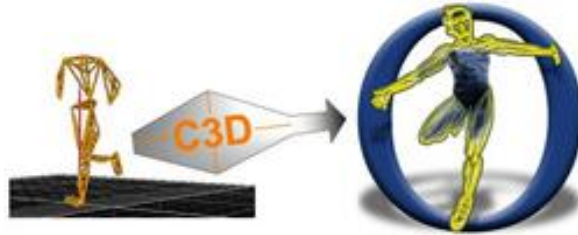


# GaitExtract Toolbox V1.71

## *User Manual*



Tim Dorn

[twdorn@stanford.edu](mailto:twdorn@stanford.edu)

<https://simtk.org/home/c3dtoolbox>

---

## Table of Contents

<b>1. INTRODUCTION.....</b>	<b>3</b>
<b>2. THE BASIC IDEA .....</b>	<b>4</b>
<b>3. TOOLBOX INSTALLATION .....</b>	<b>4</b>
<b>4. LABORATORY CONFIGURATION .....</b>	<b>5</b>
4.1. DATA EXTRACTION LABELS (SECTION 2 OF LOADLABELS.M) .....	5
4.2. FORCE PLATE IDENTIFICATION .....	6
4.3. ANALOG EMG CONFIGURATION .....	9
4.4. COORDINATE SYSTEMS .....	10
4.5. CENTER OF PRESSURE CALCULATION .....	12
4.6. SETTING UP ALTERNATE LABS (SECTION 1 OF LOADLABELS.M) .....	15
<b>5. EVENT LABELING .....</b>	<b>18</b>
5.1. FOOT DETECTION ALGORITHM.....	19
<b>6. DETAILED FUNCTION LIST.....</b>	<b>22</b>
BATCHEMGPROCESS(C3DKEY, EMGSETNAME, EMGPROCESSTASKS, FILESUFFIX*) .....	23
CREATEEVENT(C3DFILE, FOOT, LABEL, FRAME) .....	24
EXTRACTMOTFILE(PROCESSTASK1, VALUE1, PROCESSTASK2, VALUE2, ...) .....	25
GENERATEMOTFILE(DATAMATRIX, COLNAMES, FILENAME).....	26
GENERATETRCFILE(C3DKEY, MARKERPOS, MARKERSET).....	27
GETEVENTS(C3DFILE, DIRECTION*, FP_ORDER*, FP_SEQUENCE*).....	28
GETKINETICS(C3DKEY, PLOTTOG*, FILTERFREQ*, MARKERSDYN*).....	32
GETMARKERS(C3DKEY, MARKERSETNAME, FILTER*) .....	34
GETMVC(C3DKEY, EMGSETNAME, WINDOWSIZE, *MVCMETHOD) .....	36
LOADLABELS() .....	37
MULTIPLEMGPROCESS(C3DFILE, EMGSETNAME, EMGPROCESSTASKS, INTERVAL4TIME) .....	38
PROCESSEMG(EMGVEC, {PROCESSTASK1, VALUE1, PROCESSTASK2, VALUE2, ...}) .....	39
WRITEXML(TYPE, C3DKEY*) .....	41
<b>7. EXAMPLE SUBJECT TRIAL .....</b>	<b>42</b>
<b>8. REVISION INFORMATION .....</b>	<b>43</b>

Copyright (c) 2008 Tim Dorn

Use of the GaitExtract Toolbox is permitted provided that the following conditions are met:

1. The software is not distributed or redistributed. Software distribution is allowed only through <https://simtk.org/home/c3dtoolbox>.
2. Use of the GaitExtract Toolbox software must be acknowledged in all publications, presentations, or documents describing work in which the GaitExtract Toolbox was used.
3. Credits to developers may not be removed from source files
4. Modifications of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

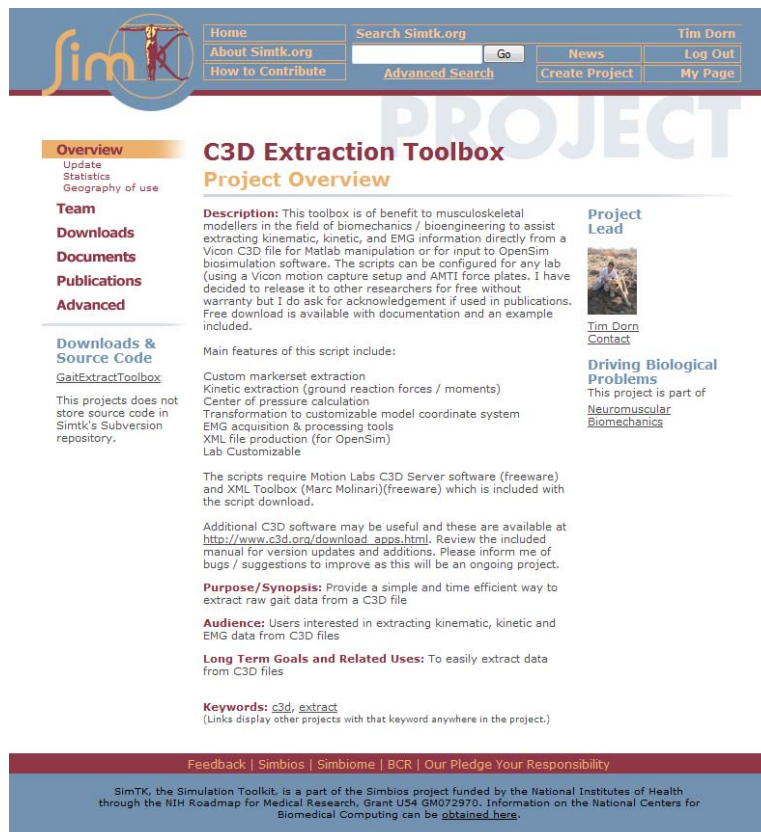
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR BUSINESS INTERRUPTION) OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# 1. Introduction

This toolbox is primarily used for extracting and processing experimental gait data into a format that can be directly used with OpenSim. It can be configured for any laboratory setup, specifically in terms of the coordinate systems used, force plate names, marker sets, EMG channels, etc. Use of this toolbox for specific configurations / laboratories will require changes to the parameters in *loadLabels.m*. In this manual, we assume that:

- Matlab is installed (R2007a or later)
- Vicon Nexus or Vicon Workstation was used to acquire experimental gait data. While this is not strictly required for use of the toolbox, the user manual will describe process assuming that Vicon is being used.

Extraction is performed directly from the C3D file of trial, which is easily obtained from the data acquisition software (i.e. Vicon), so data can be processed offline on any computer with Matlab. The GaitExtractToolbox is open-source software and is freely available from <https://simtk.org/home/c3dtoolbox> (Figure 1).



The screenshot shows the SimTK website interface for the C3D Extraction Toolbox project. The page features a navigation bar with links for Home, About Simtk.org, How to Contribute, Search Simtk.org, News, Log Out, Create Project, and My Page. The main content area is titled "C3D Extraction Toolbox Project Overview" and includes a description of the toolbox's purpose, a list of main features, and a project lead section. The footer contains a navigation bar with links for Feedback, Simbios, Simbiome, BCR, and Our Pledge Your Responsibility, along with a note about the project's funding by the National Institutes of Health.

**SimTK** Home Search Simtk.org Tim Dorn  
About Simtk.org Go News Log Out  
How to Contribute Advanced Search Create Project My Page

### Overview

Update  
Statistics  
Geography of use

### Team

### Downloads

### Documents

### Publications

### Advanced

### Downloads & Source Code

[GaitExtractToolbox](#)

This projects does not store source code in Simtk's Subversion repository.

## C3D Extraction Toolbox

### Project Overview

**Description:** This toolbox is of benefit to musculoskeletal modellers in the field of biomechanics / bioengineering to assist extracting kinematic, kinetic, and EMG information directly from a Vicon C3D file for Matlab manipulation or for input to OpenSim biosimulation software. The scripts can be configured for any lab (using a Vicon motion capture setup and AMTI force plates. I have decided to release it to other researchers for free without warranty but I do ask for acknowledgement if used in publications. Free download is available with documentation and an example included.

**Main features of this script include:**

- Custom markerset extraction
- Kinetic extraction (ground reaction forces / moments)
- Center of pressure calculation
- Transformation to customizable model coordinate system
- EMG acquisition & processing tools
- XML file production (for OpenSim)
- Lab Customizable

The scripts require Motion Labs C3D Server software (freeware) and XML Toolbox (Marc Molinar)(freeware) which is included with the script download.

