ALBA: Agile library for biomedical applications

SoftwareX 31 (2025) 102188

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

ALBA: Agile library for biomedical applications

Gianluigi Crimi [a], Nicola Vanella [a], Enrico Schileo [a], Giordano Valente [a], Giulia Fraterrigo [a,b], Fulvia Taddei [a,*]

[a] Bioengineering and Computing Laboratory, IRCCS Istituto Ortopedico Rizzoli, Via di Barbiano 1/10, Bologna, Italy
[b] Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Turin, Italy

ARTICLE INFO

Keywords:
Open-source framework
Biomedical images
Data fusion
Biomechanical models
Computer-assisted medicine
Computational biomechanics

ABSTRACT

Efficient and unified software tools to manage the complex imaging and modelling workflows needed for computational biomechanics research are lacking. The Agile Library for Biomedical Applications (ALBA) is an open-source C++ versatile framework that handles various data types (e.g. 2D and 3D images, surfaces, finite element models, point clouds, motion analysis data), and offers an easily extensible platform for the rapid development of specialised applications that can manage, visualise, and manipulate biomedical data. The already available functionalities have been developed in the context of computational biomechanics, quantitative image analysis and pre-operative planning in orthopaedics. Software applications built with ALBA attracted the interest of the scientific community and are currently used, both inside and outside the original research group, in finite element modelling of bones and musculoskeletal modelling. The further ALBA adoption by other centres might increase the spread of a positive attitude towards open and reproducible research in the biomechanical community and increase the sharing of algorithms and data.

1. Motivation and significance

In all fields of computational bone biomechanics and/or orthopaedic research, the role of biomedical images is of uppermost importance. In computational (e.g. Finite Element) studies of bone mechanical behaviour medical images represent the starting data to identify shapes/structures and material properties of bone [1]. In the studies on the musculoskeletal system where the forces acting on the skeleton are investigated through musculoskeletal modelling, medical imaging data are generally used for model personalisation [2]. Similarly, biomedical images are always involved in the broad field of computational tools aimed at supporting, at different levels, the decisional process in orthopaedic surgery (e.g. pre-operative planning software to help choose the most adequate treatment option). In all these fields of research, complex workflows operating on medical images are developed/implemented. They usually comprise segmentation, visualization, registration, elaboration, and integration of medical images from different sources.

Despite several open-source libraries (VTK, ITK, etc.), or applications (3D Slicer, ITKSnap, etc.), are available, still the whole workflows presented in the literature are rarely open-source or freely available, making it difficult for other researchers to fully reproduce or reuse them. It is also generally almost impossible to test, for example, the impact a newly developed algorithm might have on a procedure without re-implementing the whole workflow, with a huge waste of effort and the real risk of endlessly reinventing the wheel. We believe that open software frameworks that facilitate the integration of different chunks of software in complex workflows can help in overcoming this problem. As a research group, we recently co-founded and we actively participate in the Open and Reproducible Musculoskeletal Imaging Research (ORMIR) community,[1] that pursues the development of open software to quantitatively analyse musculoskeletal images [3], but we have been active in the open software community for longer.

Aim of the present work is to describe the open-source framework *Agile Library for Biomedical Applications* (ALBA). ALBA leverages on open-source libraries and offers a variety of already implemented importers/exporters of biomedical data, views, interactors, and operations to work on biomedical images. The aim of ALBA is thus to offer a platform for the rapid development of specialised applications, in the field of computational biomechanics and computer aided software, to support

* Corresponding author.
E-mail addresses: gianluigi.crimi@ior.it (G. Crimi), nicola.vanella@gmail.com (N. Vanella), enrico.schileo@ior.it (E. Schileo), giordano.valente@ior.it (G. Valente), giulia.fraterrigo@ior.it (G. Fraterrigo), fulvia.taddei@ior.it (F. Taddei).

[1] https://ormir.org.

https://doi.org/10.1016/j.softx.2025.102188
Received 4 September 2024; Received in revised form 8 April 2025; Accepted 1 May 2025

orthopaedic surgery and research. ALBA can be also used as a library that offers advanced functionalities to operate on biomedical data, as was done in AlbaWrap,[2] an early-stage wrapper that exposes some ALBA functionalities in Python.

The framework is currently adopted by the Bioengineering and Computing Laboratory of IRCCS Istituto Ortopedico Rizzoli in almost all its research projects. Three examples are:

- in a recent work ALBA was used to develop a specialised software application to measure abdominal aorta calcifications on latero-lateral RX images and propose a new quantitative score (Fusaro et al., 2022). The score was evaluated against the semiquantitative most adopted one to test its repeatability on a real clinical cohort, by five different operators from four different medical specialties mimicking a real-world scenario, and provided a significant increase in repeatability, thus a decrease of the Minimum Detectable Difference (MDD) of the measurements;
- in a previous project, the ALBA framework was used to develop the *nmsBuilder* v2.0 software[3] a freely available application to process biomedical data and create subject-specific musculoskeletal models. It directly integrates with the open-source software package OpenSim[4] [4,5], one of the most used application to develop models of musculoskeletal structures and create dynamic simulations of movement. *nmsBuilder* v2.0 has received considerable attention (the relative publication [6] has to date 85 citations);
- the ALBA framework was at the basis of the recent latest release of *Bonemat,*[5] a freely available application to map material mechanical properties from CT data onto finite element models. *Bonemat* supports the import/export from many commercial Finite Element (FE) solvers and is frequently used in the biomechanical scientific community. The application is widely known and adopted (the original relative publications [7,8] have to date >500 citations, but also the new release described in [9] already obtained several citations).

2. Software description

The aim of ALBA is to provide a full framework for the rapid development and publication of applications in the biomedical field. The framework package contains installer scripts, build server configuration, an application example, and all the necessary components to start building applications. ALBA framework was developed to reduce the effort required for a programmer to obtain a fully working and usable application. To obtain this goal, a lot of features are already part of the framework library and in some cases the programmer can just select the desired components from the ALBA toolbox. When a function is instead not available, the programmer can concentrate on the development of the specific function having available the support of importer/exporters, views, operations, interactors and visual pipes.

The ALBA framework is written in C++. It uses CMake for project management and Visual Studio as compiler. CMake scripts are part of the framework. The build of the framework can be automated via Jenkins, in fact also the Jenkins jobs configuration is included. ALBA was originally derived from the Multimod Application Framework (MAF) [10] from which it was definitely forked in 2017, when MAF was dismissed, and then independently developed presenting now major new/different features that deserve an updated description. (For a detailed analysis of the differences between OpenMAF and ALBA see Supplementary Material).

