighs, shanks, feet, and toes. The axes of the global reference frame are defined as follows: x-axis in the anteroposterior direction, y-axis in the mediolateral direction, and z-axis in the vertical direction. The local axes of the body segments are defined parallel to the global ones, when the subject is in a standing position, with the arms lowered down and the palms of the hands facing inward.

The joint between the pelvis and the trunk is located between T12 and L1 vertebrae, and the neck base is placed between T3 and T4, to keep it at the same height as the shoulder joints. Most body segments are connected together through spherical joints, with three exceptions: there is a universal joint between the trunk and the base of the neck, and each toe segment is attached to its corresponding foot by means of a revolute joint. This is done to avoid placing extra markers on the neck and toes, which would be required to capture the eliminated degrees of freedom. According to this model definition, the total number of degrees of freedom n_z is equal to 52: three translations and three rotations at the pelvis, two rotations at the base of the neck, one rotation at each metacarpophalangeal joint, and 42 absolute Euler angles at the remaining spherical joints. These variables are grouped into a vector of independent positions $\mathbf{z}$, which will constitute the base of both the motion-capture EKF and the equations of motion used for the inverse dynamics.

In the previous EKF implementation, the model coordinates were defined using absolute angles for the bodies with spherical proximal joints, because this allowed us to define a simpler and more efficient observation function, with a very sparse Jacobian matrix. However, using absolute angles is not the best option when solving the dynamics, so most formulations in independent coordinates rely on relative coordinates to better exploit recursivity when assembling the dynamic terms [30, 37]. Nevertheless, this set of coordinates will be kept here also for the dynamics, mainly for two reasons. First, keeping the same model variables allows reuse of the code previously developed in [18] for the observation function and its Jacobian matrix. Secondly, the dynamic terms obtained by using absolute rotations are easier to linearize. Although this advantage is not relevant to this work, it will facilitate one of its future goals: the implementation of an EKF that incorporates the equations of motion into the plant model [33–35], using the force plates as sensors.

2.2 EKF for motion capture

The dynamic system used in this work has a linear plant model, but the observation model is highly nonlinear, so it is not possible to use a conventional Kalman filter. There are many complex and accurate ways to deal with nonlinearity when using Kalman filters, such as the unscented Kalman Filter (UKF) [38, 39], however, since in this work the goal is to meet real-time requirements, the simpler extended Kalman filter (EKF) [38–40] is chosen for the task.

The equations of the predictor–corrector algorithm used by the EKF are well known, so they will not be repeated here for the sake of brevity. Instead, the definitions of all the involved terms will be addressed, namely the state vector $\mathbf{x}$, the state propagation matrix $\boldsymbol{\Phi}$, the process noise covariance matrix $\boldsymbol{\mathcal{Q}}$, the observation function $\mathbf{h}(\mathbf{x})$, and the sensors noise covariance matrix $\boldsymbol{\mathcal{R}}$.

2.2.1 Plant model

The filter uses a kinematic state-propagation model, similar to the second-order one used in [18]. However, in this case, since the calculation of the inverse dynamics needs the accelerations, a third-order discrete Wiener process acceleration model (DWPA) is used instead [40]. The state propagation is defined directly as a discrete model, where the acceleration increment at every time step is modeled as discrete Gaussian noise. For each degree of freedom i of the model, a state vector $\mathbf{x}_i$ containing its position, velocity and acceleration propagates in time between two consecutive time steps k and $k+1$ as follows:

$$\mathbf{x}_i^{k+1} = \boldsymbol{\Phi}_i \mathbf{x}_i^k + \boldsymbol{\Gamma}_i w_i^k \qquad i = 1, \ldots, n_z, \tag{1}$$

where w_i^k represents the increment of acceleration (or angular acceleration, depending on the nature of the corresponding degree of freedom) along time step k. The state-propagation matrix $\boldsymbol{\Phi}_i$ and the noise gain vector $\boldsymbol{\Gamma}_i$ are both constant and equal for all degrees of freedom, with the following structure:

$$\begin{bmatrix} z_i \\ \dot{z}_i \\ \ddot{z}_i \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z_i \\ \dot{z}_i \\ \ddot{z}_i \end{bmatrix}_{k} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \\ 1 \end{bmatrix} w_i^k \qquad i = 1, \ldots, n_z. \tag{2}$$

The time step Δt is fixed to 10 ms, since the motion-capture system provides the marker coordinates at a 100 Hz rate. The noise value w_i^k represents a random acceleration increment, which is assumed to remain constant along the whole time step. The state vector $\mathbf{x}$ and propagation matrix $\boldsymbol{\Phi}$ of the whole system are obtained by assembling all the individual $\mathbf{x}_i$ and $\boldsymbol{\Phi}_i$ terms, leading to a system of dimension $3 \times n_z$.

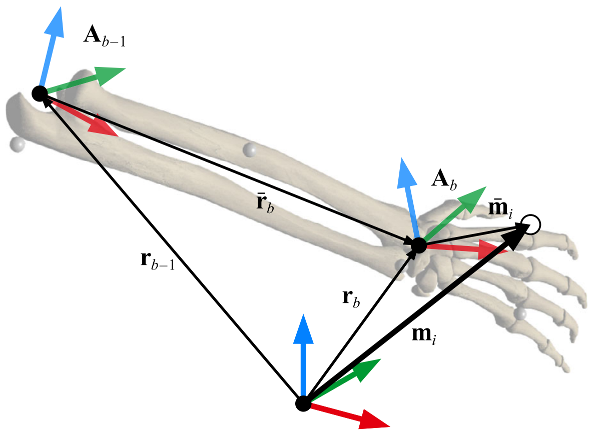


Fig. 2 Recursive calculation of absolute marker positions

Assuming that the discrete Gaussian noise w_i has a variance $\sigma_{w_i}^2$, the process noise covariance matrix $\mathcal{Q}_i$ can be obtained as [40]:

$$\mathcal{Q}_i = \Gamma_i \sigma_{w_i}^2 \Gamma_i^{\mathrm{T}} = \begin{bmatrix} \frac{1}{4}\Delta t^4 & \frac{1}{2}\Delta t^3 & \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^3 & \Delta t^2 & \Delta t \\ \frac{1}{2}\Delta t^2 & \Delta t & 1 \end{bmatrix} \sigma_{w_i}^2 \qquad i = 1, \ldots, n_z, \tag{3}$$

which is also constant, although not necessarily equal for all degrees of freedom. The complete $\mathcal{Q}$ matrix for the whole system is obtained by assembling the individual $\mathcal{Q}_i$ matrices, according to the arrangement of variables used in the state vector $\mathbf{x}$.

2.2.2 Observation model

The observation function $\mathbf{h}(\mathbf{x})$ must provide the theoretical values of the sensors, i.e., a vector $\mathbf{m}$ containing the absolute positions of the n_m optical markers, as a function of the system states $\mathbf{x}$. Each virtual marker i is considered as rigidly attached to an underlying body segment b, at a fixed location $\bar{\mathbf{m}}_i$ within its local axes. The body segment b itself is defined by the position vector $\mathbf{r}_b$ of its proximal joint, and a rotation matrix $\mathbf{A}_b$. As shown in Fig. 2, the absolute position of a given marker i is the sum of the position vector of the origin of its underlying body segment, and the local coordinates of the marker transformed into the global axes by $\mathbf{A}_b$:

$$\mathbf{m}_i = \mathbf{r}_b + \mathbf{A}_b \bar{\mathbf{m}}_i \qquad i = 1, \ldots, n_m \quad b = 1, \ldots, n_b. \tag{4}$$

The position of the origin of body b, in turn, can be calculated recursively in the same way, as the origin $\mathbf{r}_{b-1}$ of its proximal body, plus the local coordinates $\bar{\mathbf{r}}_b$ of the proximal joint of body b within the frame of reference of body $b-1$, also rotated to the global frame:

$$\mathbf{r}_b = \mathbf{r}_{b-1} + \mathbf{A}_{b-1} \bar{\mathbf{r}}_b \qquad b = 1, \ldots, n_b. \tag{5}$$

Both the local coordinates of joints and optical markers are assumed to be constant, although their values will depend on the size of the subject.

