MOtoNMS Documentation

Release 2.2

Alice Mantoan, Monica Reggiani

June 05, 2015

CHAPTER

ONE

REDISTRIBUTION: TERMS AND CONDITIONS

The latest version of this manual is provided in the GitHub Project Pages: http://rehabenggroup.github.io/MOtoNMS/.

When using MOtoNMS please acknowledge the authors and cite the following paper:

A. Mantoan, M. Reggiani, M. Sartori, Z. Sawacha, C. Pizzolato, C. Cobelli, *A MATLAB generic tool to efficiently process C3D data files for applications in OpenSim*, XXIV Congress of International Society of BIomechanics (ISB 2013), Nadal, Brazil, 2013.

CHAPTER

TWO

ACKNOWLEDGMENTS

Monica Reggiani acknowledges the financial support of the ICT programme within the Seventh Framework Programme for Research of the European Commission (BioMot project, grant number: 611695).

Alice Mantoan acknowledges the financial support of Prof. Claudio Cobelli and the Bioengineering of Human Movement Laboratory of the University of Padova.

The authors would also thank Michele Vivian and Dr. Massimo Sartori for their contributions to the initial idea of this project. Claudio Pizzolato, Peter Staab, Alessandra Scarton, Felix Jung for testing MOtoNMS. We are grateful to Dr. Leonardo Gizzi for his help in data collection at UMG and Dr. Fabiola Spolaor for her help at UNIPD. We would also like to thank Prof. David G. Lloyd, Prof. Claudio Cobelli, Dr. Zimi Sawacha, and Prof. Dario Farina for granting us access to their laboratory facilities. Finally, the authors would like to thank all the researchers at the University of Western Australia and at the Griffith University that contributed to the initial processing pipeline.

CHAPTER

THREE

NEW FEATURES IN MOTONMS 2.2

MOtoNMS 2.2 has been updated to support two largely required features:

- **trials with different directions of motion**
- **force plates with pads**

MOtoNMS 2.2 also provides the users with additional logging information about the **EMG processing** steps. An new output folder (`maxemg`) is created for each dynamic elaboration, with plots and log data related to the computation of maximum EMG values.

A last objective of this release is improving the **processing of marker trajectories**. Users have the possibility to define the gaps' maximum size that will be interpolated, and a **piecewise filtering** has been implemented for markers trajectories that still have NaN values.

Finally, computation of **joint centers** in Static Elaboration is no more mandatory.

CHAPTER

FOUR

INTRODUCTION

MOtoNMS (matlab MOtion data elaboration TOolbox for NeuroMusculoSkeletal applications) is a freely available toolbox that aims at providing a complete tool for post-processing of movement data for their use in neuromusculoskeletal software.

MOtoNMS has been design to be flexible and highly configurable, to satisfy the requests of different research groups. Users can easily setup their own laboratory and processing procedures, without constraints in instruments, software, protocols, and methodologies, everything without change in the MATLAB code.

MOtoNMS also improves the data organization, providing a clear structure of input data and automatically generating output directories.

The authors hope is that MOtoNMS may be useful to the research community, reducing the gap between experimental data from motion analysis and musculoskeletal simulation software, and uniforming data processing methods across laboratories.

Please help us in maintaining and updating the toolbox. You can provide feedback, comments, suggestions, and new modules using any of the following channels:

- the MOtoNMS Public Forum, available from the SimTK project page: https://simtk.org/home/motonms
- the GitHub repository: https://github.com/RehabEngGroup/MOtoNMS
- email: ali.mantoan@gmail.com, monica.reggiani@gmail.com.

CHAPTER

FIVE

GETTING STARTED

This section introduces MOtoNMS architecture and how to setup the software on your computer.

5.1 Architecture Overview

MOtoNMS separates data processing from configuration of the execution. This guarantees high configurability, flexibility, and a user friendly toolbox not requiring a deep confidence with MATLAB for its use.

MOtoNMS has three main *processing steps* (Fig. 5.1- orange boxes): (1) *C3D2MAT*, (2) *Data Processing*, and (3) *Static Elaboration*. Briefly, the objectives of each blocks are:

- *C3D2MAT*: retrieves data from the C3D input files and store them in organized MATLAB structures;
- *Data Processing*: works on dynamic trials processing markers trajectories, ground reaction forces, and EMG signals and produces OpenSim files (`.trc` and `.mot` format);
- *Static Elaboration*: processes static trials, computes joint centers, and stores markers trajectories in the corresponding `.trc` file.

MOtoNMS configuration goes through user-friendly MATLAB interfaces (Fig. 5.1- blue boxes), that create the following three XML configuration files with the input parameters for the processing steps (Fig. 5.1):

- `acquisition.xml` created by *Acquisition Interface*. Each data set must include an `acquisition.xml` file. It gathers all information describing the acquisition session such as the number of force plates, coordinate system orientations, marker sets, and EMG setups (see *Acquisition Interface* for details).
- `elaboration.xml` created by *Elaboration Interface: configure your elaboration*. This XML file includes all the parameters that define the data processing. Examples of these parameters are the identifiers of the trials to be processed, the cutoff frequencies for filtering the different input data, the list of markers to be written in the `.trc` files, and the method for gait event detection (see *Elaboration Interface: configure your elaboration* for details).
- `static.xml` created by *Static Interface: configure your elaboration*. This XML file defines the parameters for the static elaboration part (see *Static Interface: configure your elaboration* for details).

All the files are encoded using the XML language. We have chosen this language for its suitability in encoding information. While XML files are easily readable but their editing might be complex, therefore MOtoNMS provides user-friendly MATLAB interfaces that creates these configuration files according to the proper semantic. Indeed, XML files must respect the syntax of a grammar defined in XML Schema (XSD) files (see *Appendix B: Validation of Setup and Configuration Files* for details on XML files validation).

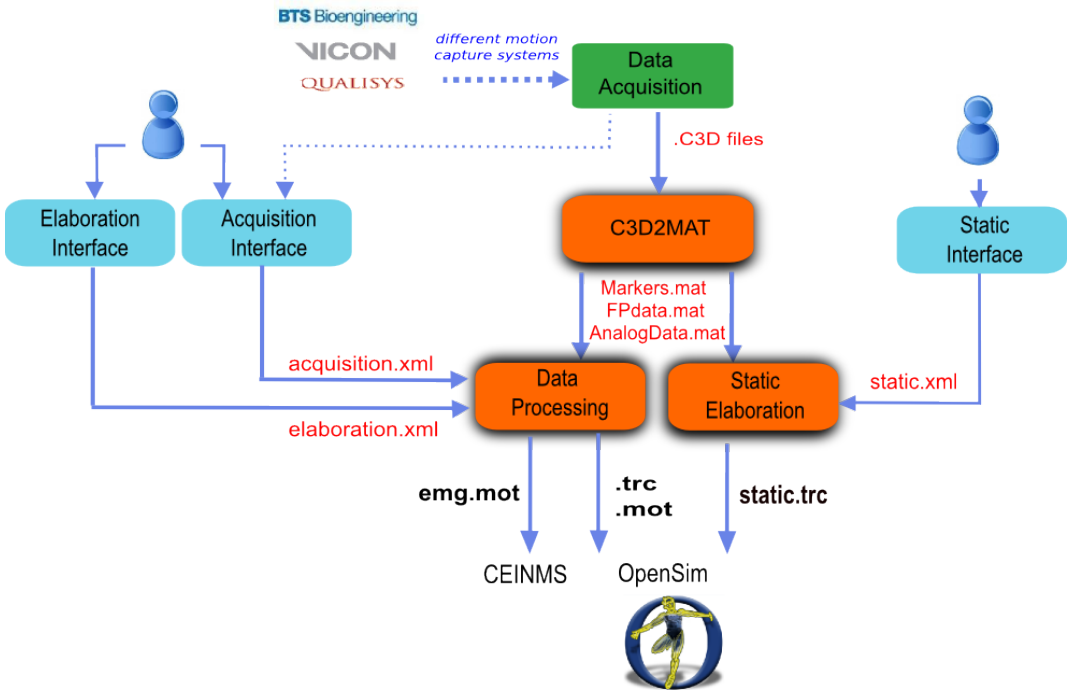

Fig. 5.1: MOtoNMS overview schema

5.2 Supported Applications: OpenSim and CEINMS

MOtoNMS has been design to support different output file formats and several musculoskeletal applications. Current version of MOtoNMS already supports OpenSim (https://simtk.org/home/opensim) and CEINMS (https://simtk.org/home/ceinms).

MOtoNMS processes data collected with different gait instrumentations and produces input files for OpenSim (`.trc` and `.mot`, standard OpenSim file formats). When available, EMG signals are also processed and can be exported in several formats (`.mot`, OpenSim motion, `.sto`, OpenSim Storage, and `.txt`, plain text format) compatible with the CEINMS toolbox (https://simtk.org/home/ceinms) and easily usable also by other applications.

5.3 Installation

5.3.1 Download

MOtoNMS is released under the GNU General Public Licence v.3or any later version. An archive of the latest release is available at the project webpage at the SimTk.org website (https://simtk.org/home/motonms/). We also provided a set of dataset to play with. If you are interested in collaborate with the development,the software repository is available in GitHub: https://github.com/RehabEngGroup/MOtoNMS.

5.3.2 Requirements

MOtoNMS is implemented in MATLAB and runs on the major operating systems (Windows, Linux, MacOS X). It works with MATLAB R2009a and later versions.

MOtoNMS requires one of the following software for the *C3D2MAT* part:

- C3Dserver (http://www.c3dserver.com/), which works only with MATLAB 32bitand in Windows.
- Biomechanical Toolkit (BTK, https://code.google.com/p/b-tk/). You can download the BTK version for your system from the BTK project site: https://code.google.com/p/b-tk/wiki/MatlabBinaries.

You also need to add the correct BTK folder to the path of MATLAB. Type the command `help btk` in the MATLAB's command window to verify that BTK is loaded in MATLAB. You should see the main documentation for the BTK toolbox. If this is not the case, check the path that you added to the list of the directories loaded by MATLAB to register the BTK toolbox.

You can choose to install either C3Dserver or BTK. According to your choice, you must use the corresponding *C3D2MAT* implementation, with C3Dserver (`C3D2MAT_c3dserver`) or with BTK (`C3D2MAT_btk`).