The ALBA framework is based on the following open-source libraries:

- VTK[6]: the visualization toolkit is the most important library. VTK is used for data representation, storage, and filtering. On top of VTK library we have built the vtkALBA module that contains a set of filters that extend VTK functionalities. The classes on this module are extensions of VTK classes and follow vtk naming conventions;
- ITK[7]: this library is mainly used for segmentation purposes;
- Grassroot DICOM (GDCM)[8]: this library is the bridge between ALBA framework and the DICOM file format;
- wxWidget[9]: used for the creation of the Graphical User Interface (GUI) and for the logging management;
- XERCES[10]: used to read/write xml files. ALBA projects are stored as xml files in the ALBA format;
- Biomechanical Toolkit (BTK)[11]: BTK is an open-source library for biomechanical analysis used for motion analysis C3D file import.

The development of ALBA follows an Agile methodology, ensuring frequent and incremental updates to the framework. This approach is driven by the evolving requirements of ongoing research, which often demands rapid adaptations to meet the specific needs of researchers. As a result, ALBA does not follow a traditional versioning system. Instead, new features and bug fixes are released promptly after successful implementation and testing, utilizing a rolling release model, similar to that employed by certain Linux distributions like Arch.

The repository follows a structured branching strategy with two permanent branches: master and develop. This approach ensures unit testing before integrating changes into the master branch. Direct commits to both develop and master are prohibited. Instead, developers must adhere to the following workflow: (1) create feature branches from master, (2) implement changes within these isolated branches, and (3) merge modifications into develop for testing and impact assessment. All changes must successfully pass comprehensive unit tests before being merged into master. External contributors may submit pull requests for bug fixes or minor feature enhancements. However, proposals involving structural modifications or significant refactoring require prior consultation with the core development team to ensure architectural consistency and facilitate integration.

3. Software architecture

The components of the ALBA architecture are depicted in Fig. 1. In the bottom block we can find the external libraries on which ALBA is based.

Over the libraries layer, there is a middle layer that contains the components required to build working applications. The modules of this layer are:

- **VME**: a Virtual Medical Entity (VME) is an object used to retain, show, edit and store data. Each VME type represents a specific data type and contains specific information. The framework defines some standard VME types like surfaces, meshes, volumes, etc. but new types can be defined. This module contains the definition of the classes for each data type managed from the framework. VMEs can be visualized, stored, and updated. The VMEs module also contains the definition of the visual pipes. Visuals pipes make it possible to view the same VME in different ways. For example, a Volume VME can be shown through an isosurface pipe, a volume rendering, or by slicing it. VMEs are organized by a hierarchical tree data structure. In this tree data structure, each node is a VME, and the tree always

[2] https://github.com/IOR-BIC/AlbaWrap.
[3] https://ior-bic.github.io/software/nmsbuilder/.
[4] https://simtk.org/projects/opensim/.
[5] https://ior-bic.github.io/software/bonemat/index.html.
[6] https://vtk.org/.
[7] https://itk.org/.
[8] http://gdcm.sourceforge.net/.
[9] https://www.wxwidgets.org/.
[10] https://xerces.apache.org/.
[11] https://biomechanical-toolkit.github.io/.

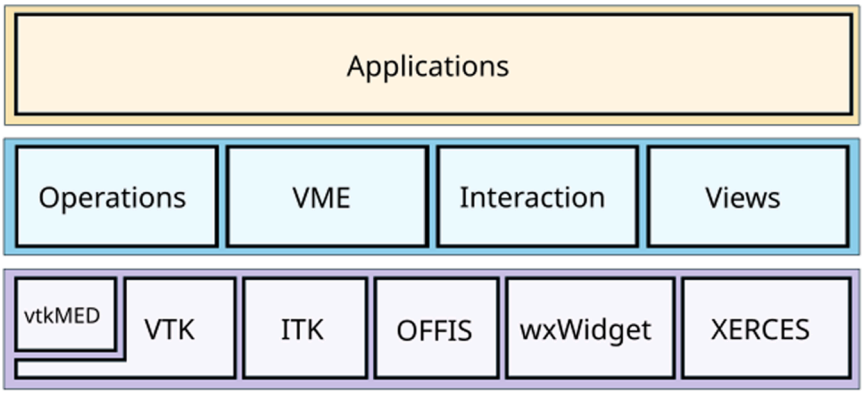

Fig. 1. The main components of the ALBA architecture.

starts with a VME Root. The hierarchy manages dependencies and the spatial pose of the VMEs. If we transform the pose of a VME all its descendants change their absolute pose to maintain the relative pose to the transformed ancestor. The rigidity of the hierarchical tree structure can be mitigated using "links" that connect different VMEs, even across separate branches. These links indicate dependency relationships and can be either mandatory or optional. The VME class follows the Observer design pattern [11] and can act both as subject and as observer ensuring consistency in the VMEs states during their manipulation. For example, a VME Meter that measures the distance between two Landmarks is notified when one of them is translated, prompting an update to the calculated distance and its graphical representation accordingly.

- **Operations:** there are three types of operations: properly defined operations, importers, and exporters. Importers and exporters serve as the interface with the rest of the world, handling input and output operations, while properly defined operations handle all other types of data manipulation. When an operation is plugged into the final application the application menus are automatically updated and the operation is placed in the right menu, depending on the type. Importers and exporters are placed under the File menu while other operations are placed on the Operations menu. All operations must extend the albaOp class and respect its structure.

  The class includes several critical methods to manage the lifecycle and effects of the operations and uses the command design pattern [11] encapsulating requests as objects and allowing more flexible control over operations. The most important methods are:
  - Accept: this method is responsible for automatically enabling the operation based on predefined criteria. It ensures that the operation is suitable for the current context before execution.
  - opRun: the opRun method initiates the execution of the operation. It serves as the primary entry point for running the operation's main logic and it is responsible for the creation of the GUI, if needed for the operation.
  - opDo: the opDo method applies the effects of the operation. This method is invoked to perform the final modifications or actions defined by the operation.
  - opUndo: the opUndo method reverses the effects of the operation. It provides a mechanism to rollback changes, ensuring that any modifications made by the operation can be undone if necessary.
- **Managers:** managers are a set of components that use the singleton design pattern [11] to expose services to other modules. These components can provide heterogeneous functionalities depending on the type of manager.
- **Interaction:** this module contains the components for the interactors, which are visual elements that can be dynamically interacted with using the mouse. We have defined both 2D and 3D interactors and there is a toolbox that can be used from views or operations to visual interact with the data. These interactors allow for actions such as crop data, select the slicing plane of a volume, translate, rotate, scale, measure distances, measure angles, etc.
- **Views:** the Views module contains the definition of the views. Views are a binding from the VMEs content to a specific representation. There are two main types of views: standard views and compound views. While standard views create a single render window, compound views can join multiple standard views and GUI components to generate a more complex view.

Mostly used VME, Operations and Views are described in following Section 4.