The $3 \times n_m$ coordinates of the optical markers used as sensors in this work are considered independent, thus leading to a diagonal sensor-noise covariance matrix $\mathcal{R}$. The total noise

Table 1 Unscaled joint coordinates

Joint	$\bar{x}$ (m)	$\bar{y}$ (m)	$\bar{z}$ (m)
Lumbar joint	0.0000	0.0000	0.0000
Neck base	0.0000	0.0000	0.2574
Head base	0.0000	0.0000	0.1604
R. shoulder	0.0000	−0.1575	0.2574
R. elbow	0.0000	0.0000	−0.2753
R. wrist	0.0000	0.0000	−0.2362
L. shoulder	0.0000	0.1575	0.2574
L. elbow	0.0000	0.0000	−0.2753
L. wrist	0.0000	0.0000	−0.2362
R. hip	0.0000	−0.0795	−0.2400
R. knee	0.0000	0.0000	−0.3875
R. ankle	0.0000	0.0000	−0.3615
R. toe base	0.1017	0.0000	−0.0511
L. hip	0.0000	0.0795	−0.2400
L. knee	0.0000	0.0000	−0.3875
L. ankle	0.0000	0.0000	−0.3615
L. toe base	0.1017	0.0000	−0.0511

affecting each marker measurement is the result of two effects: the noise introduced by the motion-capture system, and the skin-motion artifact [41]. Whereas the former is very small, and equal for all markers, the latter is significantly large and, moreover, it does not behave as Gaussian noise, since it is correlated to the motion of the underlying bodies, and does not contain a flat frequency spectrum. This fact, along with the use of a Kalman filter on a nonlinear system, renders the resulting estimation nonoptimal in a statistical sense.

The skin-motion artifact does not affect all markers equally, so their noise variances will differ. Therefore, it is possible to finely tune $\mathcal{R}$ by setting lower variances for markers that are closely attached to the bone, and higher values for the most unreliable ones. However, for the sake of simplicity, and taking into account that this work is mostly focused on obtaining a working real-time application, all markers will be assigned the same noise variance σ_m^2.

2.2.3 Model scaling

The local coordinates of the joints $\bar{\mathbf{r}}_b$ present in the observation function must match the dimensions of the subject, so they need to be properly scaled before running the Kalman filter. The reference unscaled dimensions, which are detailed in Table 1, are determined by the geometry of the 3D bone models used for the graphics output, which are in turn taken form the BodyParts3D digital library [42].

A first approach to scaling a body segment could be multiplying the components of all the $\bar{\mathbf{r}}_b$ and $\bar{\mathbf{m}}_i$ vectors attached to it by a set of three independent scaling factors, defined along the directions of its local axes. However, using three independent factors per segment leads to inaccurate results in many situations. For instance, when the marker distances along a certain local axis are too short, the scaling factor on that direction is very sensitive to small marker displacements. For this reason, a reduced set $\mathbf{k}$ of independent scaling factors is used instead. The relationship between the independent scale factors and their dependent counterparts is defined in Table 2.

Table 2 Scaling factors

Segment	k_{x_b}	k_{y_b}	k_{z_b}
Pelvis	k_1	k_2	k_3
Trunk	k_1	k_4	k_3
Neck	k_5	k_5	k_5
Head	k_5	k_5	k_5
R. arm	k_6	k_6	k_6
R. forearm	k_7	k_7	k_7
R. hand	k_7	k_7	k_7
L. arm	k_8	k_8	k_8
L. forearm	k_9	k_9	k_9
L. hand	k_9	k_9	k_9
R. leg	k_{10}	k_{10}	k_{10}
R. shank	k_{11}	k_{11}	k_{11}
R. foot	k_{12}	k_{13}	$\frac{1}{2}(k_{12}+k_{13})$
R. toes	k_{12}	k_{13}	$\frac{1}{2}(k_{12}+k_{13})$
L. leg	k_{14}	k_{14}	k_{14}
L. shank	k_{15}	k_{15}	k_{15}
L. foot	k_{16}	k_{17}	$\frac{1}{2}(k_{16}+k_{17})$
L. toes	k_{16}	k_{17}	$\frac{1}{2}(k_{16}+k_{17})$

Scaling is carried out by solving a standard nonlinear least-squares problem, where the design variables are the independent position coordinates of the model, together with the independent scale factors, and the objective function is the squared norm of the measurement residual:

$$\min_{\mathbf{z},\mathbf{k}} f(\mathbf{z},\mathbf{k}) = [\mathbf{m}_c - \mathbf{h}_s(\mathbf{x},\mathbf{k})]^{\mathrm{T}} [\mathbf{m}_c - \mathbf{h}_s(\mathbf{x},\mathbf{k})]. \tag{6}$$

This optimization problem is essentially the one proposed by Lu and O'Connor [13], to which the independent scale factors are added here as additional design variables. The residual is defined as the difference between the measured marker coordinates $\mathbf{m}_c$ and their estimated or *virtual* counterparts, provided by the observation function $\mathbf{h}_s$ of the EKF, which is augmented here in order to include the scale factors $\mathbf{k}$ as additional input arguments. Likewise, the Jacobian matrix of the measurement residual mostly coincides with that of the observation function, with additional columns corresponding to the derivatives with respect to the components of $\mathbf{k}$. Having the exact Jacobian matrix enables us to solve the optimization in a very robust and efficient way by using the Levenberg–Marquardt algorithm [43, 44].

2.2.4 Marker labeling

The motion-capture system provides a set of 3D marker coordinates every 10 ms. Ideally, each dataset should only contain the coordinates of the n_m markers attached to the subject, with the markers always sorted in the same order. This would allow for properly identifying or *labeling* the markers, so they can be assembled correctly within the measurement vector $\mathbf{m}_c$. However, it is very frequent to lose markers due to occlusions, or to have spurious reflections appear as additional markers. Moreover, the markers are not guaranteed to be sorted in any particular order. In traditional, postprocessing-based methods, these problems are addressed after the capture process, in a manual or semiautomatic way. However, in the case of a real-time application, where it is not possible to perform any manual data correction, these problems must be addressed on-the-fly.

In the present work, this is solved using the same method described in [18], which is based on two separate algorithms: the first one performs an initial marker identification and, in the subsequent time steps, the second one takes advantage of the observation function to keep labeling markers on-the-fly.

Initial labeling is approached by assuming that the subject is standing close to a known reference pose, which enables the identification of the measured markers based on their relative spatial locations. Once the experimental markers are identified, the vector of sorted coordinates $\mathbf{m}_c$ is used as an input to solve the scaling problem described in Eq. (6). If the objective function reaches a sufficiently small value after the optimization, and all the resulting scale factors are positive, the solution is regarded as valid. On the contrary, if the posture of the subject is too far from the reference, markers are likely bo be mislabeled by the initial sorting algorithm, thus leading to a large measurement residual and/or negative scaling factors after the optimization. In this case, the process is repeated at every time step in a loop, until the subject adopts an adequate posture. In the real-time application described in Sect. 3, each initial sorting and optimization process takes less than 2.5 ms, so initialization is perceived as instantaneous as soon as the reference pose is reached.

After a valid solution has been achieved, the optimal scale factors are stored and permanently applied to the model. However, before starting the EKF, the local position vectors $\bar{\mathbf{m}}_i$ of the virtual markers require a further adjustment. For the current pose of the already scaled model, their components are corrected to make the virtual markers match the measured ones. Performing such correction decreases the measurement error or *innovation* while running the Kalman filter. This allows us to reduce the marker search radius used in the continuous labeling algorithm described below, thus minimizing the risk of marker swapping.

Once the model is correctly scaled, and the local marker coordinates are adjusted, the recursive predictor–corrector algorithm of the EKF can start, using the independent positions $\mathbf{z}$ obtained from the optimization as a starting point (setting the initial velocities and accelerations to zero). As with the scaling algorithm, the EKF also requires an ordered set of experimental marker coordinates $\mathbf{m}_c$ to work. However, the initial labeling method described above can no longer be used, since identifying the markers according to their relative positions is only possible for certain poses of the subject. Conveniently, when the filter is running, the observation function provides a set of estimated marker positions, which can be exploited to identify the measured markers based on their proximity to them, through a nearest-neighbor search. This can be done very efficiently by computing a matrix of squared crossdistances between the real and the virtual marker set, as explained in [18]. If a virtual marker does not have any measured one within a predefined search radius, it is regarded as lost for that time step, so it is not used in the EKF corrector. If the algorithm detects a severe marker loss, or the innovation of the EKF becomes too large, the process must be rebooted. This means that the pose-based labeling and optimization loop must be carried out again, but this time only the positions $\mathbf{z}$ are used as design variables, since the scaling factors are already determined.