Laboratory requirements:

- Data collected with a motion capture and force plate system must be labeled and then exported as a C3D file, which is the input format for MOtoNMS. Markers labels MUST not include spaces (e.g. 'LLM' and not 'L LM') and duplicates of marker labels are NOT acceptable in the same markerset.

CHAPTER

SIX

FOLDERS: ORGANIZE YOUR WORK

Before starting with the description of MOtoNMS components, lets have a look to MOtoNMS code structure and how to organize your experimental data folders.

6.1 Code Organization

Fig. 6.1 shows MOtoNMS code structure. It is organized in threeparts:

- *Source Code* (`MOtoNMS/src/`) directories include MOtoNMS code. Functions are organized in several directories reflecting the main parts of the toolbox, that will be described in the next chapters: *Acquisition Interface* (*figure MOtoNMS overview schema* - blue boxes) and *C3D2MAT*, *Data Processing* and *Static Elaboration* (*figure MOtoNMS overview schema* - orange boxes). Two implementations for *C3D2MAT* are available, each one using a different tool (C3Dserver and BTK) to access C3D files. A directory (`src/shared`) stores MATLAB functions common to more than one processing step.
- *Setup Files* (`MOtoNMS/SetupFiles/`) directory that includes files describing the laboratory setup, markers and EMG protocols, EMG output labels depending on the final application (see *Setup Files in Data Processing*), and information for join center computation methods (see *Setup Files in Static Elaboration*). Their organization in folders matches the source code structure. Several setup files are already available in the toolbox distribution (*Appendix B: Validation of Setup and Configuration Files*) and are often used as examples in the manual. If you need to create new ones, refer to the *Setup Files* sections in each chapter as a reference for their editing.
- *Licences* (`MOtoNMS/Licenses/`) directory which includes Licenses for tools used by MOtoNMS and developed by other authors.

The next chapters of the manual describe in details each part of MOtoNMS: its objectives, how it works, required configuration files and useful setup files.

6.2 Data Organization

An advantage of MOtoNMS is that it helps in keeping your experimental data folder well organized. Data storage is a common and important issue, especially when large amount of data are involved or when the collaboration among research teams leads to sharing of data sets and results.

Thus, we have decided to force some simple rules in the storage of collected data. This allow an automatic generation of output directories with a well defined structure. Therefore, the use of MOtoNMS forces the arrangement of the processed data sets with the same structure, facilitating the retrieval of information and results and the sharing of your work with other research teams.

But, actually, the only real rule that you have to follow is as simple as placing the folders of your collected data in a folder called `InputData`.

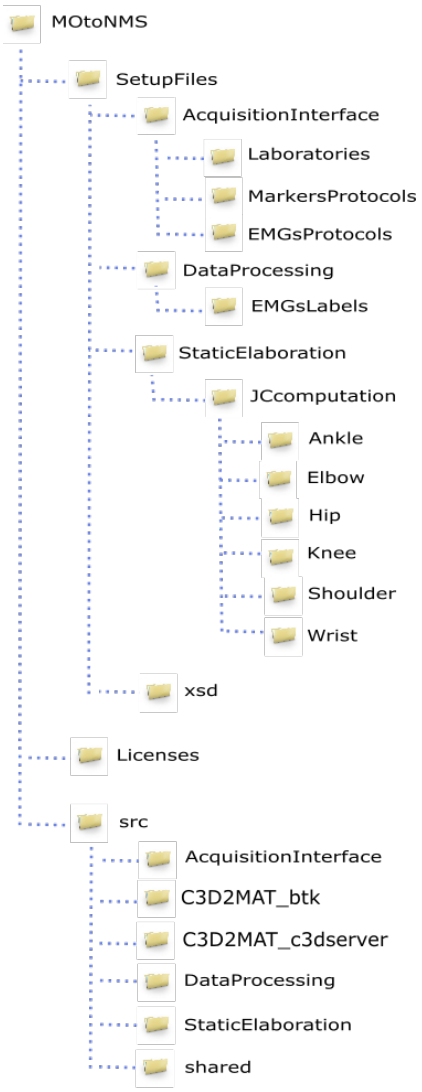

Fig. 6.1: Overview of MOtoNMS Code Organization. The distribution of MOtoNMS includes an additional folder (`TestData`) with data from four different laboratories to test the toolbox.

We then encourage users to create a different folder for each acquisition, named with the date when data were collected. These folders should then be stored in another folder named with the subject identifier.

Fig. 6.2 shows a representation of how MOtoNMS suggests to organize the data set: in black is the path of data from a single acquisition session. Inside the folder, together with the expected C3D files, you must include the `acquisition.xml` file (Fig. 6.2 - red) that fully describe the collected data (see *Acquisition Interface*).

The execution of MOtoNMS automatically creates new folders (Fig. 6.2 - green). The new path for the output folder is created based on the input file path just replacing `InputData` with `ElaboratedData` (e.g, `MyData\ElaboratedData\subjectXXX\ Year-Month-Day\`). Then the execution of the different tools create new subdirectories. *C3D2MAT* extracts data from the C3D files and stores them in mat format in the subfolder `sessionData` (`MyData\ElaboratedData\subjectXXX\Year-Month-Day\sessionData\`). `staticElaborations` and `dynamicElaborations` subfolders store the output of *Static Elaboration* and *Data Processing* parts, respectively. Finally, the results of multiple executions of these two tools, with different configurations for the same input data, are stored in different subfolders, each one named with an identifier chosen by the user through the user interface (*Elaboration Interface: configure your elaboration*).

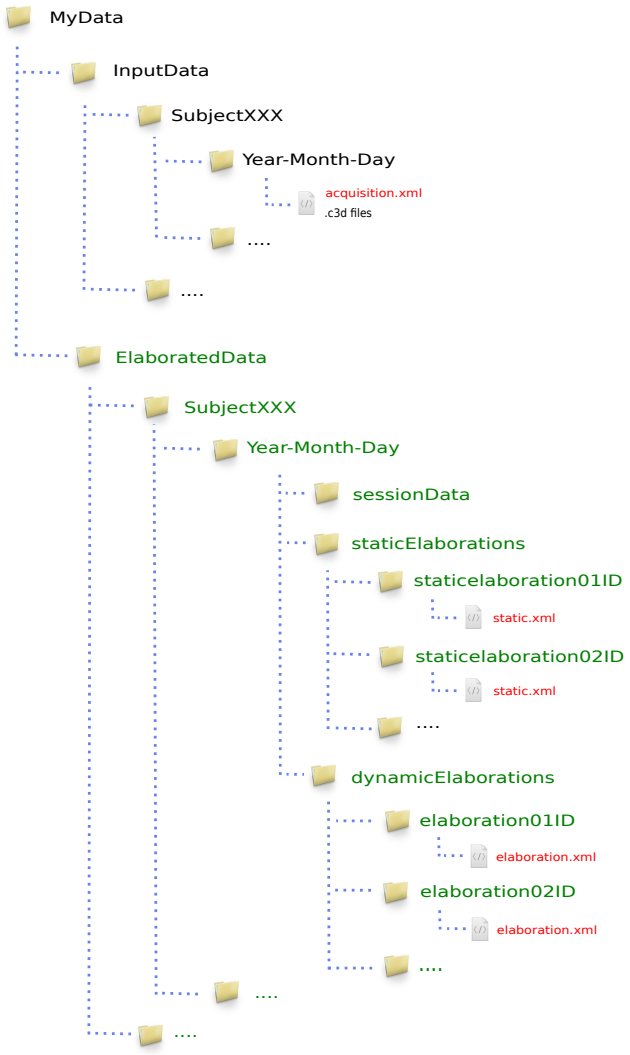

Fig. 6.2: Data Folders Organization. In black the input data that the user must provide. In red the configuration files created by MOtoNMS and in green the output folders generated by the toolbox.

CHAPTER

SEVEN

ACQUISITION INTERFACE: DESCRIBING YOUR DATA

During this step, a graphical user interface (GUI) will assist you in the creation of a file describing the acquisition session (`acquisition.xml`). This file must be created and included in the folder with the C3D data before running the next steps (*Folders: organize your work figure*). Furthermore, `acquisition.xml` is helpful when the data are shared among researchers as it allows to easily figure out all the details about the acquisition session.

The procedure is quite simple: you will go through a sequence of questions about the tracked subjects, people that collected the data, marker and EMG protocol, laboratory characteristics, etc.

The only tricky point is about the prefilled *setup files*. Information about the laboratory setup, markers and EMG protocols are usually common to several acquisitions. Therefore, instead of writing again and again the same values for each acquisition, these data are stored in setup files that can be selected from the GUI (Fig. 7.1). Thus, before running the Acquisition Interface you need to have already all the necessary *setup files*. A few examples are available in the directories:

- `MOtoNMS\SetupFiles\AcquisitionInterface\Laboratories` (Laboratory Configurations)
- `MOtoNMS\SetupFiles\AcquisitionInterface\MarkersProtocols` (Markers Protocols)
- `MOtoNMS\SetupFiles\AcquisitionInterface\EMGsProtocols` (EMG Protocols).

If you need a setup file not already included in the folders, you can add new setup files following the instruction in the sections `Setup Files` of each chapter.

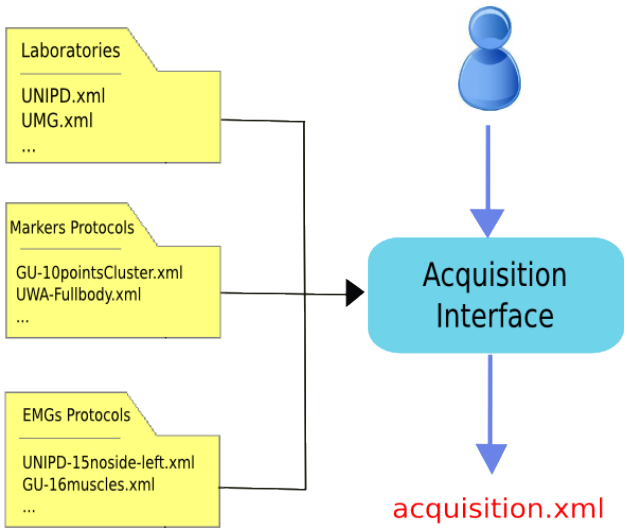

Fig. 7.1: Acquisition Interface schema. The user uses the Acquisition Interface GUI to create `acquisition.xml`. Information about Laboratory configuration and Markers and EMG Protocols are stored in prefilled setup files to avoid reentering data common to multiple acquisitions.