To illustrate module connections within the framework, we take the *Crop Volume Operation* workflow as emblematic example:

1. **User Input**: When a user selects the Volume VME via the GUI, the Managers modules dynamically identify compatible Operations by invoking their Accept() methods and enabling them in the interface.
2. **Operation Initialization**: When the user starts the crop operation from the drop-down menu, the OpRun() method creates a dedicated Interactor object and displays it in the current View.
3. **Interactive Phase**: The Interactor allows the user to define crop boundaries through direct manipulation.
4. **Execution**: Once spatial parameters are finalized, the user confirms the operation by clicking OK, triggering the execution of OpDo(). This method retrieves geometric constraints from the Interactor and applies them to generate a new cropped volume VME.

In Fig. 2 the *Crop Volume Operation* is shown applied to an MRI volume. The Crop Interactor with red handles to interactively define the region of interest is displayed in the Orthoslice View, where the MRI volume was visualised.

4. Software functionalities

ALBA applications share a standard, but customizable, GUI layout that is presented in Fig. 3.

This standard layout is composed by:

- **Menu Bar**: it contains the drop-down menus. Input/output operations are located under the menu *File;* cut & paste operations and application settings are under the menu *Edit*; the *View* menu is used to open the specific views; *Operations* contains the operations for editing or creating the data; *Wizard* is an optional menu that is enabled only if a wizard is plugged into the application; *Window* menu is used to organize open views and *Help* menu is used to access the documentation of the application.
- **Toolbar**: it contains some shortcuts to the principal functionalities that are already in the drop-down menus, the buttons used to move the camera in some default positions and the progress bar of the

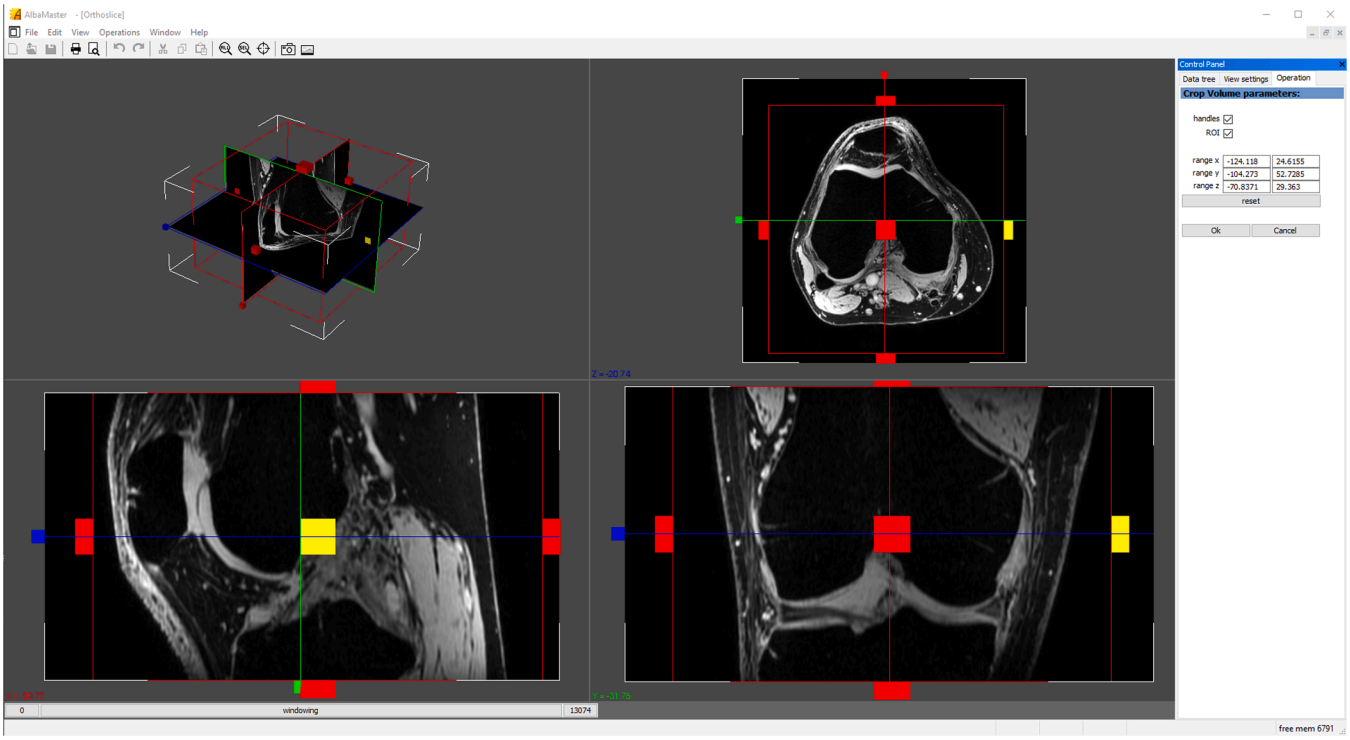

Fig. 2. The Crop Volume Operation launched on an MRI volume from an Orthoslice View.