2.3 Inverse dynamics

Apart from the kinematics provided by the EKF, solving the inverse-dynamics problem requires knowing the body-segment parameters and the ground reactions, along with a set of equations of motion. These elements will be explained in detail in this section.

Table 3 Unscaled body-segment parameters in SI units (kg, m, kg·m^2)

Segment	Mass	$\bar{x}_G$	$\bar{y}_G$	$\bar{z}_G$	$\bar{\mathbb{I}}_{xx}$	$\bar{\mathbb{I}}_{yy}$	$\bar{\mathbb{I}}_{zz}$
Pelvis	10.1285	0.0000	0.0000	−0.1841	0.138007	0.071918	0.135865
Trunk	17.7962	0.0000	0.0000	0.0884	0.150064	0.214446	0.123759
Neck	0.7568	0.0000	0.0000	0.0542	0.001509	0.001211	0.001211
Head	3.0250	0.0453	0.0000	0.0178	0.013815	0.012666	0.011455
R. arm	1.4208	0.0047	0.0075	−0.1241	0.010349	0.011026	0.002112
R. forearm	1.0000	0.0028	−0.0028	−0.0983	0.004374	0.004069	0.000675
R. hand	0.3488	0.0000	0.0000	−0.0879	0.000942	0.000930	0.000427
L. arm	1.4208	0.0047	−0.0075	−0.1241	0.010349	0.011026	0.002112
L. forearm	1.0000	0.0028	0.0028	−0.0983	0.004374	0.004069	0.000675
L. hand	0.3488	0.0000	0.0000	−0.0879	0.000942	0.000930	0.000427
R. leg	7.0208	−0.0161	−0.0125	−0.1661	0.088659	0.094880	0.023717
R. shank	2.5863	−0.0173	−0.0025	−0.1482	0.026500	0.026500	0.003381
R. foot	0.7796	0.0409	−0.0043	−0.0306	0.000517	0.002316	0.002176
R. toes	0.0635	0.0319	0.0017	−0.0046	0.000034	0.000016	0.000051
L. foot	7.0208	−0.0161	0.0125	−0.1661	0.088659	0.094880	0.023717
L. leg	2.5863	−0.0173	0.0025	−0.1482	0.026500	0.026500	0.003381
L. shank	0.7796	0.0409	0.0043	−0.0306	0.000517	0.002316	0.002176
L. toes	0.0635	0.0319	−0.0017	−0.0046	0.000034	0.000016	0.000051

2.3.1 Body-segment parameters

The body-segment parameters (BSP) comprise the mass of each body segment, the location of its center of gravity (CoG), and its tensor of inertia. These parameters have a significant effect in the inverse dynamics [45], however, since it is very difficult to obtain their exact values, they are normally scaled using reference values taken from the literature. There exist methods to obtain these values more accurately, such as performing some previous measurements and exercises [12], using optimization techniques [46, 47], or even resorting to clinical measurements [48]. However, in this work, the simpler scaling method is chosen instead, since it allows for obtaining results immediately after placing the markers on the subject, which poses an interesting advantage for real-time biofeedback applications.

The reference values used here are a combination of those published by Silva and Ambrósio [36] and Dumas et al. [49]. The latter has more recent data, so its values are used whenever possible. However, they used a kinematic model in which the lumbar joint was placed at a different location, so the pelvis and trunk data must be taken from the first cited paper. In order to match the geometry of the 3D graphics model defined in Table 1, the values from the referred papers are scaled, to obtain the reference body-segment parameters displayed in Table 3.

2.3.2 Ground-reaction forces and torques

Force plates measure the forces and moments produced by foot–ground interactions. Each plate provides the six components of the corresponding ground reaction, in the form of a 3D wrench located at its center. In the case of a single support, force plates are not strictly necessary, since inverse dynamics can provide the six components of the net external reaction.

However, in the frequent case of having more than one external contact, they become essential to eliminate the resulting indeterminacy [50]. In order to correctly transfer the reactions to the feet, the location and orientation of the force plates within the motion capture volume must be accurately determined.

During the motion-capture session, it is important to make sure that, whenever a foot steps on a plate, it stays completely within its bounds, and that two feet never step on the same plate at the same time. In the standard motion-capture workflow, when the data is postprocessed, it is necessary to determine which foot stepped on each plate, in order to correctly assign the measured reactions during the inverse dynamics analysis. In real-time biofeedback applications, however, this information is not necessarily known in advance, so an algorithm to assign the force-plate reactions to the feet in real time becomes necessary. The method used in this work performs the following steps:

– Calculate the absolute position of the centers of gravity of both feet.
– Check which force plate is bearing the highest load.
– Locate the center of pressure of said plate in global coordinates.
– Determine which foot has its center of gravity closest to this point.
– Assign that foot to the most loaded plate.
– Assign the remaining foot to the least loaded plate.

The centers of pressure are initially set to the geometric centers of the force plates. They are only recalculated when the vertical force exceeds a 1.0 N threshold, which is slightly above the amplitude of the noise present in the signal. Otherwise, if the plate is unloaded or bearing a very small load, the noisy values of forces and moments can lead to random locations of the center of pressure, usually outside the bounds of the force plate.

Assigning the most loaded plate first avoids problems when one of the feet is airborne and far away from the ground. In such a situation, the center of gravity of the standing foot is the closest one to both centers of pressure, so the results can be wrong depending on the assignment order. Once the plates have been assigned, their measured forces and moments can be translated to the center of gravity of the corresponding foot, and then added to the Cartesian forces vector $\bar{\mathbf{Q}}$, as will be explained in the next section.

2.3.3 Equations of motion

The equations of motion for the inverse-dynamics problem are stated in the same set of independent coordinates $\mathbf{z}$ used for the EKF. In order to assemble the dynamic terms in a systematic way, an intermediate set of dependent Cartesian velocities $\mathbf{v}_i$ is defined for each body i. It contains the velocity of its center of mass in global coordinates, along with its angular velocity in local coordinates:

$$\mathbf{v}_i = \begin{bmatrix} \dot{\mathbf{r}}_{G_i} \\ \bar{\boldsymbol{\omega}}_i \end{bmatrix} \qquad i = 1, \ldots, n_b. \tag{7}$$

This is similar to the set of Cartesian velocities used in the efficient semirecursive formulation described in [37]. In that work, the authors used the velocity of the point of body i that instantaneously coincides with the global origin, and the angular velocity of the same body in global coordinates, with the goal of being able to recursively accumulate the dynamic terms. In this work, however, they are defined in such a way that they lead to a constant block-diagonal mass matrix. This results in simpler and easier to linearize expressions for the dynamic terms, which has other advantages, as mentioned in Sect. 2.1.

According to the previous definition of the Cartesian velocities of body segment i, its corresponding constant mass matrix is defined as follows:

$$\bar{\mathbf{M}}_i = \begin{bmatrix} m_i\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbb{I}}_{G_i} \end{bmatrix} \qquad i = 1, \ldots, n_b, \tag{8}$$

where m_i is the total mass of the segment, $\mathbf{I}$ is the 3×3 identity matrix, and $\bar{\mathbb{I}}_{G_i}$ is the tensor of inertia of the segment in local axes, defined with respect to its center of gravity. When the products of inertia are zero, this matrix becomes purely diagonal. The forces and torques associated to these Cartesian velocities have the following form:

$$\bar{\mathbf{Q}}_i = \begin{bmatrix} \mathbf{f}_i + m_i\mathbf{g} \\ \bar{\mathbf{n}}_{G_i} - \bar{\boldsymbol{\omega}}_i \times \bar{\mathbb{I}}_{G_i}\bar{\boldsymbol{\omega}}_i \end{bmatrix} \qquad i = 1, \ldots, n_b, \tag{9}$$

where $\bar{\boldsymbol{\omega}}_i$ is the angular velocity of body i, expressed in its local reference frame, and $\mathbf{g}$ is the acceleration of gravity. The vector $\mathbf{f}_i$ corresponds to the net external force applied to body i (excluding the weight), and $\bar{\mathbf{n}}_{G_i}$ is the net external moment, calculated with respect to the center of gravity, and expressed in local coordinates. These external forces and moments include the reactions and motor torques at the joints, in addition to the ground reactions in the case of the feet.