7.1 How to run the program

1. Set MATLAB path to `src\AcquisitionInterface` folder
2. Run `mainAcquisitionInterface.m` (this calls `AcquisitionInterface.m`, the core program)
3. Answer to questions with the required information

Output: `acquisition.xml` file (saved in the `InputData` folder, *Folders: organize your work figure* - red).

At the beginning, the graphical interface asks whether the user wants to reload an `acquisition.xml` file. This is a useful functionality when the user wants to start with information stored in a file already available.

The resulting XML file is considered perfectly valid only when all the information is included (*Appendix B: Validation of Setup and Configuration Files*). However, `mainAcquisitionInterface.m` only requires the data which are mandatory for the following processing, allowing to skip some answers.

Please pay ATTENTION to the following IMPORTANT NOTES:

Warning:

- Input files folder MUST include the `acquisition.xml` file before running *Data Processing* and *Static Elaboration* steps.
- The `acquisition.xml` file is NOT required for *C3D2MAT*, so you can run *Acquisition Interface* and *C3D2MAT* in any order.
- Trials must be named as: *trial type* (`walking, running, fastwalking`, etc.) + *sequential number*. Examples: `walking1, fastwalking5`, etc.
- The sequential number MUST not include any 0 in the front (write 1 not 001), otherwise errors may occur in the processing.
- Trial names in the `acquisition.xml` MUST correspond to the actual input file names.
- Motion direction refers to the trial and is asked in the GUI (Fig. 7.2). Default value is `forward`, which means that your subject is moving according to the positive direction of the laboratory coordinate system (CS). If this is not the case, you can choose among the following additional options: `backward`, `90left`, `90right`. All of them are defined with respect to the positive direction of your laboratory CS. If your data do not fit in any of the available choices, please select the remaining option (`-`, i.e. *unconventional* in the `acquisition.xml` file). In this case, no extra rotations will be applied to your data: they will be projected in the OpenSim CS preserving the orientation they have in the laboratory CS. For static trials, please indicate the direction your subject is looking towards.
- If your force platforms include a plate pad on the surface, you can define the thickness of each pad in the laboratory setup file (*Laboratory*). Addition of the `PadThickness` tag in the file is allowed but not mandatory (*Plate Padding*). However, please note that if you add it for one force platform, you MUST do the same for ALL the other force platforms defined in the same XML, even if the pad is not present. In this case, you can set the pad thickness to 0.

7.2 Setup Files

The following files speed up the process of compiling the `acquisition.xml` file that describes your acquisition session. When MOtoNMS has been already used in your laboratory to collect data with EMG and marker protocols do not go further: someone else should have already created these files. If you are not lucky and you have to write your own setup files, do not be scared: it is a simple procedure if you follow carefully the description in this section. Additionally, once you are done, you can check that the final XML file respect the required syntax with the validation procedure (see *Appendix B: Validation of Setup and Configuration Files*).

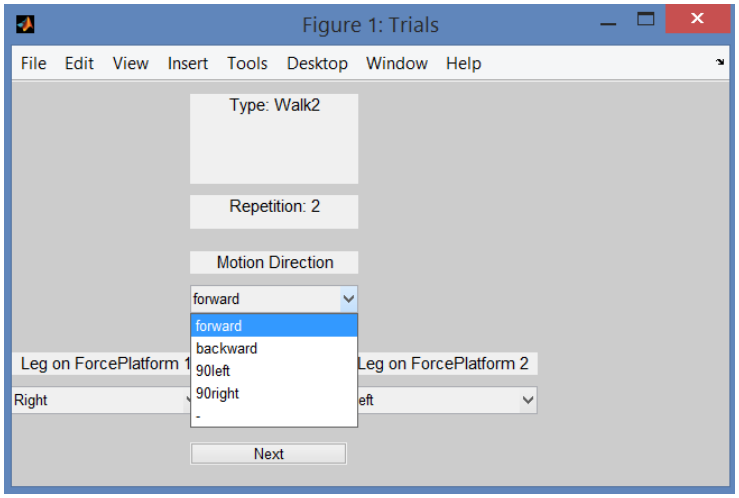

Fig. 7.2: Acquisition Interface: how to set the motion direction for each trial. This step of the GUI will ask you the leg that strikes on each force platform, and the direction of motion for each of your input data file. This information allows to support trials with different motion directions. Indeed, extra rotations can be required to align data with the positive direction of OpenSim coordinate system (CS).

7.2.1 Laboratory

This XML setup file describes the characteristics of the laboratory where data are collected. It has been introduced to avoid re-entering same data for each acquisition carried on in the same laboratory. Its name should uniquely identify the laboratory to whom it refers. Therefore, the best choice is to use a combination of laboratory name, department, university.

The following is an example of an XML setup file with information about the laboratory (available at `SetupFiles\AcquisitionInterface\Laboratories\UNIPD.xml`).

```
<Laboratory xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Name>UNIPD</Name>
  <MotionCaptureSystem>BTS</MotionCaptureSystem>
  <CoordinateSystemOrientation>ZYX</CoordinateSystemOrientation>
  <NumberOfForcePlatforms>2</NumberOfForcePlatforms>
  <ForcePlatformsList>
    <ForcePlatform>
      <ID>1</ID>
      <Brand>Bertec</Brand>
      <Type>1</Type>
      <FrameRate>960</FrameRate>
      <FPtoGlobalRotations>
        <Rot>
          <Axis>X</Axis>
          <Degrees>-90</Degrees>
        </Rot>
      </FPtoGlobalRotations>
    </ForcePlatform>
    <ForcePlatform>
      <ID>2</ID>
      <Brand>Bertec</Brand>
      <Type>1</Type>
      <FrameRate>960</FrameRate>
      <FPtoGlobalRotations>
        <Rot>
```

```
26            <Axis>X</Axis>
27            <Degrees>-90</Degrees>
28          </Rot>
29          <Rot>
30            <Axis>Y</Axis>
31            <Degrees>180</Degrees>
32          </Rot>
33        </FPtoGlobalRotations>
34      </ForcePlatform>
35    </ForcePlatformsList>
36  </Laboratory>
```

The file MUST include:

1. **Orientation of the coordinate system** (`<CoordinateSystemOrientation>` tag, line 4)

The coordinate system orientation refers to the global or laboratory coordinate system. We used the following convention:

- 1st axis: direction of motion
- 2nd axis: vertical axis
- 3rd axis: right hand rule

with the assumption that the 1st axis has the same verse of OpenSim 1st axis, i.e. it should be the positive direction of motion (Fig. 7.3). This convention requires that for any combination of the three axes (e.g. YZX, XZY, YXZ, etc...), the first axis must always be the direction of motion in your lab, the second one your vertical axis, and the last the one results from the right hand rule (Fig. 7.3). The adopted convention follows ISB recommendation [1]. Line 4 shows an example of definition of coordinate system orientation.

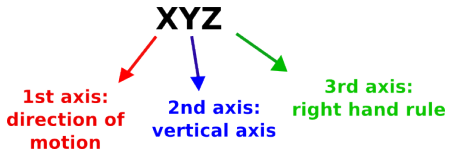

Fig. 7.3: Convention for Coordinate System Orientation: example of interpretation.

2. **Type of force platforms** (`<Type>` tag, lines 10 and 22)

Force platform (FP) type must be indicated because it influences output data (refer to The C3D File Format User Guide, by Motion Lab Systems, pag. 88 and to your force platform manual).

MOtoNMS recognizes and processes data from force platforms of type 1, 2, 3, and 4. Each plaform returns different data as shown in the following:

- type 1: [Fx Fy Fz Px Py Mz]
- type 2 and 4: [Fx Fy Fz Mx My Mz]
- type 3: [Fx12 Fx34 Fy14 Fy23 Fz1 Fz2 Fz3 Fz4]

where F are the measured reaction forces, M the moments, and P the center of pressure (CoP) along the three directions. Numbers in type 3 platform refers to the sensors (refer to the C3D File Format User Guide for additional information).

MOtoNMS can process data from several FPs of different type at the same time (i.e. FPs in your laboratory can be of different types). However, there is a single exception: if you have a FP of type 1, all FP in your laboratory should be

[1] Ge Wu and Peter R. Cavanagh, ISB Recommendation for standardization of kinematic data in the reporting, Vol. 28 No. 10. pp. 1257-1261,1995

of type 1. If this is not the case (i.e. you have at least one FP of type 1 and at least another of a different type), please contact the authors for more information on how to proceed.

3. **Rotation between the force platform and the global coordinates** (`<FPtoGlobalRotations>` tag, lines 12-17 and 24-33)

It is well known that each FP has its own coordinate system (Fig. 7.4) and that C3D files store FP data in the corresponding FP coordinate system. Therefore, the configuration file about the laboratory must provide also the transformation to rotate each FP reference system to the global one (lines 12-17 and 24-33).

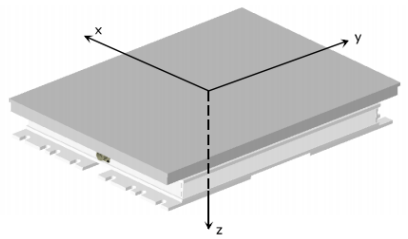

Fig. 7.4: Bertec Plate Coordinate System (from Bertec Force Plates Manual, version 1.0.0, March 2012, http://bertec.com/uploads/pdfs/manuals/Force%20Plate%20Manual.pdf).

The XML example shows how to configure two Bertec force platforms. Their relative position and coordinate systems is shown in Fig. 7.5. Lines 12-17 (FP 1) and 24-33 (FP 2) list the required rotations. When more than one rotation is required, they are listed in sequence and estimated around moving axes (lines 24-33).

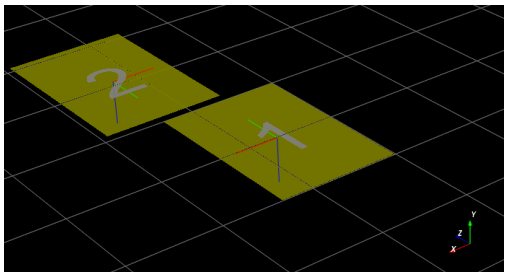

Fig. 7.5: FP and global coordinate systems orientation of UNIPD Laboratory.