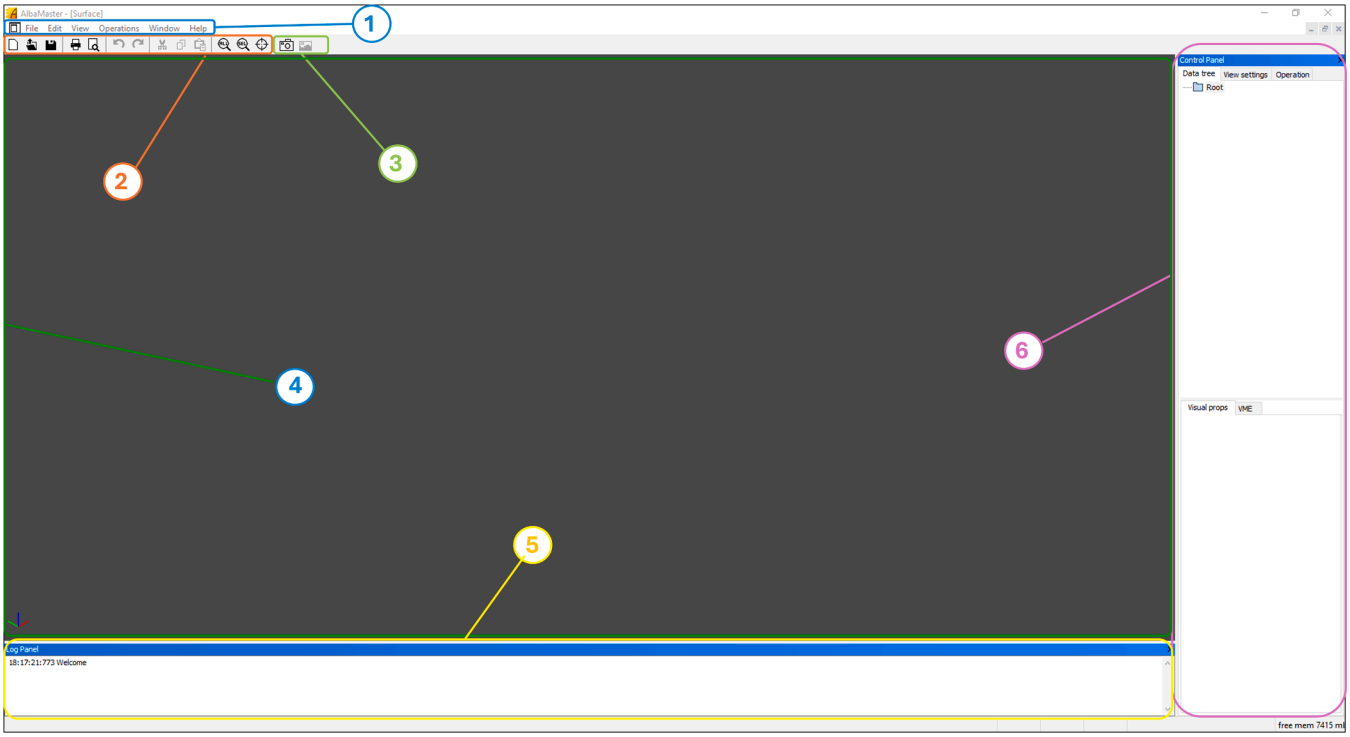

Fig. 3. The ALBA GUI with the principal components highlighted: the Menu Bar (1) with the drop-down menus; the Toolbar (2) with the shortcuts to principal functionalities; the Snapshot manager (3); in the View Area (4) a Surface View is open; the Log Panel (5); the Control Panel (6).

wizards. This toolbar can be personalized by vertical applications to speed up some recurrent user's activities.

- **Snapshot manager**: located on the toolbar, it allows you to take and manage snapshots that are saved in a hidden VME in the tree and can then be saved or printed.
- **Views Area**: all current opened views are located in this area.
- **Log Panel**: it hosts logging information used for debugging propose or for advanced users. It can be disabled.
- **Control Panel**: by default, this area contains three main tabs: *Data tree, View settings* and *Operations*. The *Data tree* tab contains all the data information for the current project, and it is split in two sub-areas. On top there is the visual representation of the Data tree,

where the user can select and show/hide VMEs in the active View. On the bottom there are the *VME* and *Visual props* tabs. The *VME* tab is used for viewing or editing the VME properties, and the *Visual props* tab is used for editing the visualization properties. The content of the *Visual props* tab changes according to the selected VME and the current view and can be used to adjust the visualization of each VME. The *View settings* tab contains the general settings of the selected view. The *Operation* tab is automatically selected when an operation is started from the menu or from the toolbar and contains the Graphical User Interface (GUI) to execute the running operation.

In Fig. 4 some of the concepts described in the previous points are shown within an example.

The ALBA framework contains a lot of components that can be plugged into the vertical applications. Some of the main features are:

- **Filters**: the *vtkALBA* module contains some useful filters that are extensions of VTK classes. Those filters are used in other modules like Operations, VMEs, and pipes. Some notable filters in this module are:
  - *vtkALBAMeshCutter*: an optimized mesh cutter that can slice FE mesh data. The input of this filter is a *vtkUnstructuredGrid* and the Output is a *vtkPolydata*.
  - *vtkALBAVolumeSlicer*: a volume slicer that manages both rectilinear grids and structured points data. This slicer works with any arbitrary plane.
- **VMEs**: the VME module contains the implementation of the data objects used by the ALBA applications. VME objects are used to hold, manage, and store data, and a store/restore mechanism is provided to store the data content in the project files. VMEs manages all the principal data types required by a Computer Aided Medicine (CAM) application but new VMEs can be defined in the application space. Some features like the metadata management and the possibility of creating links between VMEs are shared by all VMEs. Some of the most important data types are:
  - *VME Image*: used to store 2D images. The source of this VME can be an image importer, the DICOM importer (i.e. for RX or a single CT slice) or other operations like volume slicing.
  - *VME Landmark Cloud*: used to manage points sets. It is adopted in a wide range of operations like, for example, the definition of some anatomical reference points and the management of motion data. This VME works also with time-varying sets of points. Many operations related to this VME are predefined in the framework, like *Add Landmark* (to create and update landmarks), *Register Clusters* (for rigid, affine and similarity registrations of landmark clouds), etc.
  - *VME Surface*: it holds surfaces data. Surfaces can be obtained from importers, by isosurface extraction operation or by segmentation operation.
  - *VME Mesh*: it contains mesh data used in FE applications.
  - *VME Volume*: it contains a volumetric 3D dataset typically obtained from a source of 3D medical images like CT or MRI. The data contained in this dataset are stored in a regular grid of voxels.
  - *VME Refsys*: it contains an orthogonal reference system.
- **Managers:** they deal with a whole series of useful functionalities for the creation of applications and they work on a client/server basis. Among others we can mention:
  - *Help Manager*: ALBA applications can integrate a manual. If one help html file is associated with the application, ALBAparses that file and automatically associates views, operations, and VMEs with the related area in the help page.
  - *Wizards Manager*: wizards are useful to improve the learning curve of new users, or to speed up power users. The wizard manager makes it possible to automate some repetitive user tasks like selecting and showing VMEs, opening/closing views, running operations, saving the project, etc. Wizards are generated by defining and connecting single blocks.
  - *Snapshot Manager*: it is responsible for creating, viewing, storing, and printing snapshots.
- **Operations**: the *Operations* module contains the classes used to edit, import, and export data. For a list of all Operations, including Importers and Exporters, see Table 1. There are many operations in the framework. For example, we have *Crop, Resample* and *Union* operations that are used for volume editing, and a simple *Segmentation* operation is available. The *Add landmark* operation is used to create or update landmark clouds. The *Transform* operation works on spatial transformations of all VMEs that have a pose. There are some operations for *Surface editing* that can be used to smooth, split, do Boolean operations, edit normals, fill holes, etc. There are operations to make measurements in space (either 2D or 3D) and to register surfaces or landmark clouds.
- **Views**: several pre-defined views are available in the framework (see Fig. 4). Among them the most relevant ones are:
  - *Surface View*: a standard 3D view with camera management.
  - *Image View*: a standard 2D view with pan and zoom.
  - *Isosuface View*: an extension of the *Surface view* with the ability to generate and show an isosurface from volume data.
  - *Orthoslice View*: a view composed by a perspective rendering of orthogonal plane slicing, and three parallel views where one can see the slicing images for a volume in the three planes.
  - *Arbitrary slice View*: a view composed by a perspective view of a volume sliced by a free, adjustable plane, and a parallel view reporting the slicing image of the volume.
  - *Arbitrary Orthoslice View*: a view that joins the characteristics of *Orthoslice* and *Arbitrary slice* views: it contains a perspective view and three slicing views like the orthoslice, where the slicing views are orthogonal to each other but can be freely moved by the user
  - *RX-CT View*: a compound view made of eight 2D panels; two planes are used for frontal and lateral virtual RX, the other six are slicing views aligned to the Z axis.