If the velocities of all bodies are assembled into a vector $\mathbf{v}$, and the same is done to the individual dynamic terms $\bar{\mathbf{M}}_i$ and $\bar{\mathbf{Q}}_i$, thus obtaining the mass matrix $\bar{\mathbf{M}}$ and force vector $\bar{\mathbf{Q}}$ of the whole system, the Newton–Euler equations of motion can be written in matrix form as:

$$\bar{\mathbf{M}}\dot{\mathbf{v}} = \bar{\mathbf{Q}}. \tag{10}$$

In order to obtain the equations of motion in independent coordinates, the first step is to apply the principle of virtual power to these equations. This principle states that, for any given set of virtual velocities $\tilde{\mathbf{v}}$, provided that they are compatible with the kinematical constrains of the multibody system, the corresponding virtual power of all the forces acting on the system (including D'Alembert's inertia forces), is always null [51]. It can be expressed in matrix form as follows:

$$\tilde{\mathbf{v}}^{\mathsf{T}}\left(\bar{\mathbf{M}}\dot{\mathbf{v}} - \bar{\mathbf{Q}}\right) = 0. \tag{11}$$

For any allowed motion of the system, the joint reactions do not produce any power, since the constrained degrees of freedom have no relative motion. Therefore, they can be removed from $\bar{\mathbf{Q}}$ in this equation. This means that all the unknown $\mathbf{f}_i$ and $\bar{\mathbf{n}}_{G_i}$ terms corresponding to joint reactions in Eq. (9) can be eliminated, leaving only those corresponding to motor forces and torques.

The next step for obtaining the equations of motion in independent coordinates is to define a position-dependent velocity-transformation matrix $\mathbf{R}$ [51], such that:

$$\mathbf{v} = \mathbf{R}\dot{\mathbf{z}}. \tag{12}$$

For scleronomous systems, as is the case of the multibody model of the human body used in this work, the dependent accelerations are directly obtained by differentiating the previous equation in time:

$$\dot{\mathbf{v}} = \mathbf{R}\ddot{\mathbf{z}} + \dot{\mathbf{R}}\dot{\mathbf{z}}. \tag{13}$$

This transformation can be applied to both the real and virtual velocities in Eq. (11), yielding:

$$\dot{\mathbf{z}}^{*\mathsf{T}}\mathbf{R}^{\mathsf{T}}\left[\bar{\mathbf{M}}\mathbf{R}\ddot{\mathbf{z}}-\left(\bar{\mathbf{Q}}-\bar{\mathbf{M}}\dot{\mathbf{R}}\dot{\mathbf{z}}\right)\right]=0. \tag{14}$$

As opposed to what happened in Eq. (11), the virtual velocities $\dot{\mathbf{z}}^*$ appearing in this equation are independent, which means they can take any arbitrary value. Since Eq. (14) must be fulfilled for any possible set of virtual velocities, the term multiplying them must be equal to zero. This leads to the desired equations of motion of the system in independent coordinates:

$$\mathbf{R}^{\mathsf{T}}\bar{\mathbf{M}}\mathbf{R}\ddot{\mathbf{z}}=\mathbf{R}^{\mathsf{T}}\left(\bar{\mathbf{Q}}-\bar{\mathbf{M}}\dot{\mathbf{R}}\dot{\mathbf{z}}\right), \tag{15}$$

which can be rewritten as:

$$\mathbf{M}\ddot{\mathbf{z}}=\mathbf{Q}, \tag{16}$$

where $\mathbf{M}$ and $\mathbf{Q}$ are the mass matrix and generalized forces vector:

$$\mathbf{M}=\mathbf{R}^{\mathsf{T}}\bar{\mathbf{M}}\mathbf{R} \qquad \mathbf{Q}=\mathbf{R}^{\mathsf{T}}\left(\bar{\mathbf{Q}}-\bar{\mathbf{M}}\dot{\mathbf{R}}\dot{\mathbf{z}}\right). \tag{17}$$

As explained above, the joint reactions do not affect the result of this equation, so their $\mathbf{f}_i$ and $\bar{\mathbf{n}}_{G_i}$ terms can be omitted when using Eq. (9) to assemble $\bar{\mathbf{Q}}$.

2.3.4 Inverse dynamics

In order to solve the inverse-dynamics problem, the generalized forces vector from Eq. (16) must be split into two separate terms, as follows:

$$\mathbf{M}\ddot{\mathbf{z}}=\mathbf{Q}_0+\mathbf{Q}_e. \tag{18}$$

The vector $\mathbf{Q}_0$ includes only the known part of the generalized forces, i.e., those due to the measured ground reactions, together with the gravitational and velocity-dependent inertia forces. It is assembled as described in Eqs. (9) and (17), with the feet being the only bodies with nonzero $\mathbf{f}_i$ and $\bar{\mathbf{n}}_{G_i}$ terms, due to the ground reactions. All remaining external forces and torques have been now translated into the vector of unknown motor efforts $\mathbf{Q}_e$. As mentioned in Sect. 2.3.2, prior to being incorporated into $\mathbf{Q}_0$, the ground reactions measured by the force plates must be transformed to make them consistent with the definition of the Cartesian velocities $\mathbf{v}$. This implies translating the reaction moments from the force plate centers to the centers of gravity of the feet, and transforming them to the corresponding local axes.

The generalized forces included in $\mathbf{Q}_e$ are acting directly on the independent coordinates $\mathbf{z}$, which makes them difficult to interpret. Therefore, it is better to establish a more meaningful set of external efforts $\mathbf{T}$, and map them into the independent coordinates by using a position-dependent input matrix:

$$\mathbf{Q}_e=\mathbf{B}^{\mathsf{T}}\mathbf{T}. \tag{19}$$

The vector of motor efforts $\mathbf{T}$ contains the motor torques driving all the joints, expressed in the local axes of the corresponding proximal bodies. The torque at the base of the neck only has local x and y components, due to the use of a 2-DOF universal joint, and the torques at the toes are scalar, since they correspond to revolute joints. Along with the motor torques, $\mathbf{T}$ includes a residual wrench acting at the pelvis, which is intended to absorb the inconsistency between the measured ground reactions and the net external wrench predicted by inverse dynamics. This wrench should ideally be zero, but in practice it never is, due to the many sources of error present in the process [28, 52].

To facilitate its derivation, the input matrix $\mathbf{B}$ can be interpreted as a velocity transform such that:

$$\dot{\mathbf{y}} = \mathbf{B}\dot{\mathbf{z}}, \tag{20}$$

where $\dot{\mathbf{y}}$ is a set of independent Cartesian velocities, defined consistently with the power produced by the motor efforts $\mathbf{T}$. Therefore, it is formed by the relative angular velocities at the joints, expressed in the local axes of their corresponding proximal bodies, along with the velocity of the origin of the pelvis and its angular velocity, both in global axes. The computation of matrix $\mathbf{B}$ is very efficient, since the $\dot{\mathbf{y}}$ velocities are closely related to $\mathbf{v}$, so the blocks forming matrix $\mathbf{B}$ are easily derived from terms already present in matrix $\mathbf{R}$. In the real-time software implementation described in Sect. 3, the computation of the input matrix takes an average of 23 µs per time step.