Additional C3D software may be useful and these are available at [http://www.c3d.org/download\\_apps.html](http://www.c3d.org/download_apps.html). Review the included manual for version updates and additions. Please inform me of bugs / suggestions to improve as this will be an ongoing project.

**Purpose/Synopsis:** Provide a simple and time efficient way to extract raw gait data from a C3D file

**Audience:** Users interested in extracting kinematic, kinetic and EMG data from C3D files

**Long Term Goals and Related Uses:** To easily extract data from C3D files

**Keywords:** c3d, extract  
(Links display other projects with that keyword anywhere in the project.)

**Project Lead**

[Tim Dorn](#)  
Contact

**Driving Biological Problems**

This project is part of [Neuromuscular Biomechanics](#)

Feedback | Simbios | Simbiome | BCR | Our Pledge Your Responsibility

SimTK, the Simulation Toolkit, is a part of the Simbios project funded by the National Institutes of Health through the NIH Roadmap for Medical Research, Grant U54 GM072970. Information on the National Centers for Biomedical Computing can be [obtained here](#).

Figure 1: GaitExtractToolbox on the website: <https://simtk.org/home/c3dtoolbox>

## 2. The basic idea

Although there are many software packages out there to extract and process experimental gait data from Vicon, much of the software is hard coded and the lack of flexibility does not suit engineering applications. The basic purpose of this toolbox is to provide a platform to extract gait data directly from a C3D file into to a Matlab environment where higher levels of flexibility are available for data processing. Data can then be exported into \*.trc, \*.mot and \*.sto files to be used with musculoskeletal models in OpenSim.

The basic idea is to first experimentally record the kinematics, kinetics, and (optional) EMG of a dynamic activity through using motion capture cameras, force plates, and EMG recording units. The resulting trial is stored (by Vicon) as a file with a C3D extension. A working directory can be created with a copy of the C3D file inside, and a simple high level Matlab m script, which defines the processing commands that define operations to be performed on the data. The *getEvents.m* function must be executed for each C3D file to create an ‘event key’ which can then be used to extract and crop experimental data. See Section 5 for a complete list of functions that can be applied to raw data. Feel free to modify these functions or make up your own. Please cite the following reference if used in publications.

*Dorn, T. W. (2008). Gait Extract Toolbox for Matlab, Version 1.71*

## 3. Toolbox Installation

The installation requires the Matlab path to point to the directory where you have decided to install this toolbox. It also requires the installation of Motion Labs<sup>®</sup> C3D Server (C3D Viewer should also be installed and a freeware copy is provided with the toolbox). The installation m file provided will install the scripts and set the paths for when Matlab loads. To install the toolbox:

- 1) Copy the toolbox to the directory you want to install it to (i.e. C:\GaitToolbox)
- 2) Load Matlab
- 3) Set current directory to **GaitExtractToolbox\INSTALL**
- 4) Run **installGaitExtractToolbox.m**
- 5) Follow the prompts

An example is provided to test the success of the installation (see Section 7).

## 4. Laboratory Configuration

I primarily designed this toolbox to run in the Biomechanics Laboratory in the Department of Mechanical Engineering at The University of Melbourne. It is however, simple to configure for any gait lab configuration. There are infinite ways to configure a biomechanics lab in regards to, coordinate systems, force plate configurations, marker set names, and other analog channel names. Just like Vicon needs information about the laboratory to configure its parameters, so does this toolbox. Once the lab is configured, the toolbox becomes a powerful data extraction and analysis tool and can export directly into a format to be used by OpenSim (with XML setup files). This section will outline the way the toolbox has been created.

*Note: In this manual, the LABORATORY coordinate system is referred to as the VICON coordinate system.*

### 4.1. Data Extraction Labels (Section 2 of loadLabels.m)

All labels / configuration parameters are stored in the **loadLabels.m** file. This labels file is used by many of the Matlab functions that plot graphs, and load C3D data. The labels are all stored as a structure called **glab**, standing for “global labels” and its meanings are illustrated below.

#### DO NOT MODIFY

- glab.name:** *Titles of each of the major labels*
- glab.dir:** *Labels for the XYZ directions of force*
- glab.S:** *Convention for ground reaction force (GRF) output*
- glab.X:** *Convention for center of pressure (CoP) output*
- glab.Mo:** *Convention for ground reaction moment about plate origin (GRMo) output*
- glab.Mx:** *Convention for ground reaction moment about CoP (GRMx) output*

#### USER DEFINED (USER SHOULD MODIFY THESE)

- glab.[jointModel]:** *Skeletal model joint labels*
- glab.[emgSet]:** *Analog channel names for EMG extraction. These names must match the Vicon analog channel names otherwise EMG extraction won't work (see Section 4.3).*

**glab.[emgProcess]:** *A cell of EMG processing tasks to define the order of EMG processing. These names must match the labels given in processEMG.m otherwise EMG extraction won't work (see processEMG.m help).*

**glab.[markerSet]:** *Markers in the order for kinematic position extraction.*

You may add other labels as needed. The benefit of storing these labels in one script is to keep the whole program modular and easily expandable.

## 4.2. Force plate identification

There are many different types and brands of force plates used in the field of experimental biomechanics. The main brands are AMTI, Bertec and Kistler. While these plates measure ground reaction forces (GRF) directly, only the AMTI & Bertec plates measure the ground reaction moment about its origin (GRMo) directly.

In fact, these plates actually measure different components of force because they use different types of sensors in the hardware. For example, the AMTI & Bertec plates (<http://www.amti.biz>) incorporate strain gauges mounted on each corner of the plate to directly measure the main six components of force using six channels ( $F_x$ ,  $F_y$ ,  $F_z$ ,  $M_x$ ,  $M_y$ ,  $M_z$ ).

The Kistler plates (<http://www.kistler.com>) on the other hand consists of a base frame on which four piezoelectric crystal 3-component force sensors are mounted which use eight channels to output force ( $F_{x(1+2)}$ ,  $F_{x(3+4)}$ ,  $F_{y(4+1)}$ ,  $F_{y(2+3)}$ ,  $F_{z1}$ ,  $F_{z2}$ ,  $F_{z3}$ ,  $F_{z4}$ ). This clearly requires several equations to calculate our required six force components that the AMTI plates give us directly (Kistler claim that these sensors are used and placed in a way that measure pressures more accurately than AMTI but requires some computational effort to derive the final forces).

*Note: This toolbox assumes AMTI or Bertec (or any other 6 channel) type II plates are in use such that the required force parameters ( $F_x$ ,  $F_y$ ,  $F_z$ ,  $M_x$ ,  $M_y$ ,  $M_z$ ) come directly from the C3D file. The equations used to process ground data come from <http://www.kwon3d.com/theory/grf/cop.html>*

For other force plate types (i.e. 8 channel Kistler), if you have Vicon Nexus, you can open the C3D file in Nexus, and re-export it as C3D, and the force plate type will be converted to a six channel type II force plate, fully compatible with this toolbox.

**IMPORTANT NOTE: When Vicon Nexus exports a C3D file, it outputs ALL force platforms as Type 2 platforms (Fx, Fy, Fz, Mx, My, Mz) in their local force plate coordinate system and also outputs all markers in the Vicon coordinate system. So if you are using this toolbox to extract motion and kinetic data from C3D files exported by Vicon Nexus, ensure that the coordinate system is set up in loadlabels.m accordingly (see Section 4.5).**

All force plates are connected to analog channel inputs into the Vicon box for synchronization. The analog channels allow Vicon to determine which force plates are which, and this is achieved by assigning each analog channel a unique identifier. Figure 2 illustrates an example of a force plate configuration in a gait laboratory.

*Note: Force plate corners must also be identified in your data capture software to determine the force plate surface midpoints (used to calculate center of pressure, and to transform force plate signals into the laboratory frame).*

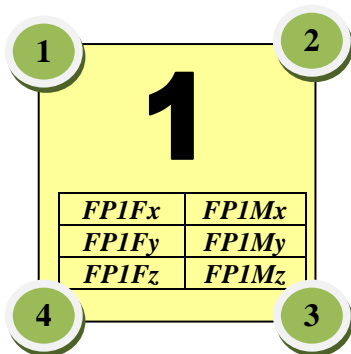
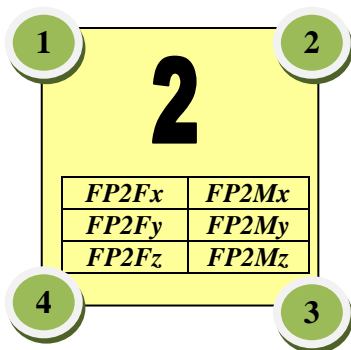
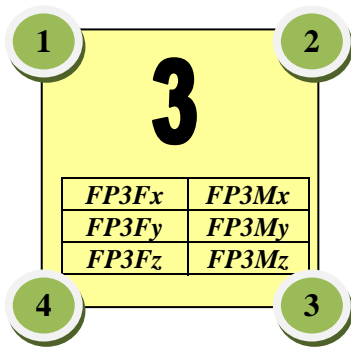


Figure 2: Force plate configuration in a gait laboratory. Values in the table denote the analog channel names. Values in the green circles denote the corner numbers.