Table 1 lists the most relevant Operations, including Import/Export ones and Views available in ALBA. In the Import/Export scheme, the Import functions that have the symmetric Export functionality are reported in bold characters, while the only Export functionalities are underlined.

5. Illustrative examples

The short videoclip provided in the Supplementary material is thought to demonstrate ALBA functionalities in the context of a finite element validation study, where model results are tested against experimental measurements of displacement and strain in human cadaver vertebrae, under physiologically admissible loads. The data showcased are taken from a validation study we recently published [12].

The key concept, which has inspired a significant part of the development of the ALBA framework, is that model validation is a fundamental prerequisite before models can be applied to any clinical data or clinical scenario. It is well acknowledged that accurate model replication of the experimental boundary conditions is key to the success of any finite element model validation experiment [13–15]. To achieve this, a single environment is needed where the user can: manage medical images and derived models; define reference systems from anatomical landmarks; perform spatial transformations and registrations (landmark or surface based, manual or automated); integrate data from medical images, engineering models, experimental signals.

The videoclip thus shows, across five scenes, how ALBA was used to interact with diagnostic images and derived models (scenes 1 and 4) and to facilitate model integration with experimental data (scenes 2, 3 and 5).

Scene 1 - Import and visualization of CT images and segmented surface

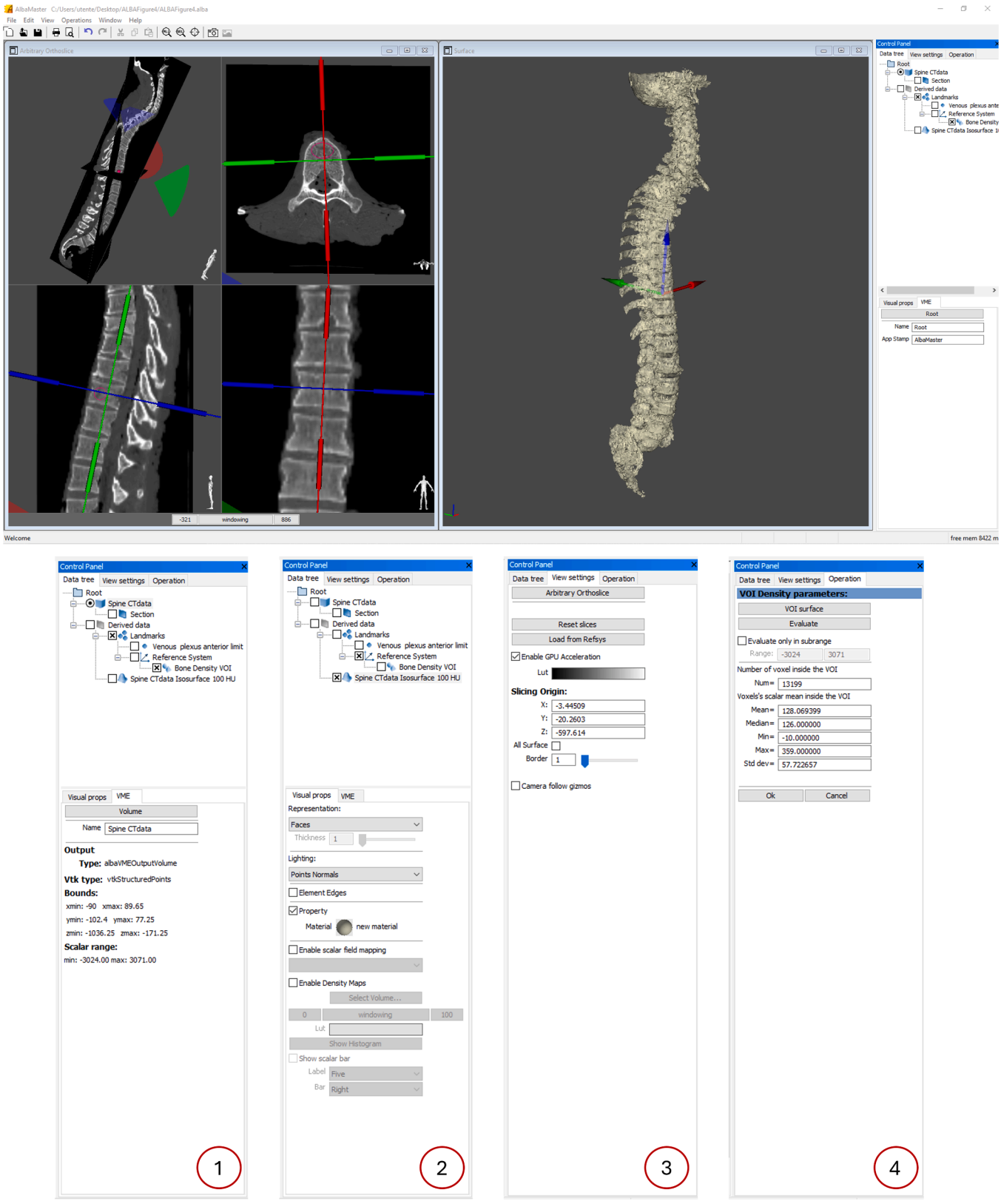

Fig. 4. Top: In the View Area an Arbitrary Orthoslice View (left) and Surface View (right) are opened. In the Arbitrary Orthoslice the CT dataset is visualised, together with the VOI region designed to measure the average density of the trabecular bone core of a selected vertebra. In the Surface View the isosurface of the spine is visualised, with the reference system defined for modelling purposes. The hierarchical data structure containing a spine CT dataset and derived entities is shown in the Control Panel. **Bottom**: some contents of the Control panel. **(1)** the VME panel of the CT volume; **(2)** the Visual Props tab for the selected VME Surface in the Surface view; **(3)** the View settings tab showing the general settings of the Arbitrary Orthoslice View; **(4)** the Operation tab to interact with the VOI Density operation.

Table 1
Most relevant functions and views available in ALBA.