At every time step, after the independent positions, velocities, and accelerations have been provided by the EKF, and the ground reactions have been measured by the force plates, the equations of motion can be assembled, so the external efforts $\mathbf{T}$ are obtained by solving the following linear system:

$$\mathbf{B}^{\mathrm{T}}\mathbf{T} = \mathbf{M}\ddot{\mathbf{z}} - \mathbf{Q}_0. \tag{21}$$

2.4 Estimation of muscle efforts

As a first approach, muscular efforts are estimated in this work using simple static optimization. This implies that muscles are introduced in the model as plain contractile actuators, ignoring the presence of elastic tendons, and neglecting most of their physiological limitations, such as the delay produced by excitation–activation dynamics, and the effect of muscle length and contraction velocity [28, 29]. Although it is a very simplistic way of modeling muscle forces, it provides reasonable results for slow movements with small muscle elongations [29, 30]. Under these assumptions, the n_F muscle forces $\mathbf{F}$ can be estimated by solving an optimization problem:

$$\begin{aligned} \min_{\mathbf{F}} g(\mathbf{F}) &= \sum_{i=1}^{n_F} \left(\frac{F_i}{F_i^0} \right)^2, \\ \text{s.t.} \quad \mathbf{J}^{\mathrm{T}}\mathbf{F} &= \mathbf{T}, \\ 0 \le F_i &\le F_i^0, \end{aligned} \tag{22}$$

where F_i^0 is the maximum isometric force of muscle i, and $\mathbf{J}$ is the matrix of moment arms, defined such that the product $\mathbf{J}^{\mathrm{T}}\mathbf{F}$ is mechanically equivalent to the joint torques provided by the inverse dynamics. The moment arms depend on the placement of the muscle-insertion points, whose reference local coordinates are adapted from the “Gait2392” model included in OpenSim [1]. The local coordinates of the insertion points are affected by the model-scaling process described in Sect. 2.2.3 in the same way as those of joints and optical markers.

The moment-arm matrix **J** can be derived by using the same reasoning used to obtain the input matrix **B**, i.e., by understanding it as a transformation matrix that turns the Cartesian velocities $\dot{\mathbf{y}}$ into the shortening rates of the active muscle segments. As with the input matrix, most of the terms involved in the assembly of **J** are already computed at this point, since they are required for solving the inverse-dynamics problem. Therefore, obtaining the exact moment-arm matrix has very little computational cost (about 34 μs per time step), making approximated methods [23, 24] unnecessary in this case.

It must be pointed out that the joint-torque constraints are only imposed to the degrees of freedom that are considered as being actually actuated by muscles. For instance, the knees are modeled here as spherical joints for the sake of simplicity, so the inverse dynamics will output relatively large abduction–adduction torques at them. However, it is obvious that these torques are indeed joint reactions, essentially produced by ligaments and surface contacts, so it does not make sense to enforce the muscles to provide them. To avoid parasitic work by the reaction forces, the knees should be modeled using a more physiologically accurate model, either by using a revolute joint with an optimized rotation axis [47], or a macro joint replicating the actual kinematics [25]. These model improvements would not have a significant impact on computational efficiency, so using them should not be a problem in a real-time application.

The optimization problem stated in Eq. (22) can be solved at a very small computational cost, because it is a standard quadratic programming problem with linear equality and inequality constraints, for which there exist several well-known solvers [43]. In order for the problem to be addressed by a standard quadratic programming algorithm, the objective function g can be rewritten in the typical quadratic form:

$$g\left(\mathbf{F}\right) = \frac{1}{2}\mathbf{F}^{\mathsf{T}}\mathbf{C}\mathbf{F} + \mathbf{D}^{\mathsf{T}}\mathbf{F}, \tag{23}$$

where the vector **D** is not present in this case, and the matrix **C** is a constant diagonal matrix such that:

$$C_{ii} = \left(\frac{\sqrt{2}}{F_i^0}\right)^2 \qquad i = 1, \ldots, n_F. \tag{24}$$

In this work, a modified version of the interior-point algorithm proposed by Mehrotra [43, 53] is implemented. The original algorithm has been tuned for this specific problem, in order to take advantage of the fact that there is no **D** vector and **C** is diagonal. This makes the optimization very efficient: with 86 muscles, the algorithm needs less than 0.15 ms to converge in a standard desktop PC, when the results obtained at each time step are fed as the initial guess to the next one.

In order to facilitate convergence, a set of additional external torques is added to the vector of design variables **F**, in a similar fashion to the *reserve actuators* used by OpenSim [1]. These actuators are introduced directly at the joints, so they can always provide the missing torque required to achieve convergence. To prevent the optimizer from relying on them too much, their maximum capacity is set to a small value, thus penalizing their use in the objective function. In an ideal situation, the reserve actuators should not be necessary but, if their activation is kept sufficiently low, they can help achieving convergence, while maintaining a reasonably low impact on the results.

3 Implementation and testing

The laboratory has a motion-capture system composed by 18 OptiTrack Flex 3 infrared cameras (OptiTrack, Corvallis OR, USA). These cameras have a resolution of 640x480 pixels, with a refresh rate of 100 Hz. In order to measure the ground reactions, two AMTI AccuGait force plates (Advanced Mechanical Technology Inc., Watertown MA, USA) are embedded into a ground platform.

3.1 Software architecture

The software for real-time motion capture, analysis, and visualization is programmed in C++. The graphical user interface (GUI), 3D graphics, multithreading and serial-port-related tasks are implemented using the Qt multiplatform framework (The Qt Company, Espoo, Finland). The Levenberg–Marquardt algorithm for model scaling, all Kalman filter computations, along with the inverse-dynamics and muscle-optimization solvers, are all implemented using the Eigen C++ template library for linear algebra [44], which is in turn configured to take advantage of the Intel® MKL libraries (Intel Corporation, Santa Clara, CA, USA).

Since there are different hardware devices involved, the software has an asynchronous multithreaded structure. The 3D coordinates of the optical markers are acquired by a separate thread using the C library provided by the manufacturer of the cameras. This thread launches a callback function every 10 ms, sending the raw marker coordinates to the main thread. In a parallel subprocess, the force plates are read through a serial-port interface. They also send the data at a 100 Hz rate, with another callback function being executed every time a data packet is received at the serial port. Since the ADC of the force plates does not provide an input for hardware clock synchronization, the data from force plates and cameras is not completely synchronized at a hardware level.

In the main thread, there are two possible states. When the program starts, the system is uninitialized, since the markers are not yet labeled and the model is unscaled. The program loops until n_m markers are found, and, when this happens, it tries to label them by using the pose-based method described in Sect. 2.2.4. Then, the scaling problem is run, and the value of the objective function is checked. If the error is below a certain threshold, it is assumed that the labeling is correct, so the EKF can start. Otherwise, the loop continues until a valid solution is found.

When the program is running the EKF, the sequence is as follows: as soon as a new dataset arrives from the cameras, the markers are labeled on-the-fly using the nearest-neighbor approach. Then, these ordered sensor measurements are used, together with the latest estimated markers and the EKF gain matrix, to correct the state vector $\mathbf{x}$. Once the updated kinematics are available, the inverse-dynamics problem is solved, using the positions, velocities, and accelerations provided by the EKF, and the most current force-plate measurements available as inputs. When the motor torques $\mathbf{T}$ are ready, the muscle optimization is carried out, leading to the estimated muscle forces $\mathbf{F}$. In order to visualize the individual activations, each muscle is assigned a color, which is mapped from blue when the muscle is deactivated, to bright red when it is fully activated. Then, this color information, along with the transformation matrices of all rigid bodies, is sent to the 3D graphics engine, which takes care of the rendering in an asynchronous way. At this point, the EKF prepares to receive the next incoming dataset from the cameras: by using the state-propagation matrix, it calculates the next *a priori* estimation of the states that, in turn, provides the estimated markers by evaluating the observation function, along with its Jacobian matrix, necessary to compute the new EKF gain matrix.