Plate Padding

An additional entry, `PadThickness` (line 6), may be included in the laboratory XML file, when data are collected with pads on the surface of FPs. This will allow to correct the computation of CoP coordinates accordingly (please refer to http://www.kwon3d.com/theory/grf/pad.html for additional information). Pad thickness value must be provided in the same unit of length of plate moments.

The following is an example of definition of a force platform including a pad of 7.5mm, in the laboratory XML file.

```
1 <ForcePlatform>
2       <ID>1</ID>
3       <Brand>Bertec</Brand>
4       <Type>1</Type>
5       <FrameRate>960</FrameRate>
6       <PadThickness>7.5</PadThickness>
7       <FPtoGlobalRotations>
8         <Rot>
```

```
          <Axis>X</Axis>
          <Degrees>-90</Degrees>
        </Rot>
      </FPtoGlobalRotations>
</ForcePlatform>

Addition of PadThickness tag in the laboratory XML file.
```

Please, note that if you add the pad thickness to one force platform, you need to do the same for all the other force platforms defined in the same XML setup file, even if the pad is not present. In this case, you can set the value to 0.

7.2.2 Markers Protocols

Each marker protocol must be defined in a separate XML file. In the following an example of an XML file with information about marker protocols (available at `SetupFiles\AcquisitionInterface\MarkersProtocols\UWA-Fullbody.xml`)

```
1 <MarkersProtocol>
2   <Name>UWA-Fullbody</Name>
3   <MarkersSetStaticTrials>LASI RASI LPSI RPSI ... </MarkersSetStaticTrials>
4   <MarkersSetDynamicTrials>C7 RACR LPSI RPSH  ... </MarkersSetDynamicTrials>
5 </MarkersProtocol>
```

In the example the markers set for both Static (`<MarkersSetStaticTrials>` tag, line 3) and Dynamic Trials (`<MarkersSetDynamicTrials>` tag, line 4) are defined. Names of the markers must match the labels used to identify the markers in the C3D files.

The labels of the markers cannot include spaces as this would prevent the creation of this configuration file.

7.2.3 EMGs Protocols

EMG protocol must be defined in an XML file. An example (available at `SetupFiles\AcquisitionInterface\EMGsProtocols\UNIPD-15noside-left.xml`) is shown in the following:

```
1  <EMGsProtocol>
2    <Name>UNIPD-15noside-left</Name>
3    <MuscleList>
4      <Muscle>Gluteus maximus</Muscle>
5      <Muscle>Gluteus medius</Muscle>
6      <Muscle>Tensor fasciae latae</Muscle>
7      <Muscle>Sartorius</Muscle>
8      .....
9    </MuscleList>
10   <InstrumentedLeg>Left</InstrumentedLeg>
11 </EMGsProtocol>
```

The XML file includes information about the name of the protocol (line 2), the list of the muscles (lines 3-9) and the instrumented leg (line 10). Muscles names (lines 4-8) MUST be those assigned during the acquisition session and, therefore, must agree with labels in the C3D files.

When EMG data are collected, the XML file must define the instrumented leg. Depending on the leg with EMG sensors its value will be `Right, Left, Both, None`. These are the only acceptable strings. When EMG signals are not collected during the acquisition session, this configuration file is not required, and the user just sets at 0 the number of EMG systems on the Acquisition Interface GUI (Fig. 7.6).

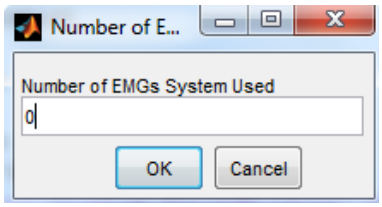

Fig. 7.6: Acquisition Interface: how to set the Number of EMGs System used during the acquisition.

CHAPTER

EIGHT

C3DTOMAT: FROM C3D TO MATLAB FORMAT

This step retrieves data from the C3D input files and store them in organized MATLAB structures (`.mat` files). This avoid to have continuous accesses to C3D files, which are computationally expensive and redundant. Any setup or configuration file is required in this step.

C3D2MAT only requires to specify the input file path. If not already available, it generates corresponding folders for the elaborated data (*Folders: organize your work*, `MyData\ElaboratedData\subjectXXX\Year-Month-Day\`) and then converts all the data in mat format and stored them in the `sessionData` folder (`MyData\ElaboratedData\subjectXXX\Year-Month-Day\sessionData\`). For each trial (i.e., C3D file in the input folder), a new subfolder is created with Markers (`Markers.mat`), Force Platform Data (`FPdata.mat`) and all data stored in the analog channels (`AnalogData.mat`). When C3D files include events, they are converted and stored as well (`Events.mat`).

You can add events to your C3D files using Mokka, a freely available software for biomechanical data analysis (http://b-tk.googlecode.com/svn/web/mokka/index.html), which supports many different file formats and allows to write on C3D files with a very user-friendly interface.

Information common to all the trials of the session are saved in the `sessionData` folder. This includes markers labels of dynamic trials (`dMLabels.mat`), analog data labels (`AnalogDataLabels.mat`), force platform information (`ForcePlatformInfo.mat`), rates (`Rates.mat`), and the trials list (`trialsName.mat`). If any inconsistency is found, it is pointed out in the MATLAB Command Window.

Two implementations for `C3D2MAT` are available, one using C3Dserver software and one exploiting BTK. They return the same results, but are based on two different tools to access C3D files. You can choose among the two alternatives according to your system requirements (refer to *Installation*).

8.1 How to run the program

1. Set MATLAB path on `src\C3D2MAT_btk` (or `C3D2MAT_c3dserver`) folder
2. Run `C3D2MAT.m`
3. Select your C3D input data folder

Output: data from C3D files in the specified data folder are converted and saved in mat format.

Please pay ATTENTION to the following IMPORTANT NOTES:

Warning:

- Input files folder MUST be inside a folder named `InputData` (*Folders: organize your work*).
- The execution of `C3D2MAT` does not overwrite information common to the acquisition session, i.e. that must be the same for all trials collected during an acquisition, as `trialsName.mat`, `Rates.mat`, `dMLabels`, `AnalogDataLabels.mat`, `ForcePlatformsInfo.mat`. Please, keep in mind to cancel the `sessionData` folder before running again the program if you add trials or modify your data set.
- Different events MUST have different name (Analysis Window Definition, `WindowFromC3D`).
- Events for foot strike and foot off must identify frames with non-zero force values (Analysis Window Definition, `StanceOnFPfromC3D`).
- If you are using `C3D2MAT_c3dserver`, remember to install C3Dserver software (*Installation*) and that it works only on MATLAB 32 bit.
- If you are using `C3D2MAT_btk`, remember to download the correct BTK version for your system and to add it to the path of MATLAB. Type the command `help btk` in the MATLAB command window to verify that BTK is loaded in MATLAB (*Installation*).
- Data gathered using FP of type 4 are stored in the C3D files in V (http://www.c3d.org/pdf/c3dformat_ug.pdf). `C3D2MAT` retrieves and directly converts force data in N and $N \cdot mm$ or $N \cdot m$ (according to the used distance unit). Be aware that the scale of results saved in `FPdata.mat` does not match with the one you see in the input C3D files.
- C3D files storing data gathered with FPs of type 4 must include information about the corresponding FP calibration matrix. This is required to convert force data from V to N and $N \cdot mm$ or $N \cdot m$.

CHAPTER

NINE

DATA PROCESSING: ELABORATE YOUR DYNAMIC TRIALS

Starting from motion data stored as MATLAB structures by *C3D2MAT*, Data Processing (Fig. 9.1) produces `.trc` and `.mot` files for OpenSim, storing respectively markers and forces information. When collected, it also processes EMG signals. Data Processing is only responsible for handling dynamic trials, static trials are managed by *Static Elaboration*.

9.1 Markers and Ground Reaction Forces Elaboration

This section provides a description of the elaboration steps for marker trajectories and ground reaction forces (Fig. 9.1 - left column).

At first, markers trajectories undergo piecewise cubic interpolation to fill possible gaps; a text file (`InterpolationNote.txt`) with information about the procedure is also saved. Interpolation is executed in the range between the first and last frames where marker occurs, while the values outside this range are set to zero. The FP Data Splitting step (Fig. 9.1) handles GRF data generated by the most common force plate types (refer to The C3D File Format User Guide) with different procedure depending on the FP type and its output. Then, the pre-processed marker data and raw GRFs are filtered with a zero-lag second order low pass Butterworth filter at customizable cut-off frequencies (*Elaboration Interface: configure your elaboration*). When not available from FP data, the centers of pressure are computed exploiting the filtered GRFs. The analysis window definition sub-block (Fig. 9.1) selects the portion of the data to be processed. Different methods are available and described in a dedicated section (Analysis Window Definition). At this step, plots of the computed values (i.e., raw EMG, envelopes, and raw and filtered forces, CoP, and moments) are stored for offline visual inspection. Processed GRFs are used to compute free torques based on filtered forces, moments and CoP for the selected frames. Finally, marker and GRF data are projected from laboratory or FP reference system to the global reference system of the selected musculoskeletal application, i.e. OpenSim. Required rotations depend on the laboratory setup described in the laboratory setup file (*Laboratory*).

9.2 EMG Processing

When available, EMG signals are also processed for EMG-driven neuromusculoskeletal applications. Currently, MOtoNMS supports CEINMS input file formats (https://simtk.org/home/ceinms), but the results can be easily adapted to other applications.

A subset of all the acquired muscles can be selected. Maximum value for each EMG signal is estimated from one or more trials defined by the user: when available, a trial of maximal voluntary isometric contraction can be used, as well as a subset of the input trials. A text file (`maxemg.txt`) logs information about this step (maximum EMGs value and the corresponding trial and time for each of them). Envelopes for the EMGs are computed and then normalized with their estimated maximum value. Obtained normalized envelopes are stored by default as a `.mot` (SIMM and OpenSim motion) file (`emg.mot`) with output labels that can be defined by the user (for details refer to *Setup Files* in this chapter). Plain text and `.sto` (OpenSim storage) file formats are also available for logging the results of EMG processing.