Import/ Export	*Toolkits*	*Medical Images*	*Surfaces*	*FEA*	*Motion Data*	*Other Data*
	VTK	Image	**STL**	**Generic Mesh**	**Landmark**	ASCII
	VTK xml	**RAW Volume**	**PLY**	ANSYS File	C3D	Digital Image Correlation (DIC)
	ITK MetaImage	**RAW Images**	VRML	ABAQUS File	EMG	Generic Point Cloud
	ALBA Project	**DICOM**			**GRF**	Wrapped Meter
						Action Line

Views	*Standard views*	Operations	*Create*	*Modify*	*Register/Measure*
	Surface		Volume	Transform	Iterative Registration
	Isosurface		Add Landmark	Edit Surface	Clusters
	Image		Average Landmark	Boolean Surface	Surface
	3D Volume Rendering		GroupMeter	Labelise Surface	VOI Density
	Global Slice		RefSys	Fill Holes	2D Measure
	Compound Views		RefSys from View Slice	Extrude Holes	Mesh QualityInertial Tensor
	Arbitrary Slice		Prober	Surface MirrorVolume Mirror	Hausdorff Distance
	Orthoslice		Slicer	Resample Volume	Calibrate Phantom
	Arbitrary Orthoslice		Extract Isosurface	Crop Volume	Volume
	RX-CT		Parametric Surface	Crop ROI	
	Slice Blend CT		Polyline	Scale Dataset	
			Spline	Mesh Deformation	
			Labeled Volume	Make Timevarying VME	
			Skeleton	Equalize Histogram	
			Wrapped Meter	Metadata Editor	
			Freeze VME		
			Segmentation		

- Import CT images (DICOM format) of a third lumbar (L3) vertebra
- Orthoslice View (slicing through coordinate planes) of the CT images
- Import surface (STL format) and Surface View of the segmented vertebra.
- Combined Orthoslice View of CT volume and segmented surface (with cortical and trabecular compartments)

Scene 2 - Extraction of experimental conditions

- Import a different CT dataset of the same L3 vertebra imaged after cutting of the caudal endplate and posterior processes (for experimental set-up needs) and application of radiopaque markers for registration purposes
- Extract isosurface automatic function on the vertebral CT images
- Creation of a vertebral-specific reference system (Refsys from view slice function) and visualization in an Arbitrary Orthoslice View (analogue of MPR view in PACS systems)
- Creation, scaling, and positioning of a plane from experimental data (Create parametric surface function)
- Creation of virtual landmarks at experimental landmarks location (Add landmark function)

Scene 3 - From experimental setup to FE model

- Import of the isosurface with landmarks and the cutting plane into the reference system of the CT images used to generate the FE model
- Manual registration of the isosurface with landmarks and the cutting plane onto the segmented surface of the vertebra
- Cut of the segmented surface of the vertebra using the experimental cutting plane with the Clip surface function.

Scene 4 - FE Model with material properties

- Import the FE model in the ASCII format ".cdb" used by Ansys (ANSYS Inc., USA) one of the most used FE packages.
- View mesh and material properties (Young's modulus) in an Orthoslice View.

Note: on a finite element mesh created with an external program, material properties were mapped from CT images using Bonemat, a freeware created with ALBA.

Scene 5 - Registration and Visualization of Experimental data on the FE Model

- Import a patch of experimental data from Digital Image Correlation (DIC) with experimental landmarks and reference system
- Landmark-assisted manual pre-registration of the experimental patch onto the segmented surface, followed by automatic Iterative Closest Point registration using the Register surface function (residual error shown upon registration)
- Import of experimental displacement data acquired with DIC
- Visualization of the experimental results on the FE model
- Export of the registered patch location to communicate with FE model results in external scientific computing software.

6. Limitations and future development

The ALBA framework enables rapid development of biomedical applications for both research and clinical purposes. However, several limitations still need to be addressed:

- Model Transition: the framework is undergoing a transition from an event-based model to a service-based model. This shift has improved execution speed and debugging efficiency. However, the transition is still in progress. Event management should be restricted to cases where it is strictly necessary, particularly those involving dynamic listener management.
- Library Updates: some dependencies are outdated. While the WxWidgets library was updated last year, the VTK library is outdated and needs to be upgraded to ensure compatibility and support for new features. The update process is already ongoing but not yet complete.
- Documentation: documentation should be enhanced to facilitate the onboarding and development process for new developers.
- Release Strategy: while the rolling release model (described in Section 2) effectively supports iterative research workflows and clinical application development, its scalability is inherently limited by team size and coordination capacity. The current approach assumes a small, tightly integrated development team capable of rapidly synchronizing continuous updates, a prerequisite that may become unsustainable as collaboration expands. If community engagement increases (a key objective of this work), the framework would

require a traditional strategy incorporating versioned releases to support distributed development and backward compatibility.
- Given resource constraints, immediate efforts prioritize new feature development and bug fixes to meet research demands. However, continuous improvements are being made to enhance the framework's stability, maintainability, and long-term usability.

7. Impact

The ALBA framework that has been developed since 2017, similarly to the previous MAF framework, followed an initial idea but then introduced over time specific features driven by the research needs expressed by the projects that supported its development. For this reason, the features exposed are mainly suited to help addressing research questions of computational biomechanics research, which is the main research topic of the group. As witnessed in the illustrative example, the features provided by ALBA can help in all sectors of finite element models validation experiments, where there is the need for an accurate reproduction of the experimental boundary conditions. Another important field of application is the analysis of combined data. For example, it is currently used in the group to investigate the effects a specific orthopaedic surgical procedure has on the progression of the treated pathology. In this case it is necessary to register different medical imaging data (e.g. MRI and CBCT), and to define anatomical reference systems, needed to register images with gait analysis data and with the results obtained from musculoskeletal modelling. This registration step is also fundamental if the loads obtained by musculoskeletal models are to be used as input for finite element models generally derived from medical images data, such as CT. The possibility of performing 2D and 3D measurements such as angles, point and surface distances, volumes and volume densities, among others, can be used in many quantitative imaging and biomechanical research studies.

The ALBA framework is a general framework that can be easily extended and personalised but already provides ready-to-use utilities that are not generally available all together, neither in commercial nor in freely available software, to help in computational biomechanical research. It also provides an easy way to generate organised data structures where medical images, medical signals, finite element models, musculoskeletal models and annotations can be saved and exchanged between researchers, hopefully fostering data and models exchange, which is extremely important to increase open and reproducible research.

Considering only the papers published after the fork of the ALBA from the original MAF, the software has been fundamental in the research activities of the developers' team as it was used for:

- Finite Element Models validation in [9,12];
- generation of musculoskeletal subject-specific models in clinical cohorts to study the outcome of specific surgical treatments [16–19];
- development of ad-hoc software applications to solve specific problems: a software for the identification and measurements of Abdominal Aorta Calcifications to implement and test a new algorithm [20] or a software to plan the skeletal reconstruction with a specific modular prosthetic system in a study to test the effect of the prosthetic device [21].