### 4.3. Analog EMG Configuration

For any analog setup, it is important that the channels are labeled correctly. The analog setup can be accessed easily in Vicon. For example, in Vicon Workstation, go to the *System* Menu and select *Analog Setup*.

Firstly, check that the force plate channel labels are correct from Section **Error! Reference source not found.**. For each force plate *i*, the corresponding labels should be *FPiFx*, *FPiFy*, *FPiFz*, *FPiMx*, *FPiMy*, *FPiMz*.

If you are recording EMG signals, these must also be recorded as analog signals. The EMG channels follow the force plate channels as shown in Figure 3. It is very important that EMG channel labels match the labels of `glab.[emgSet]` in `loadLabels.m` (See Section 4.1) for data extraction to occur correctly. Refer to the Vicon manual for help on setting up analog channels.

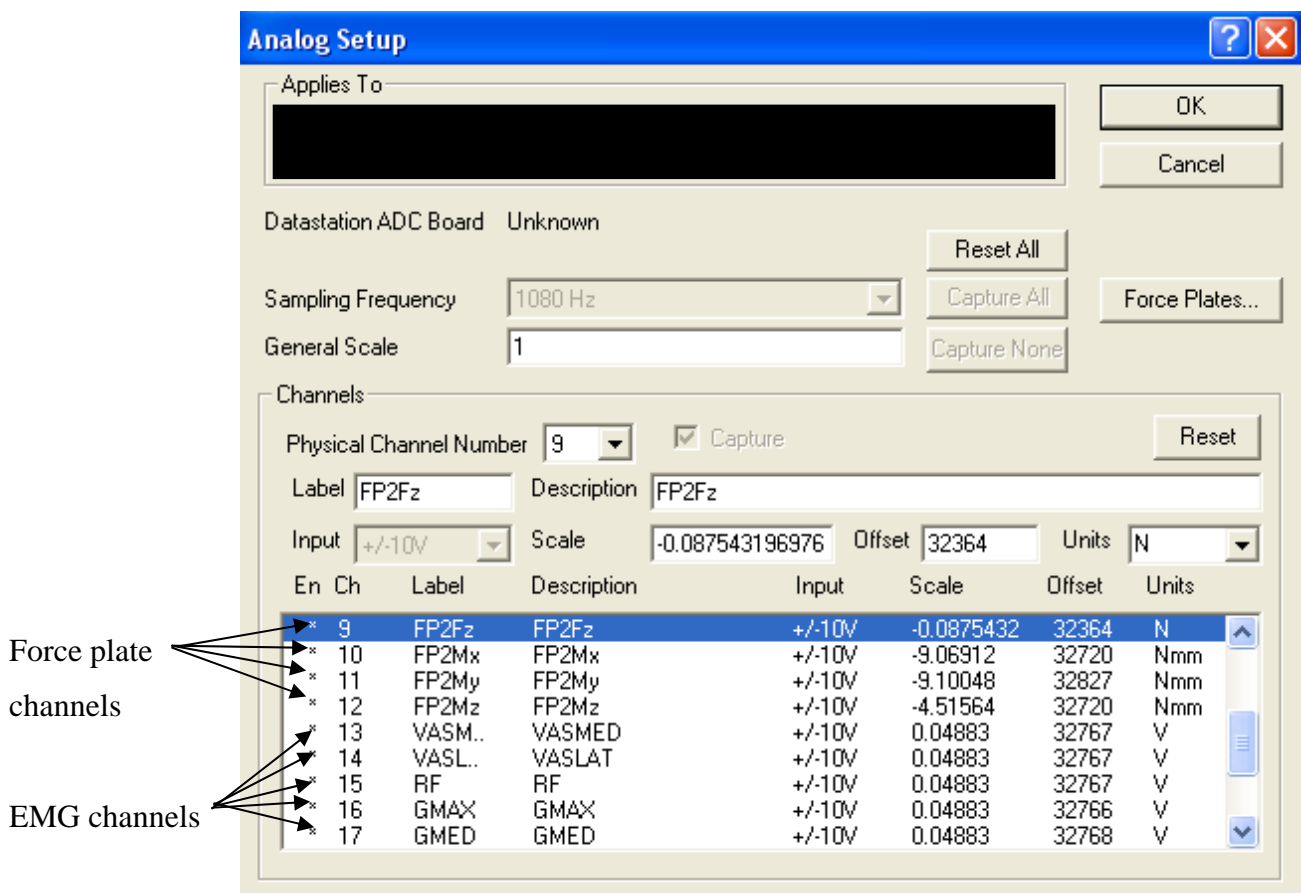


Figure 3: Analog EMG channel setup. This screenshot was taken from Vicon Workstation and may be slightly different to Vicon Nexus.

## 4.4. Coordinate Systems

There are three different coordinate systems which need to be taken into account when extracting kinetic data from the C3D file.

- Force plate (FP) coordinate system
- Laboratory (VICON) coordinate system
- Model (MODEL) coordinate system

Converting from one system to another can get a little confusing and tedious, so the function **coordChange.m** exists (see Section 5) to perform rigid body transformations from one system to another after kinetics or kinematics have been extracted (kinetics are extracted and output in the VICON coordinate system).

Note: the musculoskeletal model used with this toolbox is the Gait23xx\_Simbody in OpenSim:

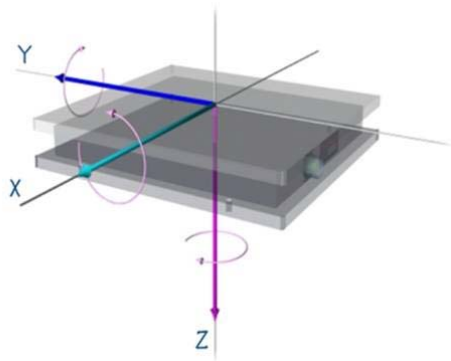
1. Anderson, F.C. and M.G. Pandy, *Dynamic optimization of human walking*. J Biomech Eng, 2001. **123**(5): p. 381-90.
2. Delp, S.L., Anderson, F.C., Arnold, A.S., Loan, P., Habib, A., John, C.T., Guendelman, E., Thelen, D.G., 2007. OpenSim: open-source software to create and analyze dynamic simulations of movement. IEEE Trans Biomed Eng 54(11), 1940-1950.

Figure 4 illustrates a coordinate system configuration for a typical gait laboratory and model.