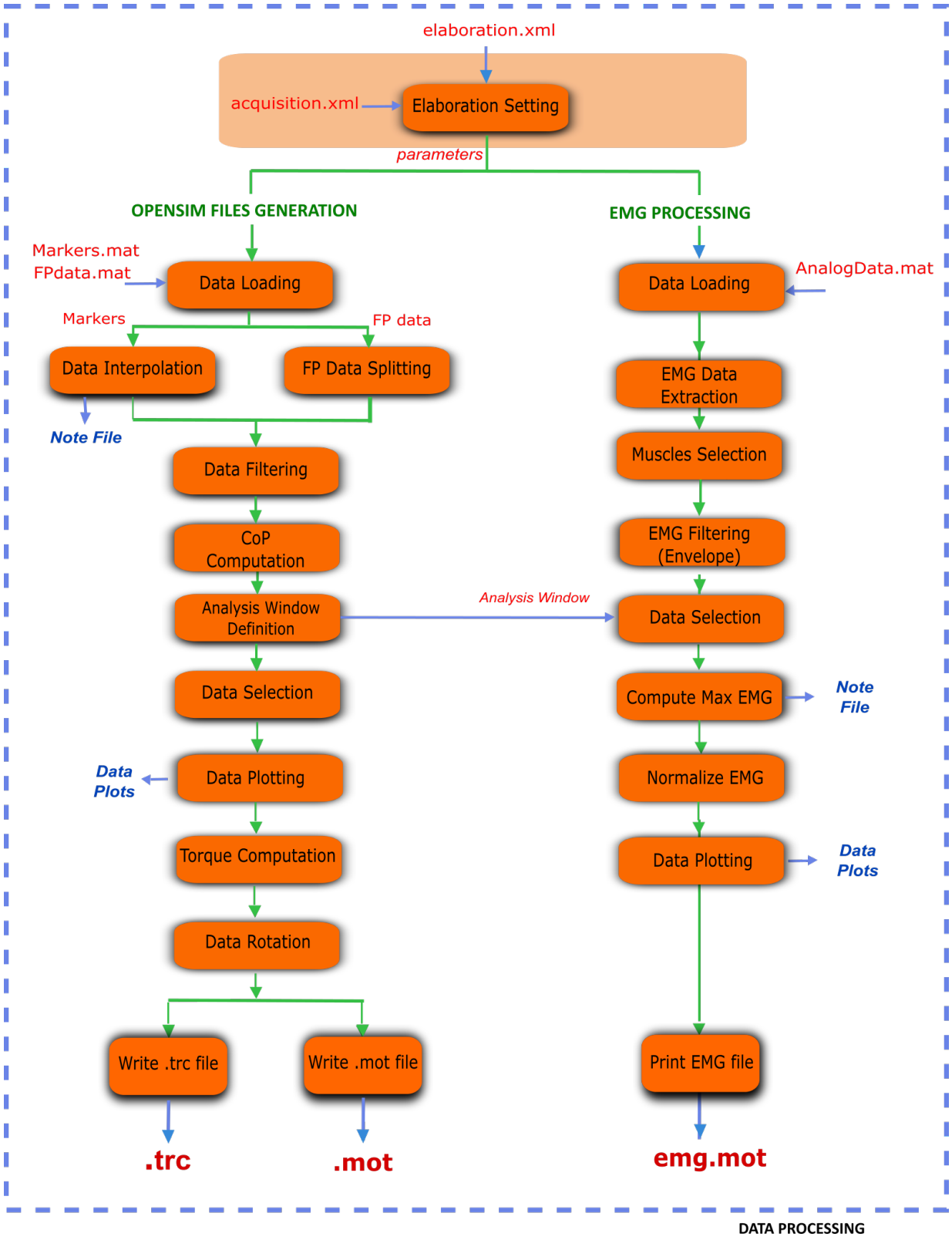

Fig. 9.1: Flowchart of steps in the Data Processing part.

9.3 Elaboration Interface: configure your elaboration

The execution of Data Processing block is univocally defined by a set of parameters selected by the user. All these parameters are saved in the `elaboration.xml` configuration file, which guarantees the high configurability and the fully reproducibility of the toolbox behavior.

The following listing shows an example of an `elaboration.xml` configuration file.

```
<?xml version="1.0" encoding="utf-8"?>
<elaboration>
<FolderName>.\InputData\UNIPDsubject\2014-06-09</FolderName>

<Trials>Walking1 Walking2 FastWalking1 FastWalking2 Running1</Trials>

<MarkersInterpolation>
   <MaxGapSize>15</MaxGapSize>
</MarkersInterpolation>

<Filtering>

  <Trial>
  <Name>Walking</Name>
  <Fcut>
    <Markers>8</Markers>
    <Forces>8</Forces>
    <CenterOfPressure>7</CenterOfPressure>
  </Fcut>
  </Trial>

  <Trial>
  <Name>FastWalking</Name>
  <Fcut>
    <Markers>10</Markers>
    <Forces>10</Forces>
    <CenterOfPressure>7</CenterOfPressure>
  </Fcut>
  </Trial>
  ....

</Filtering>

<WindowSelectionProcedure>
  <StanceOnFPfromC3D>
    <Leg>Right</Leg>
    <LabelForHeelStrike>Foot Strike</LabelForHeelStrike>
    <LabelForToeOff>Foot Off</LabelForToeOff>
    <Offset>20</Offset>
  </StanceOnFPfromC3D>
</WindowSelectionProcedure>

<Markers>C7 RA LA L5 RPSIS LPSIS RASIS LASIS RGT LGT RLE ...</Markers>

<EMGMaxTrials>FastWalking1 Running1 MVCadd MVCper</EMGMaxTrials>

<EMGsSelection>
  <EMGSet>UNIPD15nosideL-CEINMS</EMGSet>
  <EMGs>
    <EMG>
```

```
      <OutputLabel>bicfemlh_l</OutputLabel>
      <C3DLabel>Biceps femoris caput longus</C3DLabel>
    </EMG>
    ...
  </EMGs>
</EMGsSelection>

<EMGOffset>0.2</EMGOffset>

<OutputFileFormats>
  <MarkerTrajectories>.trc</MarkerTrajectories>
  <GRF>.mot</GRF>
  <EMG>.sto</EMG>
</OutputFileFormats>

</elaboration>
```

This configuration file can be obtained running a user-friendly graphical interface (GUI). The first thing the GUI will ask is to select the input data folder and next to enter an identificator for the current elaboration (Fig. 9.2). The identificator will be used to name a new folder in `ElaboratedData` storing the results of this elaboration (e.g., `elaboration01ID` and `elaboration02ID` in *Folders: organize your work figure*).

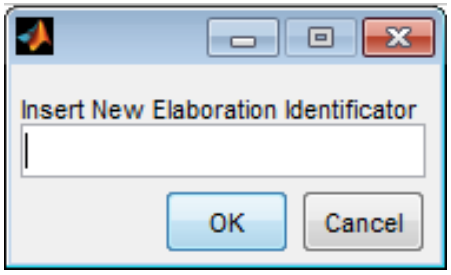

Fig. 9.2: Elaboration Interface: setting the Elaboration Identificator

Then, the GUI will ask the parameters required for the elaboration:

- Trials to be processed: a subset of the C3D files in the input folder
- Cutoff frequency for markers filtering specified for each trial type (walking, running,...) (Optional)
- Cutoff frequency for force filtering specified for each trial type (walking, running,..) (Optional)
- Cutoff frequency for CoP filtering depending on FP type (Optional)
- Analysis Window Computation method, with its own parameters (offset, C3D events labels, frames for the manual method). Available methods are further described in a dedicated section (Analysis Window Definition)
- List of markersto be written in `.trc` file

When EMG data are available:

- Trials for the computation of the maximum value of EMG signals
- Muscles to be considered in the processing (a subset of the acquired EMGs)
- EMG output labels. Different application names EMG signals differently. To avoid typing output labels several times, translation between EMG protocol labels (in C3D files) and application labels is stored in a setup file (see EMGs Labels) that can be selected from the GUI.

Besides, there are also a few additional parameters that can be set for each elaboration. As their default values usually accomplish a wide range of applications and needs, they are automatically assigned when running the GUI, so the

user is not asked to input them each time. However, if necessary, they can be manually modified directly in the `elaboration.xml` configuration file, once it has been created with the GUI and before running the elaboration. These parameters are:

- `MaxGapSize` (line 7-9)

  Marker trajectories are piecewise interpolated and each gap is considered for interpolation only if its size is smaller than the maximum gap size allowed (i.e., `MaxGapSize`). Otherwise, interpolation for that gap is not performed and a message is printed on the MATLAB Command Window. The `MaxGapSize` parameter is assigned according to the Video Frame Rate of your data. Default value is fixed to 15 frames at 60Hz [1]: the corresponding value for a different frame rate is computed and set. You can try different values for this parameter and check how it affects the interpolation process looking at the `InterpolationNote.txt` file.

- `EMGOffset` (line 58)

  MOtoNMS stores the EMG values starting `EMGOffset` second before the initial frame. While MOtoNMS has not been designed to synchronize markers, ground reaction forces and EMGs, it accounts for the need of follow-up applications to deal with the electromechanical delay of muscles providing EMG data from an anterior offset in time. Default value is set to 0.2 seconds. This setting does not allow to define analysis windows starting at the first frame. If you do not need to consider the electromechanical delay in your research, you can manually set `EMGOffset` to 0 in your `elaboration.xml` file. This also enables the elaboration of data from the first frame.

- `OutputFileFormats` (line 60-64).

  Starting from version 2.1, MOtoNMS allows also the definition of file formats for all the output (i.e. marker trajectories, GRF and EMG) within the `elaboration.xml` configuration file (lines 60-64). MOtoNMS currently supports a single choice for marker trajectories and GRF, i.e. `.trc` and `.mot` respectively, which are also the default formats. Nevertheless, the possibility to manage output file formats in the configuration file is a first step toward a future extension of available file formats. Conversely, EMG can already be stored differently according to user preference. Current available file formats are: `.mot` (SIMM and OpenSim motion, set as default), `.sto` (OpenSim storage), and plain text file (`emg.txt`).

9.4 Analysis Window Definition

Different methods to define the analysis window are available. If events are stored in C3D files, they may be selected as start and end frames of the analysis, otherwise desired frames can be inserted manually. An automatic detection of gait events is also possible based on a thresholding algorithm based on force plate data [2] . The following explains the available methods with additional details.