As not much effort has been put in the dissemination of the ALBA framework so far, its use outside the original research group has been, until recently, negligible. However, some of the software applications that have been developed using the ALBA framework, and that were made freely available (*Bonemat* and *nmsBuilder*) are widely used in the biomechanical research community. The original papers in which the applications were described are constantly and highly cited. Although some of the citations might not reflect the actual use of the software, there is evidence that numerous independent research groups use the software in their research work: see [22–25] for *Bonemat*, and [26–32] for *nmsBuilder* only to cite some recent published papers without the claim of providing an exhaustive list. This clearly indicates a strong and continuing interest in some of the ALBA functionalities.

The ALBA framework has been recently (2023) directly linked within the ORMIR community.[12] As this community adopts Python as primary coding language to bridge the gap between pure developers and applied researchers, we have in progress the wrapping of the whole ALBA framework so that its functionalities can be exposed and used in Python code. The combination of this wrapping effort with the present publication is likely to increase the awareness and the usability of the ALBA open source framework, so that many other research groups might try to build their own applications or simply use the available functionalities, as was the case in which *Bonemat* was wrapped in a Python package for FE analysis[13] [33].

8. Conclusions

ALBA is an open-source C++ versatile framework designed to facilitate the rapid development of specialised application to manage, visualise, and manipulate biomedical data. Specifically designed for biomedical/biomechanical applications, ALBA smoothly handles various data types, including medical images (CT scans, X-rays, MRIs, etc.), surfaces, motion analysis data, and finite element meshes among others. The ALBA framework is easily extensible, providing a full set of importer/exporters, views, operations, interactors and visual pipes so that programmers can concentrate only on new operation/view/VMEs/ etc. development.

The most characterising features are the availability of several otherwise inhomogeneous functionalities, some derived from medical imaging data analysis, some others from CAD software, some others from finite element models handling, or from signal processing, or computational geometry. Their simultaneous availability in a single framework has demonstrated to be very powerful in addressing problems related to musculoskeletal biomechanics and/or orthopaedic research.

The ALBA framework has already been used to develop specific applications that, when freely released, attracted the interest of the scientific community. It has been recently included in the open-source software adopted by the ORMIR community to foster its adoption by other centres with a clear inclination towards open and reproducible research.

CRediT authorship contribution statement

Gianluigi Crimi: Writing – review & editing, Writing – original draft, Supervision, Software, Methodology. **Nicola Vanella:** Writing – review & editing, Software. **Enrico Schileo:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Conceptualization. **Giordano Valente:** Writing – review & editing, Validation, Supervision, Conceptualization. **Giulia Fraterrigo:** Writing – review & editing, Visualization, Validation. **Fulvia Taddei:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

[12] https://ormir.org/software.
[13] https://pypi.org/project/py_bonemat_abaqus.

Acknowledgements

This work was partially funded by the Italian Ministry of Health with the four projects: grant number RF-2016–02364359 and RF-2018–12368274, and Donation programs 5 × 1000 year 2021.

The authors would like to thank all contributors that have participated in the development and testing of the ALBA framework over the years. In particular, we want to mention posthumously Mauro Ansaloni, for his constant contribution as Application Expert in testing all applications developed by the group.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.softx.2025.102188.

References

[1] Schileo E, Taddei F. Finite element assessment of bone fragility from clinical images. Curr Osteoporos Rep 2021;19:688–98. https://doi.org/10.1007/s11914-021-00714-7.

[2] Fregly BJ. A conceptual blueprint for making neuromusculoskeletal models clinically useful. Appl Sci 2021;11:2037. https://doi.org/10.3390/app11052037.

[3] Bonaretti S, Barzaghi L, Barzegari M, Burghardt AJ, Cameron D, Carballido-Gamio J, Crimi G, Durongbhan P, Espinosa Hernandez M, Fraterrigo G, Grassi L, Greer H, Iori G, Kuczynski M, Manske S, McCormick M, Neeteson N, Niethammer M, Pani M, Quintiens J, Sadeghi MM, Santini F, Schileo E, Stok KS, Taddei F, Tse JJ, Vicory J, Wesseling M, Wong AKO, Zukić D. Introducing the open and reproducible Musculoskeletal Imaging Research (ORMIR) community. 2023 Jul 6 doi:10.5281/zenodo.8119243.

[4] Delp SL, Anderson FC, Arnold AS, Loan P, Habib A, John CT, Guendelman E, Thelen DG. OpenSim: open-source software to create and analyze dynamic simulations of movement. IEEE Trans Biomed Eng 2007;54:1940–50. https://doi.org/10.1109/TBME.2007.901024.

[5] Seth A, Hicks JL, Uchida TK, Habib A, Dembia CL, Dunne JJ, Ong CF, DeMers MS, Rajagopal A, Millard M, Hamner SR, Arnold EM, Yong JR, Lakshmikanth SK, Sherman MA, Ku JP, Delp SL. OpenSim: simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. PLoS Comput Biol 2018;14:e1006223. https://doi.org/10.1371/journal.pcbi.1006223.

[6] Valente G, Crimi G, Vanella N, Schileo E, Taddei F. nms Builder: freeware to create subject-specific musculoskeletal models for OpenSim. Comput Methods Programs Biomed 2017;152:85–92. https://doi.org/10.1016/j.cmpb.2017.09.012.

[7] Taddei F, Pancanti A, Viceconti M. An improved method for the automatic mapping of computed tomography numbers onto finite element models. Med Eng Phys 2004;26:61–9. https://doi.org/10.1016/s1350-4533(03)00138-3.

[8] Taddei F, Schileo E, Helgason B, Cristofolini L, Viceconti M. The material mapping strategy influences the accuracy of CT-based finite element models of bones: an evaluation against experimental measurements. Med Eng Phys 2007;29:973–9. https://doi.org/10.1016/j.medengphy.2006.10.014.

[9] Schileo E, Pitocchi J, Falcinelli C, Taddei F. Cortical bone mapping improves finite element strain prediction accuracy at the proximal femur. Bone 2020;136:115348. https://doi.org/10.1016/j.bone.2020.115348.

[10] Viceconti M, Zannoni C, Testi D, Petrone M, Perticoni S, Quadrani P, Taddei F, Imboden S, Clapworthy G. The multimod application framework: a rapid application development tool for computer aided medicine. Comput Methods Programs Biomed 2007;85:138–51. https://doi.org/10.1016/j.cmpb.2006.09.010.

[11] Gamma E, editor, editors. Design Patterns: Elements of Reusable Object-Oriented Software. Design Patterns: Elements of Reusable Object-Oriented Software, 39. Boston, Mass. Munich: Addison-Wesley; 2011. printing.