Force plate – GREEN

Vicon – RED

Model – BLUE



AMTI Plate Coordinates

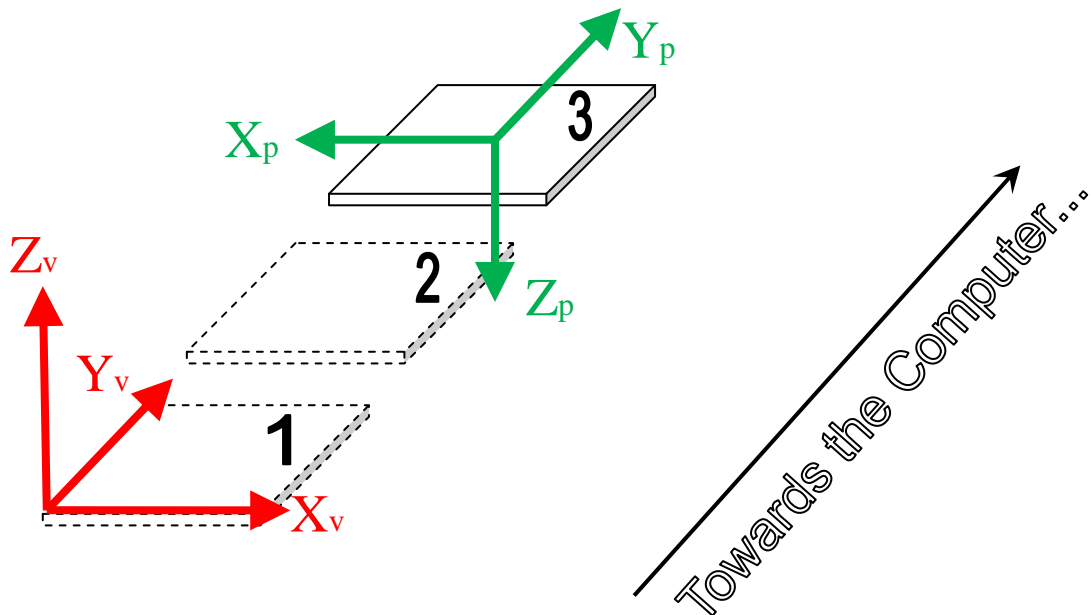
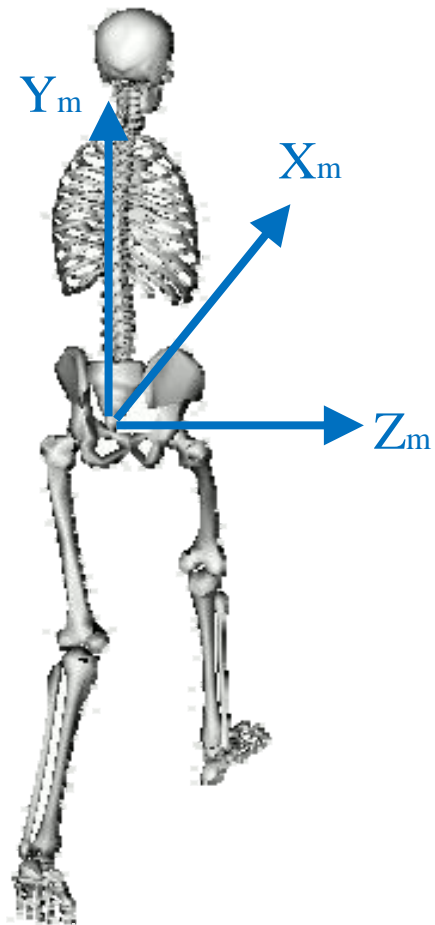


Figure 4: Three coordinate systems required for experimental gait data collection and modelling. 1) Vicon coordinate system (RED); 2) Force plate coordinate system (GREEN); Model coordinate system (BLUE).

## 4.5. Center of Pressure Calculation

The center of pressure (CoP) is the single point of application of the ground reaction force on a force plate and can be calculated by performing a moment equilibrium balance about the true origin of the force plate (where ground reactions are measured about). The center of pressure in the MODEL coordinate system is determined by summing three vectors (Figure 5).

### 1) MODEL ORIGIN TO FORCE PLATE SURFACE GEOMETRIC CENTER

This value comes from the geometric center of the force plate corners, as specified in the FORCE\_PLATFORM\CORNERS field of the C3D file. These values are originally expressed in the VICON coordinate system for each force plate and should be correctly setup in the Vicon software prior to data capture. The toolbox converts this into the MODEL coordinate system.

### 2) FORCE PLATE SURFACE GEOMETRIC CENTER TO TRUE ORIGIN

Due to slight defects in the manufacturing process, each force plate measures a force and moment about a different true origin  $\bar{O}(a,b,c)$ , which can be slightly offset from the geometric center of the plate surface  $C$  (and also lies below the surface of the plate). This vector from  $C$  to  $\bar{O}$  is determined by the manufacturer during individual calibrations and is specified in the FORCE\_PLATFORM\ORIGIN field in the C3D file. These values are originally expressed in the FORCE PLATE coordinate system for each force plate and should be correctly setup in the Vicon software prior to data capture. The toolbox converts this into the MODEL coordinate system.

*IMPORTANT NOTE: For this toolbox (and assumed in Vicon software), the ORIGIN vector is defined from  $C$  to  $\bar{O}$ . This means the  $Z_p$  value of this vector will be POSITIVE (because the force plate true origin will always be below the geometric center of the force plate surface). Some software may treat this vector in the reverse orientation, in which case, this toolbox will automatically detect and negate the vector during ground reaction processing.*

### 3) FORCE PLATE TRUE ORIGIN TO CENTER OF PRESSURE

This center of pressure value is calculated from a moment balance equation in the FORCE PLATE coordinate system. The toolbox will convert the CoP into the MODEL coordinate system.

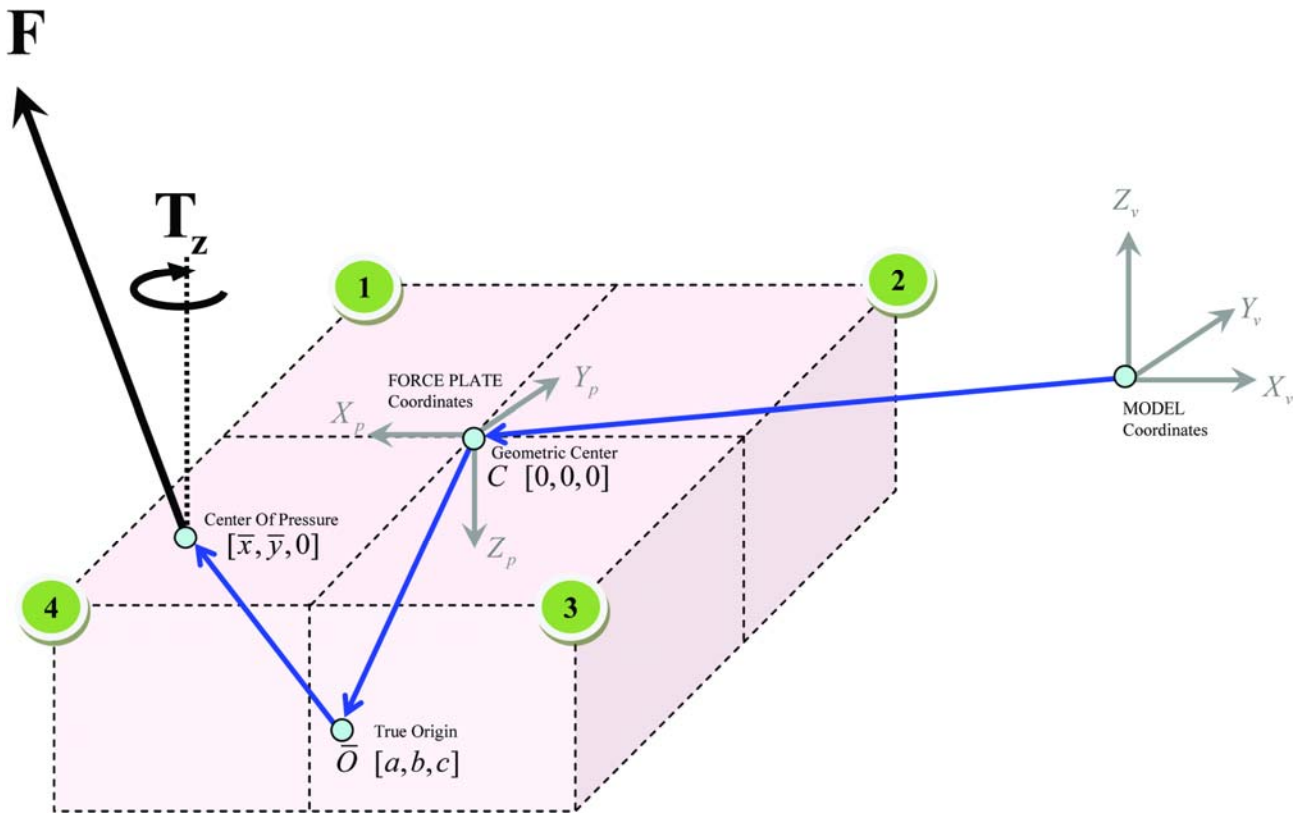


Figure 5: Calculation of the center of pressure. All coordinates shown are in the force plate coordinate system.