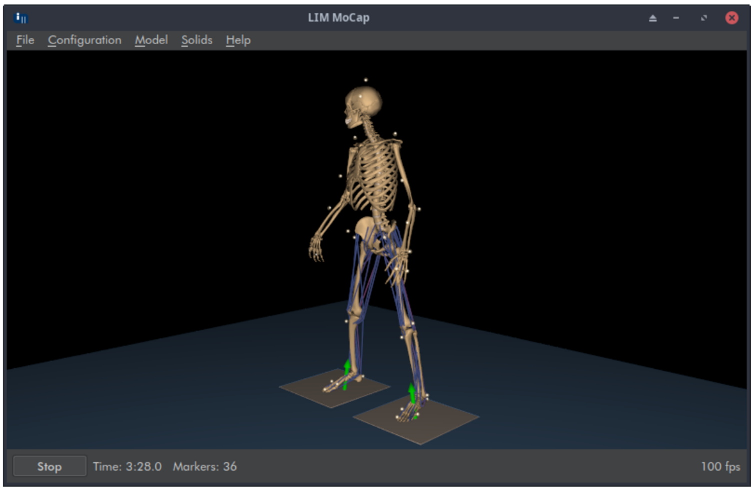


Fig. 3 Real-time motion capture software GUI

Figure 3 shows a screenshot of the GUI of the software, which displays in real time the musculoskeletal model, with the measured and estimated markers, along with the ground-reaction forces and the muscle activations.

3.2 Experimental results

The tests were carried out using the biomechanical model described in Sect. 2.1, with 36 optical markers and 86 muscles (43 at each leg). With this model size, an AMD Ryzen™ 7 3700X processor can run the algorithm in real time with an ample margin, since it needs about 0.8 ms per time step to perform all the calculations (marker labeling, EKF, inverse dynamics, and muscle optimization), and the cameras and force plates provide new data every 10 ms.

There are several sources of latency in the system: the IR cameras have to identify the markers and perform the triangulation, then the data is processed through all the algorithm steps and, finally, the 3D graphics output is generated and sent to the screen. However, the total lag of the 3D visualization is still small enough to be almost imperceptible, and the application has a very good real-time feedback. A demonstration of the application running in a motion capture laboratory is shown in Fig. 4.

The algorithm has proven fast enough to enable real-time musculoskeletal analysis, but the computational cost is not the only concern when processing a signal on-the-fly. The raw position data provided by the motion-capture system contains a significant amount of high-frequency noise, which needs to be filtered out of the signal prior to calculating velocities and accelerations. In the traditional postprocessing approach, the positions are passed through a discrete-time low-pass filter first, and velocities and accelerations are then obtained by using finite differences. In the EKF, both filtering and time differentiation occur within the predictor–corrector algorithm. The former method has the advantage that the entire time history is available at the time of processing, which allows for an additional backwards filter pass that compensates for the phase delay [12]. On the contrary, a real-time filter is *causal*, i.e., its output depends only on past and current inputs, so the delay is completely unavoidable [23, 24, 54]. The EKF, in this particular application where the system

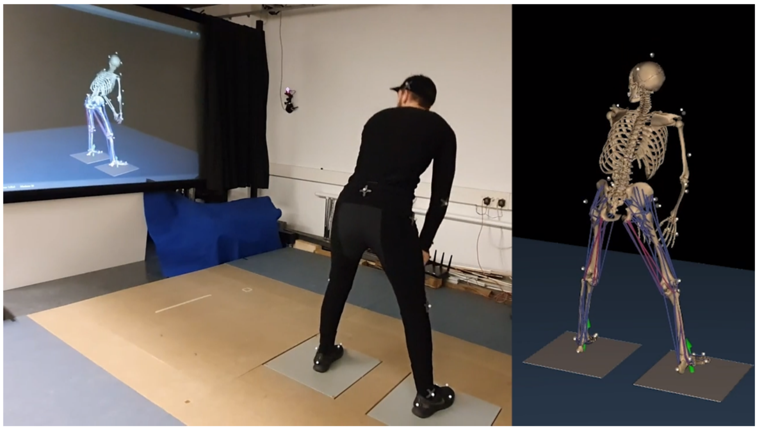


Fig. 4 Real-time demonstration of the software

has no inputs and everything is driven by position sensors, is not free from this problem, so its impact in the results must be studied.

The experiment carried out to study such effects consisted of an unloaded squat exercise, performed by a 1.65 m, 56.5 kg young female, who gave written informed consent for her participation. In this movement, the torque constraints are violated when the joint flexion angles reach high values, as muscle wrapping has not yet been accounted for in the model, so muscle-moment arms can be very inaccurate in certain postures. However, this movement is interesting from the point of view of inverse dynamics, since it involves large joint-motion ranges and produces significant joint torques. Thus, the experiment focuses primarily on the inverse dynamics, rather than the muscular efforts since, once the joint torques obtained from the real-time algorithm are proven to be correct, any further muscular analysis results will be equivalent, regardless of how the torques were obtained.

In order to obtain a lag-free reference solution, the raw sensor data from the real-time experiment was also processed through the original EKF described in [18]. Then, as was done in the referred paper, the positions obtained from the second-order EKF were filtered using a forward–backward second-order Butterworth filter, before computing the velocities and accelerations by using central differences. The reference inverse dynamics solution was finally obtained by applying the same equations of motion used in the real-time method, described in Sect. 2.3.3. In the previous work [18], the accelerations obtained from the optical system were compared to those obtained by direct measurement, in order to determine the EKF and Butterworth filter settings providing the best match. The best results were obtained for a sensor variance of 10^{-6} m^2, a process noise variance of 1.0 rad^2/s^4 (or m^2/s^4, depending on the coordinate), and a Butterworth filter cutoff frequency of 20 Hz, so these were the values used here to obtain the reference solution.

It must be pointed out that two passes of a second-order Butterworth filter with a cutoff frequency f_c have the equivalent effect of a single-pass fourth-order filter, with a lower effective cutoff frequency f_e. This frequency is obtained by squaring the gain function of a single-pass second-order Butterworth filter, and calculating at which frequency the 3 dB attenuation occurs:

$$f_e = \left(\sqrt{2} - 1\right)^{\frac{1}{4}} f_c. \tag{25}$$

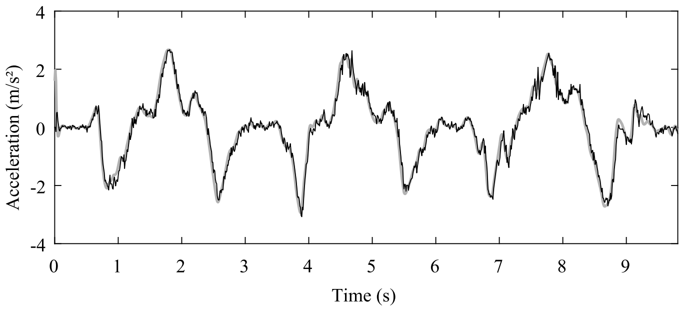


Fig. 5 Vertical acceleration of the lumbar joint: 2nd-order EKF + BW (gray) vs. 3rd-order EKF (black)

According to this expression, two passes of a 20-Hz filter result in an effective cutoff frequency of 16.04 Hz.

In the EKF, the variables most affected by the delay are the accelerations. Figure 5 displays the comparison of the reference vertical acceleration of the lumbar joint to the same magnitude obtained by the third-order EKF. Although the process noise variance has the same units in both EKFs, its physical meaning is different in each filter: in the second-order filter, the process noise represents the acceleration value during each time step whereas, in the third-order one, it represents the acceleration *increment*. Therefore, the parameters of the third-order EKF required some adjusting to better match the reference solution. In order to obtain more meaningful values, the sensor-noise variance σ_m^2 has been set in this work to 10^{-4} m^2, which is more consistent with the observed values. For this sensor-noise variance, the process-noise variance σ_w^2 providing the best results is 625 rad^2/s^4 (or m^2/s^4).

When using these filter settings, the accelerations provided by the third-order EKF are delayed by about 10 ms. This delay is essentially the same that is obtained if the reference solution is passed through a single-pass, second-order Butterworth filter with a cutoff frequency of 16.04 Hz. If the variance of the process noise of the third-order EKF is lowered, the accelerations become smoother, but the delay quickly increases to 20 or 30 ms.

To assess the effect of the acceleration delay on the joint torques, Fig. 6 shows the comparison of flexion–extension torques at the knees. As can be seen in the plots, the torques have no noticeable delay: the RMS error between both methods is about 0.5 N m for both curves. This is due to the fact that the position-dependent gravitational forces are dominant in this movement, so the delay in inertia forces does not affect the total torques in a significant way.