ComputeStancePhase: automatic detection of stance phase using a thresholding algorithm based on force plate data. This method needs only to know the leg of the stance. When only one leg is instrumented (see *EMGs Protocols*), this information can be automatically obtained from the `acquisition.xml` file. When both legs are instrumented, the user is asked to indicate the leg he/she wants to consider. Once that the leg is defined, the stance phase is selected based on force data from the FP struck by this leg (line 3 in the following listing). It is also possible to add an offset to the stance phase such that the analysis window will start offset frames before the heel strike and end offset frames after the toe off (line 2).

```
1 <ComputeStancePhase>
2   <Offset>20</Offset>
3   <Leg>Right</Leg>
4 </ComputeStancePhase>
5
6 Example of ComputeStancePhase tag in an elaboration.xml file
```

[1] <http://biomch-l.isbweb.org/threads/4734-Summary-Missing-Markers?highlight=markers+interpolation>
[2] Rueterbories J et al., Medical Engineering & Physics 32: 545-552,2010

StanceOnFPfromC3D:method for the selection of analysis window based on events stored in C3D files. It looks for the windows defined by the selected events and chooses the one when the instrumented leg struck the force plate.

Events must be specified by the user and correspond to the label used in the C3D files (*Elaboration Interface: configure your elaboration*). Instrumented leg can be selected from the EMG protocol (*EMGs Protocols*, line 11 in the following listing) or defined by the user when both limbs are instrumented.

Events must precisely define the stance window: MOtoNMS checks force platform values to ensure the validity of the events on the C3D files and errors may be raised. Finally, this method allows to add an offset to the selected events to consider a wider analysis window (line 5).

```
1 <StanceOnFPfromC3D>
2   <Leg>Right</Leg>
3   <LabelForHeelStrike>Foot Strike</LabelForHeelStrike>
4   <LabelForToeOff>Foot Off</LabelForToeOff>
5   <Offset>5</Offset>
6 </StanceOnFPfromC3D>
7
8 Example of StanceOnFPfromC3D tag in an elaboration.xml file
```

WindowFromC3D: this method allows to select any kind of events stored in the C3D files. It differs from the previous one since it does not focus on specific gait events such as foot strike or foot off. Thus, force data and the instrumented leg are not checked during the elaboration.

The identification of the right event in the C3D files requires that different labels are used for the automatic selection of events. The user is asked to state the full labels (i.e., the `Context`, which might be `Right, Left, or General`, + `Label`) associated to the start and stop events he/she wants to select for the Analysis Window (lines 2-3 in the following listing). `Offset` can also be applied to this method (line 4).

```
1 <WindowFromC3D>
2   <FullLabelForStartEvent>Right Foot Strike</FullLabelForStartEvent>
3   <FullLabelForStopEvent>Right Foot Off</FullLabelForStopEvent>
4   <Offset>0</Offset>
5 </WindowFromC3D>
6
7 Example of WindowFromC3D tag in an elaboration.xml file
```

Manual: a manual definition of frames to be considered for the elaboration has been implemented as well. It requires to specify start and end frame for each trial (lines 4-5 in the following listing). This gives the user the full freedom in the choice of the analysis window, but it has the drawback to be quite long and complex in the compilation of the `elaboration.xml` file, especially in the case of several trials, as start and end frame must be defined for each one.

```
1  <Manual>
2    <TrialWindow>
3      <TrialName>Walking1</TrialName>
4      <StartFrame>133</StartFrame>
5      <EndFrame>196</EndFrame>
6    </TrialWindow>
7    ...
8
9  </Manual>
10
11 Example of Manual tag in an elaboration.xmlfile
```

9.5 How to run the program

9.5.1 Create settings file for elaboration (Elaboration Interface)

1. Set MATLAB path on `src\DataProcessingfolder`
2. Run `ElaborationInterface.m`

Output: it generates `elaboration.xml` file, which will be saved in the elaboration folder. It also asks if the user wants to run the data processing code using parameters from the `elaboration.xml` file just created.

9.5.2 Run processing

If you already have the XML configuration file, you can skip the execution of the Elaboration Interface and run directly the code through the command:

```
runDataProcessing(ElaborationFilePath)
```

where `ElaborationFilePath` is the path of the folder where the `elaboration.xml` file you want to run is located.

Example

```
runDataProcessing(C:\MOtoNMSv10\TestData\ElaboratedData\GUsubject\date
\testRightStanceFromC3D)
```

Output: `.trc`, `.mot`, and `emg.mot|.sto|.txt` for each trial

Additional files are also created for further possible analysis:

- `InterpolationNote.txt` information about the marker interpolation step;
- filtered data within the analysis window and the analysis window frames in `.mat` format;
- Plot comparing raw and filtered force plate data within the analysis window;
- Plot of EMG raw;
- Plot of EMG envelopes within the analysis window for each muscle;
- `AllNormalizedEnvelopes.fig`: plot of normalized EMG linear envelopes of all the muscles;
- EMG selected for the elaboration: raw (`EMGs/Raw/EMGsSelectedRaw.mat`), linear EMG envelopes before (`EMGs/Envelope/EMGsSelectedEnvelope.mat`) and after normalization (`EMGs/Envelope/emg.mat`), in mat format;
- `maxemg/maxemg.txt` and `maxemg/maxemg.mat`, storing the maximum EMG value for each muscle, together with the trial and time in which they occur;
- EMG selected for the maximum EMG values computation: raw (`maxemg/EMGsRawForMax.mat`) and the corresponding linear envelopes (`maxemg/EMGsEnvelopeForMax.mat`);
- Plot of raw EMG, envelope and maximum EMG value for each muscle, corresponding to the trial where the maximum EMG value occurs.

Please pay ATTENTION to the following IMPORTANT NOTES:

Warning:
- C3D files MUST be converted in mat format before the execution of Data Processing (*C3D2MAT*).
- Input data folder requires an `acquisition.xml` file. This can be generated with `mainAquisitionInterface.m` (*Acquisition Interface*).
- Static trial MUST NOT be selected: Data Processing is only for dynamic trials. To process static trials use Static Elaboration.
- Selection of the first 0.2 sec is NOT possible due to the `EMGOffset`. Be careful to select an Analysis Window with at least 0.2 sec time before the start frame. This value can be changed in the `elaboration.xml` file.
- If your application does not need data to manage the electromechanical delay, you can set the `EMGOffset` to 0 in your `elaboration.xml` file.

9.6 Setup Files

The only setup file required for this step is about the EMGs. Thus, if you didnt collect EMGs, you can skip this part.

9.6.1 EMGs Labels

If you gathered EMG signals, probably you need to process them and use the results of the processing for some investigations and/or other applications. Some applications require predefined names for their input data. For example, CEINMS software used fixed labels for each EMG signal in input, that usually differ from those assigned during the acquisition session. Therefore, after the execution of Data Processing, you would be required to manually change the original EMG labels stored in your results according to the needs of CEINMS or any other application you want to use.

MOtoNMS allows you to avoid this tedious manual process, and does it for you. The toolbox can save results changing EMG labels coming from the acquisition (stored in the C3D files) in those desidered by the user. You just have to specify the labels you need in your results, and the `emg.txt` output file will be saved using them.

However, you are not asked to input this information, usually common for a set of elaboration, each time you are running the Elaboration Interface GUI. It would be time consuming, boring and therefore error prone, especially considering the number of EMG signals that can be acquired during an acquisition. Instead, this association between EMG labels stored in C3D files and those required for an application is defined in a XML setup file, that can be selected through the GUI.

This file MUST be saved in `SetupFiles\DataProcessing\EMGsLabels\` and named with the sequence of EMG protocol and application names. (e.g., `UNIPD15nosideL-CEINMS.xml`). The name stands for the EMG labels collected in a certain laboratory that must be translated in those required for a certain application.

An example of how to compile it is shown in the following listing (available at: `SetupFiles\DataProcessing\EMGsLabels\UNIPD15nosideL-CEINMS.xml`). You have to manually edit this file but it is fairly easy and you can check your file with respect to the required syntax with the validation procedure (see *Appendix B: Validation of Setup and Configuration Files*).

```
<EMGSet>
  <EMG>
    <OutputLabel>bicfemlh_l</OutputLabel>
    <C3DLabel>Biceps femoris caput longus</C3DLabel>
  </EMG>
  <EMG>
    <OutputLabel>gaslat_l</OutputLabel>
    <C3DLabel>Gastrocnemius lateralis</C3DLabel>
  </EMG>
```

```
  <EMG>
    <OutputLabel>gasmed_l</OutputLabel>
    <C3DLabel>Gastrocnemius medialis</C3DLabel>
  </EMG>
  <EMG>
    <OutputLabel>gmax_l</OutputLabel>
    <C3DLabel>Gluteus maximus</C3DLabel>
  </EMG>
  .......
</EMGSet>
```

CHAPTER

TEN

STATIC ELABORATION: PROCESS YOUR STATIC TRIALS

OpenSim requires static trials for the scaling procedure [1]. The objective of the elaboration of these trials is to produce for each static trial a markers trajectories file (`.trc`), optimized for the Scaling in OpenSim.

Starting from the `.mat` structures created by *C3D2MAT*, Static Elaboration apply the same filtering procedure used in Data Processing. The main step is then joint centers (JC) estimation. These are points usually recommended to improve the accuracy of the OpenSim scaling procedure.

In the current toolbox version, methods for hip (HJC), knee (KJC), ankle (AJC), elbow (EJC), shoulder (SJC) and wrist (WJC) joint centers computation are available. More specifically, HJC can be assessed exploiting Harrington [2], while the others can be computed as the mid points between anatomical landmarks specified in the corresponding setup files (*Setup Files*). However, the structure of the software intentionally allows an easy integration of other algorithms for both lower and upper limbs joints. The resulting trajectories are then added to the markers list defined by the user (*Static Interface: configure your elaboration*). The updated list is finally rotated into OpenSim reference system and stored in the output `.trc` file.

10.1 Static Interface: configure your elaboration

As Data Processing (*Elaboration Interface: configure your elaboration*), the execution of the Static Elaboration block is fully defined by a set of parameters selected by the user.

All these parameters are enclosed in the `static.xml` configuration file, which can be obtained running a graphical user interface.