Taking moments about the force plate's true origin:

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} 0 & c & \bar{y} - b \\ -c & 0 & -(\bar{x} - a) \\ -(\bar{y} - b) & \bar{x} - a & 0 \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ T_z \end{bmatrix}$$

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} (\bar{y} - b)F_z + cF_y \\ -cF_x - (\bar{x} - a)F_z \\ (\bar{x} - a)F_y - (\bar{y} - b)F_x + T_z \end{bmatrix}$$

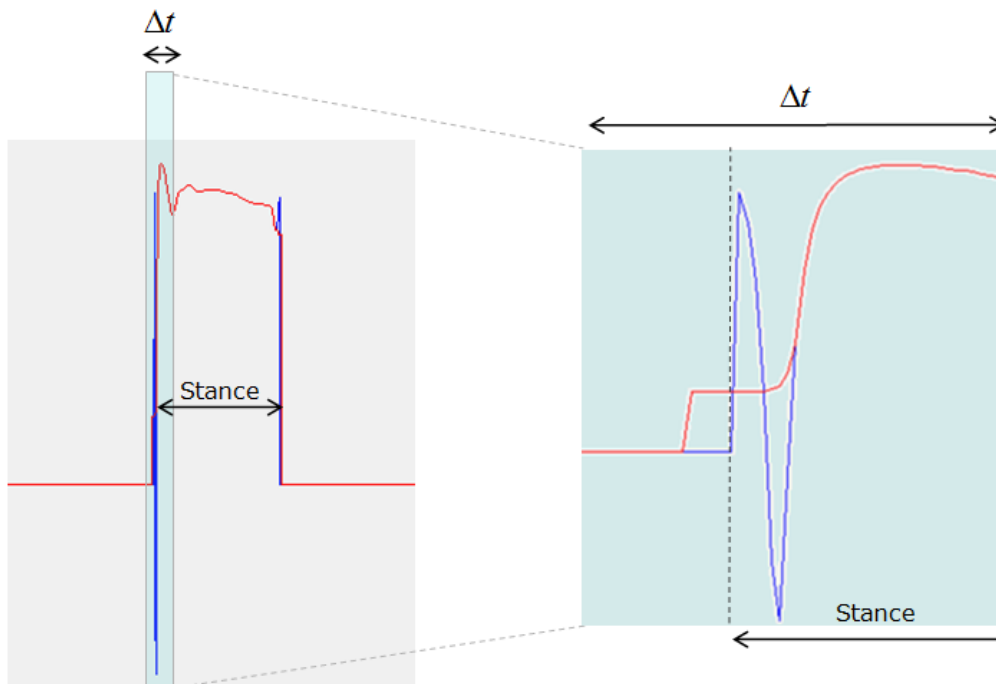
This leads to three equations and three unknowns:  $(\bar{x}, \bar{y}, T_z)$ .  $T_z$  represents a vertical free moment about the center of pressure. Solving for the three unknowns:

$$\bar{x} = -\frac{(M_y + cF_x)}{F_z} + a$$

$$\bar{y} = -\frac{(M_x + cF_y)}{F_z} + b$$

$$T_z = M_z - (\bar{x} - a)F_y + (\bar{y} - b)F_x$$

Noises in recorded ground reaction forces and moments will propagate to the CoP calculation. Furthermore, since the CoP is calculated by dividing by the vertical GRF, it is most sensitive at early and late stance where the vertical GRF is low, and discontinuities (or spikes) can occur (Figure 6). Applying a Butterworth filter to eliminate the discontinuities would not be ideal because doing so would modify the entire CoP trajectory. Furthermore, accurate CoP values may require to set weightings for the foot constraint points during a muscle induced acceleration. A CoP spike reduction algorithm was designed to filter out only the discontinuities in the first and last few frames of stance. The algorithm performs successive passes of the first and last few frames of CoP during stance, and looks for rapid discontinuities (successive frames of opposing CoP gradients). Where discontinuities occur, the “spike” magnitude is reduced by a factor of a half until the curve becomes smooth. Figure 6 illustrates the effect of the filter on the calculated anterior CoP during the stance phase of walking.



**Figure 6: Discontinuities (or spikes) can occur in early and late stance due to low vertical ground reaction forces. The blue line represents the standard CoP calculation. The red line represents the corrected CoP trajectory after CoP spike filter.**

## 4.6. Setting up alternate labs (Section 1 of loadLabels.m)

To set up the toolbox for data that was captured in a specific laboratory, you need to modify several parameters, all located in one file: **loadLabels.m**

**1a) Set up force plate numbers:** You need to set up the standard force plate channel names, and these names must be selectable via a numbering system. For example, you should not have arbitrary force plate channel names that don't have any meaning. For example, one possible naming system is as follows: All force plate channels begin with a prefix 'FP', followed by the force plate number, followed by 'F' or 'M', denoting a force or moment, followed by 'x', 'y', or 'z' denoting the direction. So the channel name 'FP3My' represents the Y moment recorded by force plate 3 (in FP coordinates). Numbering force plates is important when it comes to choosing which force plates the region of interest occurs. As an example, during running, a force plate may be skipped due to a large stride length.

```
glab.FP.string = '%s%d%s';      % Prefix(String), Plate(Int), Suffix(String)

glab.FP.prefix = {'FP', 'FP', 'FP', 'FP', 'FP', 'FP'};
glab.FP.suffix = {'Fx', 'Fy', 'Fz', 'Mx', 'My', 'Mz'};

glab.FP.verticalForceIndex = 3;
glab.FP.filterOrder = 4;
glab.vertForceCutoff = 10;
```

This says that force plate  $i$  is labeled as follows:

X force:	$F_{xi}$	X moment:	$M_{xi}$
Y force:	$F_{yi}$	Y moment:	$M_{yi}$
Z force:	$F_{zi}$	Z moment:	$M_{zi}$

Z is the vertical direction (in FP coordinates), corresponding to the 'FPiFz' channel.

If ground reaction forces and moments are filtered (in **getKinetics.m**), a zero phase Butterworth filter is used. The filter order is defined by `glab.FP.filterOrder` field.

A note on the way the toolbox (**getEvents.m**) determines which foot is on which force plate (foot-plate sequence): for each interval (an interval is defined by the space between two

successive events), the average vertical force is calculated for every force plate in the C3D file. If the average force is greater than a cutoff (*glab.vertForceCutoff*), the plate is deemed “active”. Active plate indices together with the event labels are used to determine the foot-plate sequence (so it is important to label events accurately – see Section 5).

*Note: if force plates are not “zeroed” prior to data collection, a constant nonzero vertical force offset (noise) may appear in C3D file. This may confuse the toolbox when trying to determine the set of “active” force plates for a given interval. In cases where vertical noise exists, it may be necessary to increase the vertical force cutoff (glab.vertForceCutoff). It is set to 10N by default (if this parameter is not explicitly specified in the loadLabels.m file.*

**1b) Set up coordinate systems:** The toolbox requires rigid body transformations between the three main coordinate systems (VICON, FP, and MODEL) to extract all data into the MODEL coordinate system (for OpenSim). Because the relationship between VICON and FP coordinate systems are defined inside the C3D file by the force plate corner orientations, the user only needs to supply the relationship between the VICON and MODEL coordinate systems. For motion capture systems, the Z direction in the VICON laboratory frame is always vertical. Therefore, depending on how the laboratory is set up, the forward direction of gait may occur along one of four possible VICON directions: +X, -X, +Y, -Y. For each of these cases, the user needs to describe the transform between the VICON and MODEL coordinate systems.

```
% X direction (Vicon) Gait -> transformation vectors
glab.transform.VICMODEL(1,:) = [1 3 -2];

% -X direction (Vicon) Gait -> transformation vectors
glab. transform.VICMODEL(2,:) = [-1 3 2];

% Y direction (Vicon) Gait -> transformation vectors
glab. transform.VICMODEL(3,:) = [2 3 1];

% -Y direction (Vicon) Gait -> transformation vectors
glab. transform.VICMODEL(4,:) = [-2 3 -1];
```

For example, if a subject is walking forward in along the Y VICON direction, then the following transformation holds:

$$\text{Model (X)} = \text{Vicon (Y)}$$

$$\text{Model (Y)} = \text{Vicon (Z)}$$

$$\text{Model (Z)} = \text{Vicon (X)}$$



Conversely, if the subject has another trial whereby they turn around and again walk forward, now in the –Y VICON direction, the following transformation will now hold:

$$\text{Model (X)} = \text{Vicon (-Y)}$$

$$\text{Model (Y)} = \text{Vicon (Z)}$$

$$\text{Model (Z)} = \text{Vicon (-X)}$$

This transformation can easily be seen on the diagram on page 7. The user is encouraged to draw a similar diagram for their laboratory to assist in determining the transformations.