4 Conclusions

In this work, a complete algorithm for real-time motion capture and musculoskeletal analysis is described, implemented, and experimentally tested. It is shown that using a third-order EKF with positions, velocities, and accelerations as state variables, along with 3D optical marker coordinates as sensors, combined with ground-reaction measurements, it is possible to reconstruct the kinematics, solve the inverse-dynamics problem, and obtain and visualize the muscle efforts in real time. The approach has been successfully implemented in a fully functional software application, with a fast and responsive visual feedback. The main improvement of the method presented here over related literature references, in which kinematics is solved by optimization methods that rely on the availability of proper marker data,

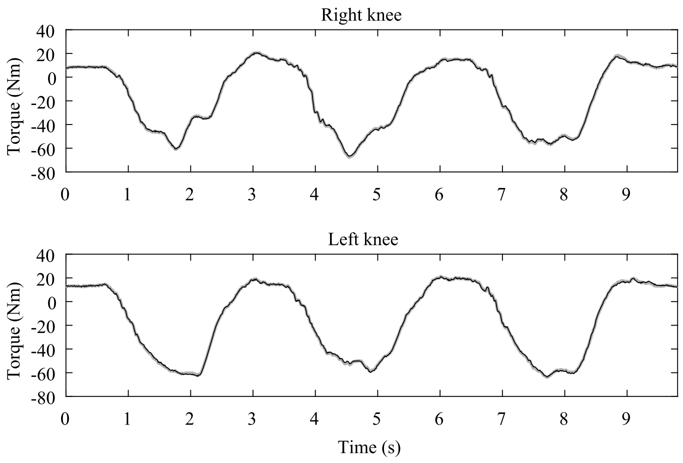


Fig. 6 Knee flexion–extension torques: 2nd-order EKF + BW (gray) vs. 3rd-order EKF (black)

is that the Kalman filter allows us to cope with marker occlusions in a very robust way, while maintaining the accuracy of real-time inverse dynamics, since accelerations are delayed about the same as when using the more common filter-and-differentiate approach.

The implementation used in this work presents some limitations. The most obvious one is the lack of muscle wrapping, which leads to inaccurate muscle-moment arms in certain postures. This can be easily solved by introducing a simple wrapping model based on dynamic via points [25, 26], which would greatly improve the accuracy at a small computational cost, or using more elaborate approaches, such as geometric surface-contact models [27]. If the application requires a complex wrapping model that cannot be run in real time, lookup tables [30] or polynomial fitting techniques [23, 24] can be used to provide approximate results more efficiently. Another relevant issue is the use of plain static optimizaton for estimating muscle forces. However, since the algorithm requires only 0.8 ms for evaluating each time step, which may be significantly reduced by using a more efficient multibody formulation [30, 37], there is still headroom for introducing a simplified physiological muscle model, as was done by Murai et al. [22], or even using a more complex static physiological-optimization approach [28, 29]. In the latter case, the inclusion of activation dynamics and tendon elasticity in the muscle model requires the integration of first-order ODEs, so the computational cost may become prohibitive for a large number of muscles. Nevertheless, there are many situations in which acceptable results can be efficiently obtained by assuming some simplifications [30]. The last important limitation is the acceleration delay. In very dynamic movements, such as a golf swing, the inertia forces may become dominant, so the delay will be translated to the joint torques and, consequently, to the muscle forces. However, being a real-time application, some kind of filtering delay is unavoidable, and 10 ms should be acceptable in most applications.

Apart from implementing an improved muscle model with wrapping, and possibly even fatigue [55], the next further development will be incorporating the equations of motion within the EKF plant model [33–35], with the force plates acting as sensors, such that the Kalman filter will include the motor torques as state variables. By including the equations of motion along with the external reactions in the plant model, the system will be no longer driven by position sensors alone, thus improving the quality of the estimation.

Author contributions U.L. wrote the manuscript text and prepared the figures. M.P. and U.L. developed the algorithm and its software implementation. M.P. and F.M. performed the experiments. J.C. supervised all the work. All authors have read and agreed to the submitted version of the manuscript.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work was financed by the Ministry of Science, Innovation and Universities of Spain, through the project PGC2018-095145-B-I00 “Estudio de la relación entre eficiencia y nivel de detalle en modelos biomecánicos del cuerpo humano”, cofinanced by the EU through the EFRD program. Moreover, F.M. would like to acknowledge the support of the Galician Government and the Ferrol Industrial Campus by means of the postdoctoral research contract 2022/CP/048.

Data Availability The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy restrictions.

Declarations

Ethical approval Ethical review and approval were waived for this study, due to the noninvasive and nondangerous character of the experiments.

Consent to participate Written informed consent was obtained from all subjects involved in the study.

Competing interests The authors declare no competing interests.

References

1. Delp, S.L., Anderson, F.C., Arnold, A.S., Loan, P., Habib, A., John, C.T., Guendelman, E., Thelen, D.G.: OpenSim: open–source software to create and analyze dynamic simulations of movement. IEEE Trans. Biomed. Eng. **54**(11), 1940–1950 (2007)
2. Damsgaard, M., Rasmussen, J., Christensen, S.T., Surma, E., de Zee, M.: Analysis of musculoskeletal systems in the AnyBody modeling system. Simul. Model. Pract. Theory **14**(8), 1100–1111 (2006)
3. Shippen, J., May, B.: BoB – biomechanics in MATLAB. In: Proccedings of 11th International Conference Biomdlore 2016 (2016)
4. Giggins, O.M., Persson, U.M., Caulfield, B.: Biofeedback in rehabilitation. J. NeuroEng. Rehabil. **10**(1), 60 (2013)
5. O’Sullivan, K., O’Sullivan, L., O’Sullivan, P., Dankaerts, W.: Investigating the effect of real-time spinal postural biofeedback on seated discomfort in people with non–specific chronic low back pain. Ergonomics **56**(8), 1315–1325 (2013)
6. Carpinella, I., Cattaneo, D., Bonora, G., Bowman, T., Martina, L., Montesano, A., Ferrarin, M.: Wearable sensor-based biofeedback training for balance and gait in Parkinson disease: a pilot randomized controlled trial. Arch. Phys. Med. Rehabil. **98**(4), 622–630.e3 (2017)
7. Cerqueira, S.M., Silva, A.F.D., Santos, C.P.: Smart vest for real-time postural biofeedback and ergonomic risk assessment. IEEE Access **8**, 107583–107592 (2020)
8. Chan, Z.Y., Zhang, J.H., Au, I.P., An, W.W., Shum, G.L., Ng, G.Y., Cheung, R.T.: Gait retraining for the reduction of injury occurrence in novice distance runners: 1–year follow–up of a randomized controlled trial. Am. J. Sports Med. **46**(2), 388–395 (2018)
9. Adesida, Y., Papi, E., McGregor, A.H.: Exploring the role of wearable technology in sport kinematics and kinetics: a systematic review. Sensors **19**(7), 1597 (2019)