The GUI will ask first to select the input data folder and next to enter an identifier for the current elaboration. The identifier will be used to name the new folder storing the results of this elaboration (e.g. `staticelaboration01ID`, `staticelaboration02ID` in *Folders: organize your work figure*). As shown in *Folders: organize your work figure*, all static elaborations are grouped in the `staticElaborations` folder to avoid confusion with the output folders of Data Processing.

Then, the GUI will ask the user the parameters required for the elaboration, which are briefly described in the following:

- Static trial to be processed
- Cut-off frequency for markers filtering (Optional)
- Joints for center computation (Optional)
- Methods for the computation of the joint centers, which are defined selecting the corresponding setup file (*Setup Files*)
- List of markersto be written in `.trc` file

[1] <http://simtk-confluence.stanford.edu:8080/display/OpenSim/Getting+Started+with+Scaling>
[2] Harrington ME, et al., Journal of Biomechanics,Biomechanics 40:595602, 2007

An example of `static.xml` configuration file is shown in the following:

```
<?xml version="1.0" encoding="utf-8"?>

<static>
  <FolderName>.\InputData\UWAsubject\2010-05-11\</FolderName>
  <TrialName>Static1</TrialName>
  <Fcut>8</Fcut>
  <JCcomputation>
    <Joint>
      <Name>Ankle</Name>
      <Method>AJCMidPoint</Method>
      <Input>
        <MarkerNames>
          <Marker>LLMAL</Marker>
          <Marker>RLMAL</Marker>
          <Marker>LMMAL</Marker>
          <Marker>RMMAL</Marker>
        </MarkerNames>
      </Input>
    </Joint>
    <Joint>
      <Name>Hip</Name>
      <Method>HJCHarrington</Method>
      <Input>
        <MarkerNames>
          <Marker>LASI</Marker>
          <Marker>RASI</Marker>
          <Marker>LPSI</Marker>
          <Marker>RPSI</Marker>
        </MarkerNames>
      </Input>
    </Joint>
    <Joint>
      <Name>Knee</Name>
      <Method>KJCMidPoint</Method>
      <Input>
        <MarkerNames>
          <Marker>LeLFC</Marker>
          <Marker>RiLFC</Marker>
          <Marker>LeMFC</Marker>
          <Marker>RiMFC</Marker>
        </MarkerNames>
      </Input>
    </Joint>
  </JCcomputation>
  <trcMarkers>C7 CLAV LACR LASH LPSH LUA1 LUA2 LUA3 ....</trcMarkers>
</static>
```

Before going into details about this file we need to give you some additional information about JC computation. Different methods for the computation of the different JC may require different input data. Usually these input data includes the markers position during the static acquisition. However, different marker protocols use different labels to identify the same body landmark. Thus, it is necessary to define the connection between the marker required by a method (i.e. the left and right anterior and posterior superior iliac spine forthe Harrington method) and their names according to the markers protocol used for the data collection. As it would be too complex and error prone to do it for each elaboration, this information is stored in a setup file (see *Setup Files* in this chapter), one for each JC computational method. Each file describes how the landmarks of interested are named in different marker protocols. A user can add new protocols to the file when required.

Static Elaboration retrieved from the Setup File of the selected JC computation method the marker labels in the marker protocol used in the static trial selected for the processing and save them in the `static.xml` file (lines 12-17, 24-29, and 36-41).

The list of markers to be stored in the `.trc` file (line 45) MUST NOT include the estimated JC labels, as it will be automatically updated.

10.2 How to run the program

10.2.1 Create settings file for static elaboration (Static Interface)

1. Set MATLAB path on `src\StaticElaborationfolder`
2. Run `StaticInterface.m`

`StaticInterface.m` is the program implementing the Static Interface.

Output: it generates `static.xml` file.

At the end of the program the user is prompt with the request if he/she wants to run the elaboration code with the just created `static.xml` file.

10.2.2 Run static elaboration

If you have already the configuration file with the parameters of your elaboration (`static.xml`) , you can run directly the static elaboration with the command:

```
runStaticElaboration(ConfigFilePath)
```

where `ConfigFilePathis` the full path of the folder where your `static.xml` file is located.

Output: `static.trc`, with the processed markers trajectories and the computed JC

Additional files are also generated to help in validation of obtained results:

- computed joint centers coordinate in .mat format
- plot of estimated JCs in the laboratory reference system

Please pay ATTENTION to the following IMPORTANT NOTES:

Warning:
- For any JC computation method, a setup file MUST be predefined (see *Setup Files*).
- The marker protocol used in the data collection of the static trial must be among the one in the setup file for the selected JC computation method (see *Setup Files*).
- Be careful to specify list of markers within the `MarkerNames` tag in the same order of the list of markers within the `MarkersFullNames` tag (lines 5-10 in the next listing).
- The plot of estimated JCs is based on data from the first frame: if JCs plot seems wrong there could be a problem on data in the first frame
- *C3D2MAT* code MUST be run on the static trial before the static elaboration.

10.3 Setup Files

The main information that the user have to define for a static elaboration is the joint centers he/she is interested in computing, the methods to be used for their computation and the parameters each method requires. The last one might

be very long and error prone to be edited at each elaboration, so we decided that it would be easier to enclose this parameters in a setup file. The Static Interface will thus ask you to select a setup file for each JC computation. The following explains how to fill these setup files.

10.3.1 Joint Center Computation Methods

Each implemented JC computation method requires a file that list how the required input markers are labelled in each marker protocol. This file MUST be saved in `SetupFiles\StaticElaboration\JCcomputation\` within the folder of the corresponding joint, as shown in Fig. 10.1. While not mandatory, our suggestion is to name the file with the acronym of the joint name followed by the identifier of the JC computation method. When only a method is available for a joint, the Static Interface will not ask to choose the setup file and select the only one available in the folder.

The following listing is an example of a setup file created for the Harrington method at the hip joint (available at: `SetupFiles\StaticElaboration\JCcomputation\Hip\HJCHarrington.xml`)

```
1  <Method>
2    <Name>HJCHarrington</Name>
3
4    <Input>
5      <MarkerFullNames>
6        <Marker>Left Anterior Superior Iliac Spine</Marker>
7        <Marker>Right Anterior Superior Iliac Spine</Marker>
8        <Marker>Left Posterior Superior Iliac Spine</Marker>
9        <Marker>Right Posterior Superior Iliac Spine</Marker>
10     </MarkerFullNames>
11   </Input>
12
13   <MarkersDefinition>
14     <Protocol>
15       <Name>UWA-Fullbody</Name>
16       <MarkerNames>
17         <Marker>LASI</Marker>
18         <Marker>RASI</Marker>
19         <Marker>LPSI</Marker>
20         <Marker>RPSI</Marker>
21       </MarkerNames>
22     </Protocol>
23
24     <Protocol>
25       <Name>UNIPD_CASTforOpenSim</Name>
26       <MarkerNames>
27         <Marker>LASIS</Marker>
28         <Marker>RASIS</Marker>
29         <Marker>LPSIS</Marker>
30         <Marker>RPSIS</Marker>
31       </MarkerNames>
32     </Protocol>
33     ...
34   </MarkersDefinition>
35 </Method>
```

The file first lists the input required for the method (`<Input>` tag, lines 4-11). Required markers are listed with their full name within the `<MarkersFullNames>` tag (lines 5-10). Then, a tag named `MarkersDefinition` follows (line 13-34). It consists of a list of protocols and MUST include, for each of them, the corresponding input markers names. When new protocols are available, they must be added to the list of the method used for joint computation.

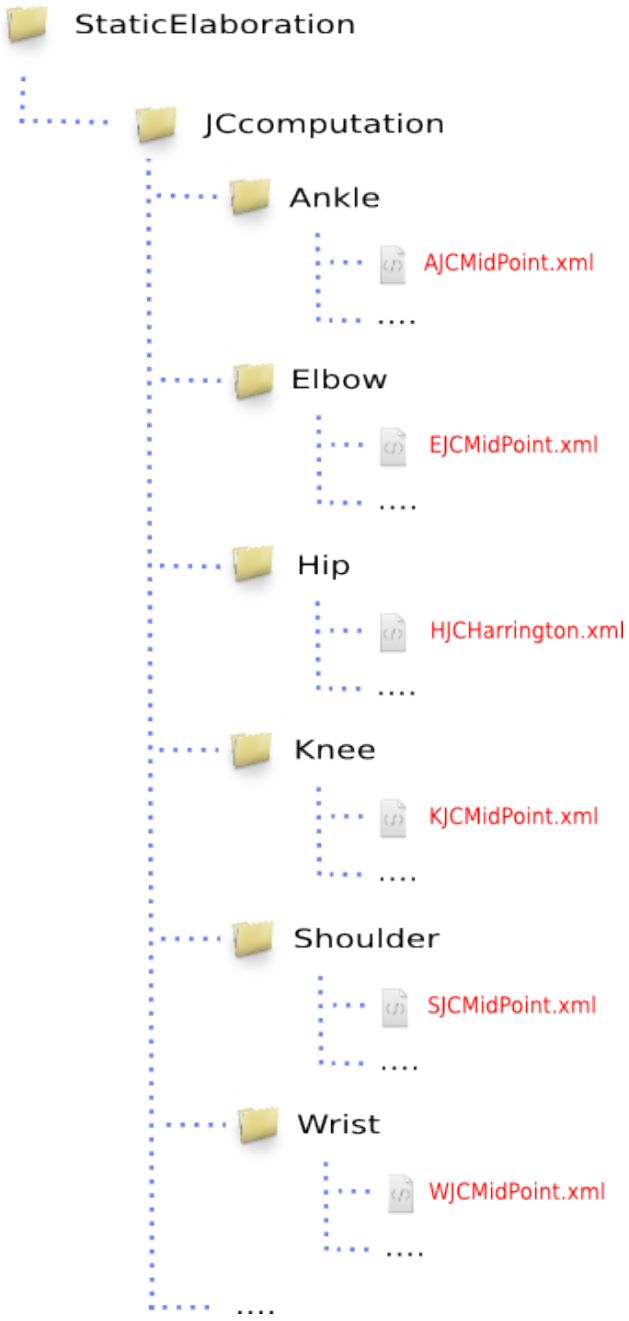

Fig. 10.1: Setup files organization for the Static Elaboration code.