*IMPORTANT NOTE: Each time you run **getEvents.m**, local copies of the transform matrices are stored in the C3Dkey Matlab structure these local copies are used for extracting all subsequent data (i.e. ground forces, center of pressure, etc). So if you change any transforms in the **loadlabels.m** file, ensure that you rerun **getEvents.m** for the trial to update the local transform copy in the C3Dkey so that these changes become applicable.*

**1c) Set up special markers:** Two sets of special markers are currently defined.

```
glab.offsetMarker = 'SACRUM_MARKER';
```

A single marker label can be defined to rigidly translate (align) the extracted markers to the OpenSim “ground” platform so that the model is standing at the start of the platform when the trial begins. This transformation is performed in the model X and Z direction only. This is handy because laboratory (VICON) origins may differ from lab to lab and not necessarily correspond to the global origin in the OpenSim environment. Note: A marker placed on the posterior pelvis or trunk region is usually a good choice for an offset marker.

```
glab.legLengthMarkers = {'RASIS', 'RMALLEOLUS', 'LASIS', 'LMALLEOLUS'};
```

(Optional): A set of four markers that define the proximal/distal locations for each leg. This is used to calculate and output the subject’s leg length (**getEvents.m**) to a file when a static trial is being extracted.

**1d) Set up joint model names:** This is simply a structure containing the joint names in the OpenSim model. Currently the joint label is only used to get create the coordinates file in **getKinetics.m**. This is only needed if using the toolbox for OpenSim simulations.

**1e) Set up the directory to store toolbox output:** Output images, mat files, and text files from toolbox function calls can be stored in this directory for future reference. `glab.storeInfo` must have a backslash at the end. *i.e.* `!\MyDir\`. This functionality is toggled by `glab.storeInfo` (1 = on, 0 = off)

```
glab.storeInfo = 1;  
glab.infoDirectory = '!\GaitExtract\';
```

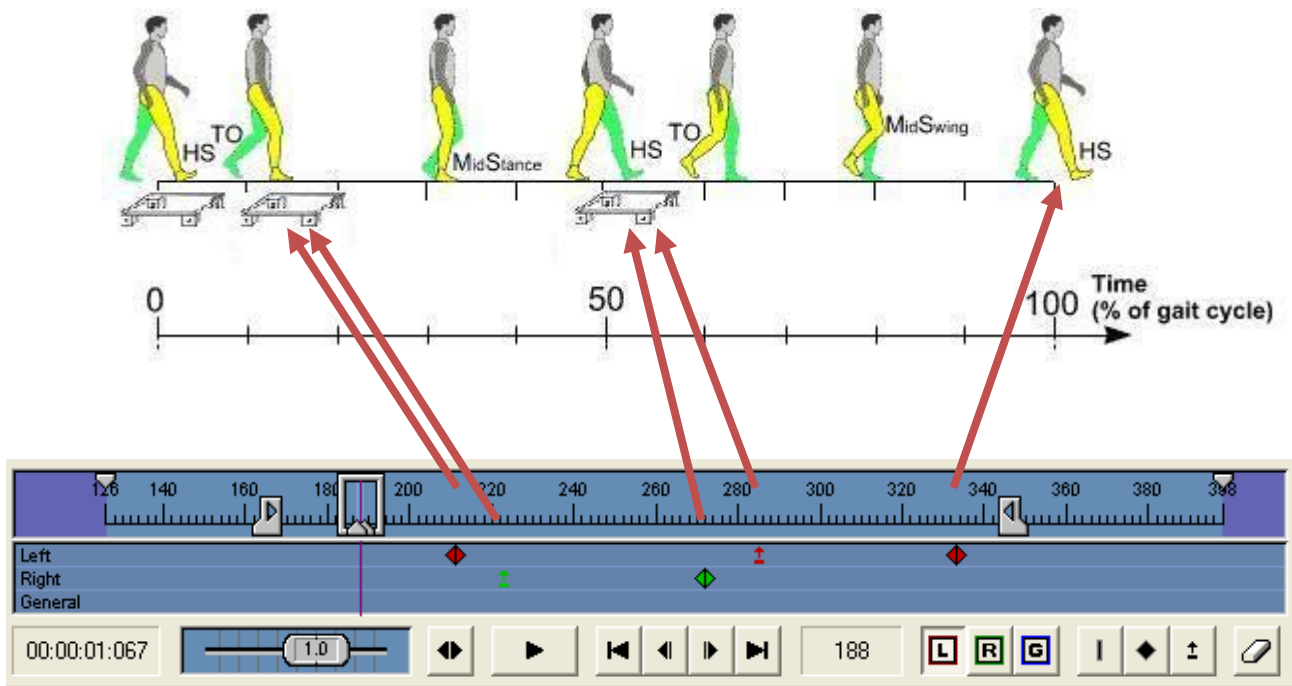
**Set up your marker sets and EMG labels:** `loadLabels.m` also contains all local settings which are dependant on the model being used. See section 4 for more details.

If you are analyzing trial files from multiple labs, place the `loadLabels.m` file for each lab in its own directory. That way, the `loadLabels.m` file in the current working directory will be higher in the path list and will get executed. Type `'which loadLabels'` in Matlab to verify this.

## 5. Event Labeling

In Section 4.5 (part 1a), you may recall that the foot-plate sequence is determined by examining each force plate's vertical force in each event interval (space between two successive events). This foot sequence represents which foot is on which force plate (termed the "active" force plate) at any given time and will eventually become the ground reactions that are applied to the foot segments in OpenSim. In order to detect these foot-plate sequences, events corresponding to foot strike (FS) and foot off (FO) for each leg should be defined accordingly in the C3D file. However, the foot-plate sequence detection algorithms only work for forward human gait. For other forms of movement (such as landing or backward walking), it is possible to override the foot-plate detection algorithm and explicitly specify the right and left foot "active" force plates for each event interval. See the description of `getEvents.m` in Section 6 for information on how this is done.

Events are also used for cropping the trial. Trials are always cropped between the first and last event. An illustration of event labelling is shown below.



For consistent labeling in Vicon, the following guidelines may be useful:

Foot Strike (FS) – defined as the frame just before the GRF vector appears on the foot  
 (also known as heel strike (HS) in walking gait)

Foot Off (FO) – defined as the frame just after the GRF vector disappears from the foot  
 (also known as toe off (TO) in walking gait)

*Important Note: If you have cropped a trial in Vicon (i.e. the above trial has been cropped from frame 126 to 398), events that existed outside this time frame are **not** removed from the C3D file and may cause problems when using this toolbox. It is important to ensure that all events outside the cropped regions are removed. Alternatively, you can crop the trial, remove all events, and begin labeling from scratch.*

## 5.1. Foot Detection Algorithm

To apply an external GRF recorded by the force plate onto a foot in the model, it must be known which foot-side (left or right) the GRF belongs to. Thus it is important to group the foot strikes into left and right components. Event tags label the stages of locomotion (foot-strike and foot-off on each leg). This information was used to design an algorithm to automatically detect the foot-side striking each plate. The algorithm was driven by two assumptions: 1) the stance foot force was associated with only one plate (i.e. each foot hits exactly one plate during stance); and 2) locomotion occurs in the direction of increasing force plate identification (i.e. plate 1 strike → plate

2 strike → plate 3 strike). In short, the algorithm cycled through each set of two consecutive event tags (denoted as a sequence) and determined the active force plates (plates that were recording a force profile). Combining the active plates with the event information allowed the algorithm to detect the foot-sides during the sequence. The pseudo code below (Figure 7) outlines the foot detection algorithm in detail.

```
for each sequence in the trial {
    // Determine "active" plates in the sequence by examining the average
    // vertical force across each plate in the laboratory. If the average
    // force was greater than a fixed cutoff force (i.e. 10N), the plate
    // was deemed "active". Sort these plates in ascending order following
    // assumption #2
    for i:numForcePlatesInTrial {
        averageForce(i) = getAverageForceInSequence;
    }
    activeForcePlates = sort(getPlates(averageForce > 10));
    numberActivePlates = length(activeForcePlates);

    if numberActivePlates == 0 {
        // No feet on the ground (double float)
        RightFootPlate = 0;
        LeftFootPlate = 0;
    }

    elseif numberActivePlates == 1 {
        // One foot on the ground (single support)
        // The foot-side is determined by examining the first event
        // label of the sequence. If it is a FOOTSTRIKE, then the
        // foot-side is equal to the side of the FOOTSTRIKE. If it is
        // a FOOTOFF, then the foot-side is the equal to the opposite foot-
        // side of the FOOTOFF.
        if firstEvent == FOOTSTRIKE {
            if footStrikeSide == RIGHT {
                RightFootPlate = activeForcePlates(1);
                LeftFootPlate = 0;
            }
            elseif footStrikeSide == LEFT {
                RightFootPlate = 0;
                LeftFootPlate = activeForcePlates(1);
            }
        }
        elseif firstEvent == FOOTOFF {
```

```

        if footOffSide == RIGHT {
            RightFootPlate = opp(activeForcePlates(1));
            LeftFootPlate = 0;
        }
        elseif footOffSide == LEFT {
            RightFootPlate = 0;
            LeftFootPlate = opp(activeForcePlates(1));
        }
    }

elseif numberActivePlates == 2 {
    // Two feet on the ground (double support)
    // There are now two foot-sides and two active plates. The
    // algorithm needs to determine which foot is on which plate. Note
    // that double support phase will only be present in walking, not
    // running. In walking, double support occurs when an ipsilateral foot
    // makes contact with the ground during mid-stance of the
    // contralateral foot. The contralateral foot will be characterized by
    // a FOOTOFF event and will be the foot behind the ipsilateral foot.
    contralateralFoot = getFoot(find(FOOTOFF_event))
    if contralateralFoot == RIGHT {
        RightFootPlate = activeForcePlates(1);
        LeftFootPlate = activeForcePlates(2);
    }
    elseif contralateralFoot == LEFT {
        RightFootPlate = activeForcePlates(2);
        LeftFootPlate = activeForcePlates(1);
    }
}
}