10. Slade, P., Habib, A., Hicks, J.L., Delp, S.L.: An open–source and wearable system for measuring 3D human motion in real-time. IEEE Trans. Biomed. Eng. **69**(2), 678–688 (2022)
11. Vaughan, C.L., Davis, B.L., O'Connor, J.C.: Dynamics of Human Gait, 2nd edn. Kiboho Publishers, Cape Town (1992)
12. Winter, D.A.: Biomechanics and Motor Control of Human Movement. Wiley, Hoboken (2009)
13. Lu, T.W., O'Connor, J.J.: Bone position estimation from skin marker co–ordinates using global optimisation with joint constraints. J. Biomech. **32**(2), 129–134 (1999)
14. Cerveri, P., Pedotti, A., Ferrigno, G.: Robust recovery of human motion from video using Kalman filters and virtual humans. Hum. Mov. Sci. **22**(3), 377–404 (2003)
15. Halvorsen, K., Johnston, C., Back, W., Stokes, V., Lanshammar, H.: Tracking the motion of hidden segments using kinematic constraints and Kalman filtering. J. Biomech. Eng. **130**(1), 011012 (2008)
16. Senesh, M., Wolf, A.: Motion estimation using point cluster method and Kalman filter. J. Biomech. Eng. **131**(5), 051008 (2009)
17. Bonnet, V., Richard, V., Camomilla, V., Venture, G., Cappozzo, A., Dumas, R.: Joint kinematics estimation using a multi–body kinematics optimisation and an extended Kalman filter, and embedding a soft tissue artefact model. J. Biomech. **62**, 148–155 (2017)
18. Cuadrado, J., Michaud, F., Lugrís, U., Pérez Soto, M.: Using accelerometer data to tune the parameters of an extended Kalman filter for optical motion capture: preliminary application to gait analysis. Sensors **21**, 427 (2021)
19. Pizzolato, C., Reggiani, M., Modenese, L., Lloyd, D.G.: Real–time inverse kinematics and inverse dynamics for lower limb applications using OpenSim. Comput. Methods Biomech. Biomed. Eng. **20**(4), 436–445 (2017)
20. Giarmatzis, G., Zacharaki, E.I., Moustakas, K.: Real–time prediction of joint forces by motion capture and machine learning. Sensors **20**(23), 6933 (2020)
21. Sartori, M., Reggiani, M., Farina, D., Lloyd, D.G.: EMG–driven forward–dynamic estimation of muscle force and joint moment about multiple degrees of freedom in the human lower extremity. PLoS ONE **7**(12), e52618 (2012)
22. Murai, A., Kurosaki, K., Yamane, K., Nakamura, Y.: Musculoskeletal–see–through mirror: computational modeling and algorithm for whole-body muscle activity visualization in real time. Prog. Biophys. Mol. Biol. **103**(2–3), 310–317 (2010)
23. van den Bogert, A.J., Geijtenbeek, T., Even-Zohar, O., Steenbrink, F., Hardin, E.C.: A real-time system for biomechanical analysis of human movement and muscle function. Med. Biol. Eng. Comput. **51**(10), 1069–1077 (2013)
24. Stanev, D., Filip, K., Bitzas, D., Zouras, S., Giarmatzis, G., Tsaopoulos, D., Moustakas, K.: Real–time musculoskeletal kinematics and dynamics analysis using marker– and IMU-based solutions in rehabilitation. Sensors **21**(5), 1–20 (2021)
25. Delp, S., Loan, J., Hoy, M., Zajac, F., Topp, E., Rosen, J.: An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures. IEEE Trans. Biomed. Eng. **37**(8), 757–767 (1990)
26. Livet, C., Rouvier, T., Dumont, G., Pontonnier, C.: An automatic and simplified approach to muscle path modeling. J. Biomech. Eng. **144**(1), 014502 (2021)
27. Scholz, A., Sherman, M., Stavness, I., Delp, S., Kecskeméthy, A.: A fast multi–obstacle muscle wrapping method using natural geodesic variations. Multibody Syst. Dyn. **36**(2), 195–219 (2015)
28. Thelen, D.G., Anderson, F.C.: Using computed muscle control to generate forward dynamic simulations of human walking from experimental data. J. Biomech. **39**(6), 1107–1115 (2006)
29. Michaud, F., Lamas, M., Lugrís, U., Cuadrado, J.: A fair and EMG–validated comparison of recruitment criteria, musculotendon models and muscle coordination strategies, for the inverse–dynamics based optimization of muscle forces during gait. J. NeuroEng. Rehabil. **18**(1), 17 (2021)
30. Lamas, M., Mouzo, F., Michaud, F., Lugrís, U., Cuadrado, J.: Comparison of several muscle modeling alternatives for computationally intensive algorithms in human motion dynamics. Multibody Syst. Dyn. **54**(4), 415–442 (2022)
31. Quental, C., Folgado, J., Ambrósio, J., Monteiro, J.: A multibody biomechanical model of the upper limb including the shoulder girdle. Multibody Syst. Dyn. **28**(1–2), 83–108 (2012)
32. Laitenberger, M., Raison, M., Périé, D., Begon, M.: Refinement of the upper limb joint kinematics and dynamics using a subject–specific closed–loop forearm model. Multibody Syst. Dyn. **33**(4), 413–438 (2014)
33. Cuadrado, J., Dopico, D., Pérez, J.A., Pastorino, R.: Automotive observers based on multibody models and the extended Kalman filter. Multibody Syst. Dyn. **27**, 3–19 (2012)
34. Sanjurjo, E., Naya, M.A., Blanco-Claraco, J.L., Torres-Moreno, J.L., Giménez-Fernández, A.: Accuracy and efficiency comparison of various nonlinear Kalman filters applied to multibody models. Nonlinear Dyn. **88**(3), 1935–1951 (2017)

35. Pyrhönen, L., Jaiswal, S., García-Agundez, A., García Vallejo, D., Mikkola, A.: Linearization-based state–transition model for the discrete extended Kalman filter applied to multibody simulations. Multibody Syst. Dyn. **57**(1), 55–72 (2022)
36. Silva, M.P.T., Ambrósio, J.A.C.: Kinematic data consistency in the inverse dynamic analysis of biomechanical systems. Multibody Syst. Dyn. **8**(2), 219–239 (2002)
37. Cuadrado, J., Dopico, D., Gonzalez, M., Naya, M.A.: A combined penalty and recursive real-time formulation for multibody dynamics. J. Mech. Des. **126**(4), 602–608 (2004)
38. Simon, D.: Optimal State Estimation: Kalman, H_∞, and Nonlinear Approaches. Wiley, New York (2006)
39. Gibbs, B.P.: Advanced Kalman Filtering, Least–Squares and Modeling: A Practical Handbook. Wiley, New York (2011)
40. Bar-Shalom, Y., Li, X.R., Kirubarajan, T.: Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software. Wiley, New York (2004)
41. Benoit, D.L., Ramsey, D.K., Lamontagne, M., Xu, L., Wretenberg, P., Renström, P.: Effect of skin movement artifact on knee kinematics during gait and cutting motions measured in vivo. Gait Posture **24**(2), 152–164 (2006)
42. Mitsuhashi, N., Fujieda, K., Tamura, T., Kawamoto, S., Takagi, T., Okubo, K.: BodyParts3D: 3D structure database for anatomical concepts. Nucleic Acids Res. **37**(Database), D782–D785 (2009)
43. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (2006)
44. Guennebaud, G., Jacob, B., et al: Eigen v3 (2010). http://eigen.tuxfamily.org
45. Rao, G., Amarantini, D., Berton, E., Favier, D.: Influence of body segments' parameters estimation models on inverse dynamics solutions during gait. J. Biomech. **39**(8), 1531–1536 (2006)
46. Vaughan, C., Andrews, J., Hay, J.: Selection of body segment parameters by optimization methods. J. Biomech. Eng. **104**(1), 38–44 (1982)
47. Gamage, S.S.U., Lasenby, J.: New least squares solutions for estimating the average center of rotation and the axis of rotation. J. Biomech. **35**(1), 87–93 (2002)
48. Durkin, J.L., Dowling, J.J., Andrews, D.M.: The measurement of body segment inertial parameters using dual energy X-ray absorptiometry. J. Biomech. **35**(12), 1575–1580 (2002)
49. Dumas, R., Chèze, L., Verriest, J.P.: Adjustments to McConville et al. and Young et al. body segment inertial parameters. J. Biomech. **40**(3), 543–553 (2007)
50. Lugrís, U., Carlín, J., Pàmies-Vilà, R., Font-Llagunes, J.M., Cuadrado, J.: Solution methods for the double–support indeterminacy in human gait. Multibody Syst. Dyn. **30**(3), 247–263 (2013)
51. García de Jalón, J., Bayo, E.: Kinematic and Dynamic Simulation of Multibody Systems: The Real–Time Challenge. Springer, New York (2011)
52. Kuo, A.D.: A least–squares estimation approach to improving the precision of inverse dynamics computations. J. Biomech. Eng. **120**(1), 148–159 (1998)
53. Mehrotra, S.: On the implementation of a primal–dual interior point method. SIAM J. Optim. **2**(4), 575–601 (1992)
54. Oppenheim, A.V., Schafer, R.W.: Discrete–Time Signal Processing, 3rd edn. Pearson Education, Upper Saddle River (2014)
55. Michaud, F., Frey-Law, L.A., Lugrís, U., Cuadrado, L., Figueroa-Rodríguez, J., Cuadrado, J.: Applying a muscle fatigue model when optimizing load–sharing between muscles for short–duration high–intensity exercise: a preliminary study. Front. Physiol. **14**, 1167748 (2023)