[12] Baleani M, Fraterrigo G, Erani P, Rota G, Berni M, Taddei F, Schileo E. Applying a homogeneous pressure distribution to the upper vertebral endplate: validation of a new loading system, pilot application to human vertebral bodies, and finite element predictions of DIC measured displacements and strains. J Mech Behav Biomed Mater 2023;140:105706. https://doi.org/10.1016/j.jmbbm.2023.105706.

[13] Chen Y, Dall Ara E, Sales E, Manda K, Wallace R, Pankaj P, Viceconti M. Micro-CT based finite element models of cancellous bone predict accurately displacement once the boundary condition is well replicated: a validation study. J Mech Behav Biomed Mater 2017;65:644–51. https://doi.org/10.1016/j.jmbbm.2016.09.014.

[14] Dahan G, Trabelsi N, Safran O, Yosibash Z. Verified and validated finite element analyses of humeri. J Biomech 2016;49:1094–102. https://doi.org/10.1016/j.jbiomech.2016.02.036.

[15] Grassi L, Schileo E, Taddei F, Zani L, Juszczyk M, Cristofolini L, Viceconti M. Accuracy of finite element predictions in sideways load configurations for the proximal human femur. J Biomech 2012;45:394–9. https://doi.org/10.1016/j.jbiomech.2011.10.019.

[16] Valente G, Pitto L, Schileo E, Piroddi S, Leardini A, Manfrini M, Taddei F. Relationship between bone adaptation and in-vivo mechanical stimulus in biological reconstructions after bone tumor: a biomechanical modeling analysis. Clin Biomech Bristol Avon 2017;42:99–107. https://doi.org/10.1016/j.clinbiomech.2017.01.017.

[17] Valente G, Benedetti MG, Paolis MD, Sambri A, Frisoni T, Leardini A, Donati DM, Taddei F. Long-term functional recovery in patients with custom-made 3D-printed anatomical pelvic prostheses following bone tumor excision. Gait Posture 2022;97:73–9. https://doi.org/10.1016/j.gaitpost.2022.07.248.

[18] Valente G, Benedetti MG, De Paolis M, Donati DM, Taddei F. Differences in hip musculoskeletal loads between limbs during daily activities in patients with 3D-printed hemipelvic reconstructions following tumor surgery. Gait Posture 2023;102:56–63. https://doi.org/10.1016/j.gaitpost.2023.03.005.

[19] Valente G, Grenno G, Dal Fabbro G, Zaffagnini S, Taddei F. Medial and lateral knee contact forces during walking, stair ascent and stair descent are more affected by contact locations than tibiofemoral alignment in knee osteoarthritis patients with varus malalignment. Front Bioeng Biotechnol 2023;11:1254661. https://doi.org/10.3389/fbioe.2023.1254661.

[20] Fusaro M, Schileo E, Crimi G, Aghi A, Bazzocchi A, Barbanti Brodano G, Girolami M, Sella S, Politi C, Ferrari S, Gasperini C, Tripepi G, Taddei F. A novel quantitative computer-assisted score can improve repeatability in the estimate of vascular calcifications at the abdominal aorta. Nutrients 2022;14:4276. https://doi.org/10.3390/nu14204276.

[21] Fraterrigo G, Schileo E, Simpson D, Stevenson J, Kendrick B, Taddei F. Does a novel bridging collar in endoprosthetic replacement optimise the mechanical environment for osseointegration? A finite element study. Front Bioeng Biotechnol 2023;11:1120430. https://doi.org/10.3389/fbioe.2023.1120430.

[22] Castro APG, Altai Z, Offiah AC, Shelmerdine SC, Arthurs OJ, Li X, Lacroix D. Finite element modelling of the developing infant femur using paired CT and MRI scans. PLOS ONE 2019;14:e0218268. https://doi.org/10.1371/journal.pone.0218268.

[23] Henyš P, Vořechovský M, Stebel J, Kuchař M, Exner P. From computed tomography to finite element space: a unified bone material mapping strategy. Clin Biomech 2022;97:105704. https://doi.org/10.1016/j.clinbiomech.2022.105704.

[24] Irarrázaval S, Ramos-Grez JA, Pérez LI, Besa P, Ibáñez A. Finite element modeling of multiple density materials of bone specimens for biomechanical behavior evaluation. SN Appl Sci 2021;3:776. https://doi.org/10.1007/s42452-021-04760-9.

[25] Kok J, Grassi L, Gustafsson A, Isaksson H. Femoral strength and strains in sideways fall: validation of finite element models against bilateral strain measurements. J Biomech 2021;122:110445. https://doi.org/10.1016/j.jbiomech.2021.110445.

[26] Modenese L, Montefiori E, Wang A, Wesarg S, Viceconti M, Mazzà C. Investigation of the dependence of joint contact forces on musculotendon parameters using a codified workflow for image-based modelling. J Biomech 2018;73:108–18. https://doi.org/10.1016/j.jbiomech.2018.03.039.

[27] Montefiori E, Fiifi Hayford C, Mazzà C. Variations of lower-limb joint kinematics associated with the use of different ankle joint models. J Biomech 2022;136:111072. https://doi.org/10.1016/j.jbiomech.2022.111072.

[28] Akhundov R, Saxby DJ, Diamond LE, Edwards S, Clausen P, Dooley K, Blyton S, Snodgrass SJ. Is subject-specific musculoskeletal modelling worth the extra effort or is generic modelling worth the shortcut? PLOS ONE 2022;17:e0262936. https://doi.org/10.1371/journal.pone.0262936.

[29] Lavaill M, Martelli S, Kerr GK, Pivonka P. Statistical quantification of the effects of marker misplacement and soft-tissue artifact on shoulder kinematics and kinetics. Life 2022;12:819. https://doi.org/10.3390/life12060819.

[30] Charles JP, Grant B, D'Août K, Bates KT. Subject-specific muscle properties from diffusion tensor imaging significantly improve the accuracy of musculoskeletal models. J Anat 2020;237:941–59. https://doi.org/10.1111/joa.13261.

[31] Li G, Ao D, Vega MM, Shourijeh MS, Zandiyeh P, Chang S-H, VO Lewis, Dunbar NJ, Babazadeh-Naseri A, Baines AJ, Fregly BJ. A computational method for estimating trunk muscle activations during gait using lower extremity muscle synergies. Front Bioeng Biotechnol 2022 Dec 13. https://doi.org/10.3389/fbioe.2022.964359.

[32] Curreli C, Di Puccio F, Davico G, Modenese L, Viceconti M. Using musculoskeletal models to estimate in vivo total knee replacement kinematics and loads: effect of differences between models. Front Bioeng Biotechnol 2021 Jul 28. https://doi.org/10.3389/fbioe.2021.703508.

[33] Pegg EC, Gill HS. An open source software tool to assign the material properties of bone for ABAQUS finite element simulations. J Biomech 2016;49:3116–21. https://doi.org/10.1016/j.jbiomech.2016.07.037.