```

**Figure 7: Algorithm used to automatically detect foot-sides that correspond to each plate. This algorithm was implemented inside the GaitExtractToolbox and used to batch prepare each subject for OpenSim.**

## 6. Detailed Function List

The following pages will explain in some detail the functions contained within the toolbox. This section will be continually updated as more functions are created. If anyone has made and tested their own function, it would be appreciated to pass on to extend the toolbox.

Please report any suggestions / bugs / extensions to Tim Dorn: [t.dorn@pgrad.unimelb.edu.au](mailto:t.dorn@pgrad.unimelb.edu.au)

A few notes before the function list:

\* next to a variable input denotes optional inputs

GRF = Ground Reaction Force

CoP = Center of Pressure

GRMo = Ground Reaction Moment about Origin

GRMx = Ground Reaction Moment about CoP

[eVecGlob, EMGVecGob] =

**batchEMGprocess(C3Dkey, emgSetName, emgProcessTasks, fileSuffix\*)**

---

**Description:** Batch process EMG signals.

**Version:** November 2008

**Inputs:** C3Dkey = key of dynamic C3D file (from [getEvents.m](#))  
emgSetName: the label of EMG names contained in the EMG set  
(this must be defined in loadlabels.m as a cell: glab.[emgSetName])  
emgProcessTasks: the EMG processing options in the order of execution  
(this must be defined in loadlabels.m as a cell: glab.[emgProcessTasks])  
fileSuffix\* = suffix of mot file that is saved if this is not included, or empty, then file is not saved

**Outputs:** eVecGlob = structure of processed EMG  
EMGVecGlob = structure of raw EMG

**Notes:** N/A

## **createEvent(c3dFile, foot, label, frame)**

---

**Description:** Create an event in a C3D file and save the C3D file.

**Version:** June 2009

**Inputs:**

- c3dFile: the name of the C3D file
- foot: 'R' for right foot event, 'L' for left foot event, 'G' for general foot event
- label: 'FS' for footstrike, 'FO' for footoff, 'GEN' for general event
- frame: video frame number to add the event at

**Outputs:** OVERWRITES the input C3D file with new events (irreversible with this code so make sure to backup the original C3D file first!)

**Notes:** N/A



[out, plots2make, labels, dataFile] =

**extractMotFile(ProcessTask1, Value1, ProcessTask2, Value2, ...)**

---

**Description:** Extract (and plot) data from a saved OpenSim data file (\*.mot or \*.sto) into a Matlab structure

**Version:** July 2009

**Inputs:** See inside the m file for all the options

**Outputs:**

- out.name: trial name
- out.labels: extracted data labels
- out.data: extracted data (after filtering)
- out.filtFreq: low pass filter frequency
- performPlots: indices of plots extracted
- labels: all labels from dataFile
- dataFile: filename used

**Notes:** If no input arguments are given i.e. data = extractMotFile, the function allows you to select a file for plotting and will superimpose over existing plots if a common variable is being plotted.

## **generateMotFile(dataMatrix, colnames, filename)**

---

**Description:** Generate a motion \*.mot file readable by OpenSim

**Version:** Nov 2008

**Inputs:** dataMatrix = data matrix to write to file (first column should be time)  
colnames = cell array of column name strings  
filename = string containing the output filename (must include extension)

**Outputs:** output motion file (\*.mot)

**Notes:** Number of data columns must match the number of column names or an exception will be thrown.

## **generateTrcFile(C3Dkey, markerpos, markerset)**

---

**Description:** Generate a marker \*.trc file readable by OpenSim

**Version:** June 2009

**Inputs:** C3Dkey: the C3D key structure from getEvents  
markerpos = array of marker positions  
          for M markers: should contain 1+3M columns  
          (time + XYZ of each marker)  
markerset = cell array of strings containing the names of markers  
          e.g. markerset = {'M1', 'M2', 'M3'};

**Outputs:** output marker file (\*.trc)

**Notes:** Number of data columns must match the number of column names or an exception will be thrown.



offsetTime\* = offset time in seconds added to frame event (default = 0). Useful for external device trigger delays to Vicon.

**Outputs:** C3Dkey is a structure which contains key information about the trial:

C3Dkey.transform.FPMODEL: transform matrix from FP → MODEL

C3Dkey.transform.VICMODEL: transform matrix from VICON → MODEL

C3Dkey.transform.FPVICON: transform matrix from FP → VICON

C3Dkey.transform.MODELFP: transform matrix from MODEL → FP

C3Dkey.transform.MODELVIC: transform matrix from MODEL → VICON

C3Dkey.transform.VICONFP: transform matrix from VICON → FP

C3Dkey.name: subject name

C3Dkey.markerSet: marker set used for the trial

C3Dkey.c3dFile: C3D file name

C3Dkey.direction: direction of forward facing value

C3Dkey.aFreq: analog frequency

C3Dkey.vFreq: video (VICON) frequency

C3Dkey.r: video/analog frequency ratio

C3Dkey.mass: subject mass (SIMPLE TRIAL ONLY)

C3Dkey.numFrames.uncroppedV: number of total video frames in the trial

C3Dkey.numFrames.uncroppedA: number of total analog frames in the trial

C3Dkey.numFrames.croppedV = number of cropped frames (video)

C3Dkey.numFrames.croppedA = number of cropped frames (analog)

C3Dkey.event.txt: label of events

C3Dkey.event.times: times of events

C3Dkey.event.percent: percentage of cycle that events occur

C3Dkey.event.Vframe: video frames of events

C3Dkey.event.Aframe: analog frames of events

C3Dkey.event.Vframe0: video frames of events (starting at frame 1)

C3Dkey.event.Aframe0: analog frames of events (starting at frame 1)

C3Dkey.event.times0: times of events (starting at time 0)

C3Dkey.interval.txt: txt interval of events  
C3Dkey.interval.time: time interval of events  
C3Dkey.interval.Vframe: video frame interval of events  
C3Dkey.interval.Aframe: analog frame interval of events  
C3Dkey.interval.time0: time interval of events (starting at time 0)  
C3Dkey.interval.Vframe0: video frame interval of events (starting at frame 1)  
C3Dkey.interval.Aframe0: analog frame interval of events (starting at frame 1)

C3Dkey.sequence.frames: force plate frame sequence  
C3Dkey.sequence.plates: force plate number sequence  
C3Dkey.sequence.txt: force plate event text sequence

C3Dkey.offset: offset in mm to put the model on the platform in OpenSim  
C3Dkey.averageSpeed: average trial speed (m/s)

C3Dkey.FP\_order: Force plate order (in terms of stepping #)  
C3Dkey.FP\_order\_inv: Force plate order inverse  
C3Dkey.trialType: trial type (either will be SIMPLE or GENERAL)  
C3Dkey.numPlatesTotal = total number of force plates in the trial  
C3Dkey.numPlatesUsed = number of force plates used for extraction  
C3Dkey.stanceFrames = (1,:) - Right leg, (2,:) - Left leg

C3Dkey.allowed.markers: 1 if markers can be extracted from this c3dfile  
C3Dkey.allowed.kinetics: 1 if markers can be extracted from this c3dfile  
C3Dkey.allowed.EMG: 1 if EMG can be extracted from this c3dfile

Time Vectors (of labeled event):

-----  
C3Dkey.timeVec.c3dAnalogFrame: actual analog frame number (from C3D)  
C3Dkey.timeVec.analogFrame: analog frame number (starting at 1)  
C3Dkey.timeVec.c3dVideoFrame: actual video frame number (from C3D)  
C3Dkey.timeVec.videoFrame: video frame number (starting at 1)  
C3Dkey.timeVec.Asec: analog time (sec) -> starting at 0 sec  
C3Dkey.timeVec.Vsec: video time (sec) -> starting at 0 sec

C3Dkey.timeVec.Apercent: analog percentage of labeled event  
C3Dkey.timeVec.Vpercent: video percentage of labeled event

**Notes:** Events must be examined and labeled in Vicon before using this script. For static trials, events need to be labeled at the start and end of where you want the static pose data to be cropped. Any type of event (FS or FO) is fine for this. For dynamic trials, there must be at least two events (start and end), but it can also have any number of intermediate events in between (i.e. left foot off, right foot strike, etc) to label the major phases of the gait cycle.