CHAPTER

ELEVEN

ERROR MESSAGES

A complete section dedicated to handled error messages will be available in the next version of this User Manual.

Meanwhile, for any doubt or problem, please refer to the MOtoNMS Public Forum, available from the SimTK project page (https://simtk.org/home/motonms), or write an email to Alice Mantoan, ali.mantoan@gmail.com.

In case of error, a useful function for debugging is `save_to_base()`, with input argument set to 1 (`save_to_base(1)`). It copies all variables in the calling function to the base MATLAB workspace.This allow to check the internal variables from the MATLAB command prompt after the calling function ends.

CHAPTER

TWELVE

APPENDIX A: SETUP FILES

This appendix lists the setup files already available in MOtoNMS distribution. They are stored in `MOtoNMS\SetupFiles\` folder (*Code organization figure*). They are selected by the users through the GUIs to create the configuration files (`acquisition.xml`, `elaboration.xml` and `static.xml`) required for processing.

12.1 Acquisition Interface

Laboratories

- `UNIPD.xml`
- `UMG.xml`
- `GU.xml`
- `UWA.xml`

Markers Protocols

- `UNIPD_CASTforOpenSim.xml`
- `GU-10pointsCluster.xml`
- `UMG-OpenSim.xml`
- `UWA-Fullbody.xml`

EMGs Protocols

- `UNIPD-15noside-left.xml`
- `GU-16muscles.xml`
- `UWA-16muscles-r.xml`

12.2 Data Processing

EMGs Labels

- `UNIPD15nosideL-CEINMS.xml`
- `UNIPD15nosideL-OpenSim.xml`
- `GU-CEINMS.xml`
- `UWA-CEINMS.xml`

12.3 Static Elaboration

JC computation

Ankle

- `AJCMidPoint.xml`

Elbow

- `EJCMidPoint.xml`

Hip

- `HJCHarrington.xml`

Knee

- `KJCMidPoint.xml`

Shoulder

- `SJCMidPoint.xml`

Wrist

- `WJCMidPoint.xml`

CHAPTER

THIRTEEN

APPENDIX B: VALIDATION OF SETUP AND CONFIGURATION FILES

This section is only for people that needs to create new setup files, or want to manually change their configuration files. Please, try to use the graphical user interfaces and avoid playing directly with the XML configuration files. It took us a lot of time to implement them, therefore make us happy and use them :).

The only real reason to play directly with the XML files is when you start doing something new: (1) setup a new laboratory, or a new protocol either for (2) EMGs or (3) makers or (4) you need different output labels for your processed EMGs, or maybe you want to introduce a new way to (5) compute joint center.

In all these case, you need to manually create new setup files.

Usually these XML files are really simple and you can copy one already available and easily understand what you need to change. But when you are done, it is a good practice to check the syntax of your XML file against the grammar. Again, as it took us quite a lot to develop a grammar for each possible XML file, please make us happy and use it. Additionally, this is also really helpful for you as you can be sure that your file is syntactically correct and ready to be used in MOtoNMS.Indeed, errors in editing the setup files result in execution errors when running the source code; these may not be easy to understand if you are not an expert of MATLAB language and MOtoNMS behavior.

There are many possible tools that you can use. We just suggest a couple of the easiest to be used because everything is online and you do not have to install anything on your computer.

Choose one of the following links:

http://www.freeformatter.com/xml-validator-xsd.html

http://www.corefiling.com/opensource/schemaValidate.html

and upload your XML file and the corresponding XMLSchema (the `.xsd` file).

The following tables are listing the XML Schema for each type of XML setup and configuration files that you find in MOtoNMS.

XML Setup Files	XML Scheme
GU-16muscles.xml UNIPD-15noside-left.xml UWA-16muscles-r.xml	EMGsProtocol.xsd
GU.xml UMG.xml UNIPD.xml UWA.xml	Laboratory.xsd
GU-10pointsCluster.xml UMG-OpenSim.xml UNIPD_CASTforOpenSim.xml UWA-Fullbody.xml	MarkersProtocol.xsd
UNIPD15nosideL-CEINMS.xml UNIPD15nosideL-OpenSim.xml GU-CEINMS.xml UWA-CEINMS.xml	EMGLabels.xsd
AJCMidPoint.xml EJCMidPoint.xml HJCHarrington.xml KJCMidPoint.xml SJCMidPoint.xml WJCMidPoint.xml	JCcomputation.xsd

XML Configuration Files	XML Scheme
acquisition.xml	acquisition.xsd
elaboration.xml	elaboration.xsd
static.xml	static.xsd

CHAPTER

FOURTEEN

APPENDIX C: REVISION HISTORY

RELEASE 2.2 (*June 5, 2015*)

New Features

- Processing of data along different motion directions with respect to the laboratory reference system (`forward`, `backward`, `90right`, `90left`)
- Computation of CoP coordinates for force platforms with pads
- Possibility to skip the computation of joint centers in `StaticElaboration`
- Interpolation of gaps in marker trajectories with size less than `MaxGapSize` (chosen by the user in the `elaboration.xml` file)
- Piecewise filtering for marker trajectories having NaN values due to a missed interpolation
- Inclusion of EMG labels while saving EMG data in `.mat` format
- Definition of a new output folder (`maxemg`) for each dynamic elaboration, with plots and log data related to the computation of maximum EMG values
- Storing of all raw EMGs selected for the computation of maximum EMG values and the corresponding envelopes as `.mat` files
- Addition of information about the trial and the time corresponding to each maximum EMG value, when printing the `maxemg.txt` output file and logging in `.mat` format (`maxemg.mat`)
- Plot of raw EMG and envelope for each muscle, corresponding to the trial where the maximum EMG value occurs
- Availability of multiple formats documentation (GitHub Project Pages)
- Compatibility with MATLAB R2014b

Code Changes

- Added optional `MotionDirection` element in `acquisition.xsd` to support trials with different directions of motion
- Added optional `PadThickness` element in `laboratory.xsd` and `acquisition.xsd`, to account for plate padding in the computation of CoP coordinates
- Added `MarkersInterpolationType` with `MaxGapSize` element in `elaboration.xsd`, to let the user define the gaps' maximum size for the interpolation of marker trajectories
- Modified identification of the first and last frame for marker trajectories: added case of NaN values when markers are initially not visible
- Added possibility to disable warnings from BTK tool in `C3D2MAT_btk`

- Moved saving of `maxemg.txt` output file inside `maxemg` folder
- Handled error that can occur if input C3D file names do not include the repetition number (as required)
- Handled error in Y axis scale setting in `EnvelopePlotting.m`
- Renamed `CHANGES.txt` to `CHANGES.md`

Bug Fixes

- Fixed reading of C3D files without data from force platforms (FP)
- Fixed handling of FP data when a laboratory has more than 2 FPs of different types
- Fixed selection of `Leg on ForcePlatform` in `AcquisitionInterface` when there are more than 2 FPs in the laboratory
- Fixed definition of `timeStartFrame` and `timeEndFrame` in `selectionData.m` to account for an initial starting condition of t=0 and frame number=1. Fixed accordingly the definition of `frameArray` in `writetrc.m`
- Fixed computation of the hip joint center (HJC) with the Harrington method (`HJCHarrington.m`) when the input static file has a frame number lower than 3

RELEASE 2.1 (*September 8, 2014*)

New Features

- Compatibility with MacOS X operating systems
- Envelope plots with normalization scale (% max)
- Plot of normalized EMG linear envelopes for all the muscles
- `.sto` (OpenSim storage) file format for EMG output
- `.mot` (SIMM and OpenSim motion) file format for EMG output (new default)

Code Changes

- Changed `elaboration.xsd` to add support of different output file formats, preserving compatibility with previous versions.
- Renamed `mainStaticElaboration.m` as `StaticInterface.m`
- Moved main programs (`C3D2MAT.m`, `ElaborationInterface.m`, `StaticInterface.m`) to functions
- Moved internal functions in private folders
- Renamed all `readme.txt` to `README.md`
- Modified y axis scale setting of envelope plots
- Modified data storage structure: added `dynamicElaborations` folder to group all the multiple executions of `DataProcessing`

Bug Fixes

- Removed addition of mean values after EMG filtering
- Fixed units in EMGs plotting

- Fixed x label of envelope plots
- Fixed trial type identification for filtering cutoff definition

RELEASE 2.0 (*May 9, 2014*)

New Features

- Support to MATLAB 64 bit and multiplatform (`C3D2MAT` based on BTK)
- EMG selection using Analog Labels from each C3D input file
- Shoulder, elbow, and wrist JC computation for static trials, and examples of setup files for Griffith University markerset
- Missing values for markers trajectories identified by NaN instead of 0 in `.trc` output files

Code Changes

- Added `src/shared` folder to store functions common to several steps
- Modified filtering of markers trajectories: they are filtered only when visible and only if they have no gaps (`DataFiltering.m`, `ZeroLagButtFiltfilt.m`)
- Modified filtering of GRF data from type 1 force platform: filtering is applied only to non zero values to avoid smoothing due to zero values (data from force platform of type 1 are stored in C3D files after thresholding)
- Modified data interpolation: markers trajectories are interpolated only if gaps of consecutive frames are shorter than a fixed number defined according to the video frame rate (`DataInterpolation.m`)
- Modified retrieval of `AnalogData` in `C3D2MAT`: removed assumption of analog data stored only in analog channels subsequent to those dedicated to force data. Now they can be stored in any analog channel independently from force data.
- Renamed `replaceWithNans.m` as `replaceMissingWithNaNs.m`
- Renamed `matfiltfilt2.m` as `ZeroLagButtFiltfilt.m`
- Removed warning messages caused by the lack of subject's first and last names when loading a predefined `acquisition.xml`
- Added last selected folder in text fields of graphical user interfaces (GUIs)

Bug Fixes

- Modified transformation of COP coordinates from local to global reference system: translation added only for non zero values.
- User is not required to set a new identifier each time he/she load an already available `elaboration.xml` file as in version 1.0.
- Changed the definition of the interval where markers are visible in `replaceMissingWithNaNs.m` (the definition of var 'index')
- Fixed the computation of the hip joint center (HJC) with the Harrington method (`HJCHarrington.m`)

RELEASE 1.0 (*February 17, 2014*)

Initial Release