You must also ensure that the lab setup is correct in *loadLabels.m*. See section 4.5 for more information.

[GRF, CoP, GRMo, GRMx] =

**getKinetics(C3Dkey, plottog\*, filterFreq\*, markersDyn\*)**

---

**Description:** Extract and process kinetic data (GRF, CoP, GRMo, GRMx) from a C3D file.

**Version:** Sept 2010

**Inputs:** C3Dkey: the C3D key frames structure from getEvents

plottog\* = toggles the display of various plots

0 = no plots (default)

1 = plot kinetic graphs only (all in MODEL coordinates)

2 = plot kinetic graphs & kinetic verification (all in MODEL coordinates)

4 = plot kinetic verification only (all in MODEL coordinates)

filterFreq\* = filter frequency for GRF, GRMo (optional)

< 0 means no filtering is done.

Uses a 4<sup>th</sup> order low pass Butterworth filter.

markersDyn\* = optional dynamic markers data structure (from getMarkers.m) to aid with the verification of extracted kinetics (used in verifyKinetics.m)

**Outputs:** The output is set up as follows:

GRF(1,:) Right Foot --> GRF X

GRF(2,:) Right Foot --> GRF Y

GRF(3,:) Right Foot --> GRF Z

GRF(4,:) Left Foot --> GRF X

GRF(5,:) Left Foot --> GRF Y

GRF(6,:) Left Foot --> GRF Z

CoP(1,:) Right Foot --> CoP X

CoP(2,:) Right Foot --> CoP Y

CoP(3,:) Right Foot --> CoP Z

CoP(4,:) Left Foot --> CoP X

CoP(5,:) Left Foot --> CoP Y

CoP(6,:) Left Foot --> CoP Z



GRMo(1,:) RF -> GRM X about FP origin  
GRMo(2,:) RF -> GRM Y about FP origin  
GRMo(3,:) RF -> GRM Z about FP origin  
GRMo(4,:) LF -> GRM X about FP origin  
GRMo(5,:) LF -> GRM Y about FP origin  
GRMo(6,:) LF -> GRM Z about FP origin

GRMx(1,:) RF -> GRM X about CoP  
GRMx(2,:) RF -> GRM Y about CoP  
GRMx(3,:) RF -> GRM Z about CoP  
GRMx(4,:) LF -> GRM X about CoP  
GRMx(5,:) LF -> GRM Y about CoP  
GRMx(6,:) LF -> GRM Z about CoP

**Notes:**

- 1) Events MUST be labeled in VICON (and hence the C3D file).
- 2) Corners must be defined in VICON properly. Looking down onto the plate from above: corner 1: where X, Y are both most positive in FP coordinates  
Corners 2, 3, 4 going CLOCKWISE (refer to Vicon manual for more information)
- 3) Force plate origins must be defined properly from force plate true origin to the center of the plate surface... in FP coordinate system (given in FP manual)
- 4) The key file (used by **getEvents.m** is slightly appended to by adding the force plate order, and the result is saved to key.mat
- 5) Output is saved as a \*.mot file used by OpenSim

**Additional Options inside the m file:**

```
lw = 3; % Set plot line width  
opt = 'b'; % Plotting options  
titlesize = 13; % Title font size
```

**markers =**

**getMarkers(C3Dkey, markerSetName, filter\*)**

---

**Description:** Extract marker position data from a C3D file.

**Version:** October 2011

**Inputs:**

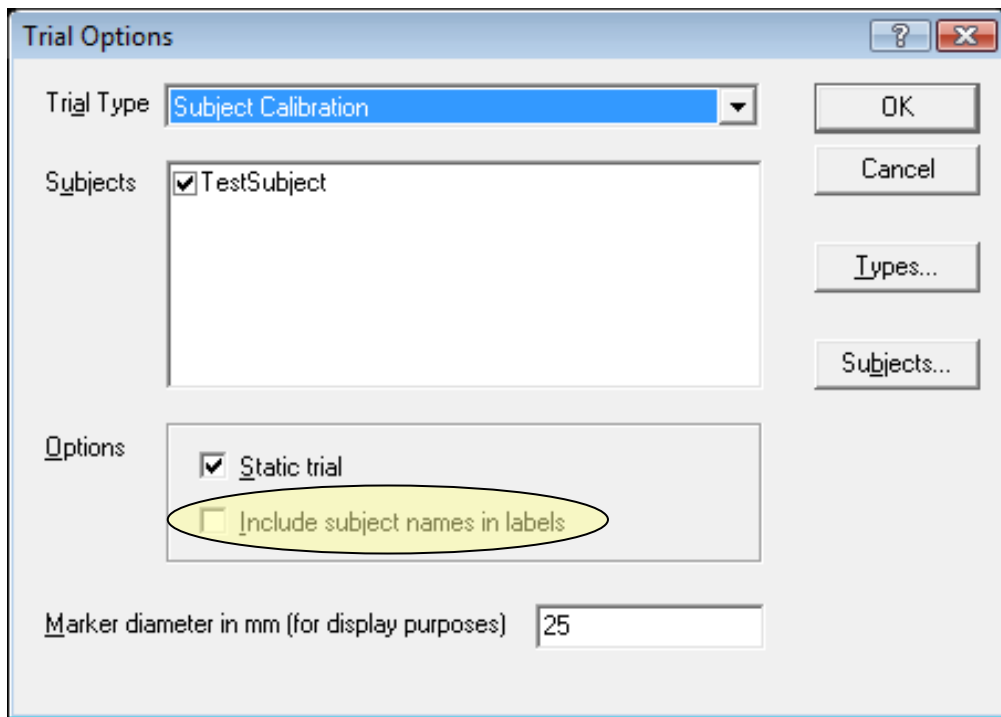
- C3Dkey: the C3D key frames structure from getEvents
- markerSetName: the label of markers contained in the marker set  
(this must be defined in loadlabels.m as glab.[markerSetName])
- filter: optional low pass filter the marker positions
  - if filter = 0, no filtering is done (default)
  - if filter > 0, filtering is done at (filter) Hz

**Outputs:**

- markers.data: the marker position data
- markers.label: the marker position labels
- markers.units: units of marker positions
- markers.divide\_to\_meters: scale to divide to convert to meters
- markers.SUCCESS: marker names that have been extracted successfully
- markers.FAILED: marker names that have failed (do not exist)
- markers.MISSINGMARKERS: marker names that have markers missing

**Notes:** The output is saved as a \*.trc file used by OpenSim

It is important that the label names stored in the 'glab.markers' variable in loadLabels.m match the label names used in Vicon for extraction to be successful. Sometimes Vicon stores the marker labels as [markerName:SubjectName]. This will result in an error since the strings do not match. To resolve this, ensure that “Include subject names in labels” check box is switched OFF in Vicon → Trials → Options. The data is extracted in the order defined in glab.markers.



[MVC, MVCset] =

getMVC(C3Dkey, emgSetName, windowSize, \*MVCmethod)

---

**Description:** Obtain the Maximum Voluntary Contraction (MVC) voltage from a set EMG.

**Version:** June 2008

**Inputs:** C3Dkey: the C3D key frames structure from getEvents

emgSetName: the label of EMG names contained in the EMG set (this must be defined in loadlabels.m as glab.[emgSetName])

windowSize = gliding 'look ahead' window size (sec)

\*MVCmethod = calculating method of MVC (used once the maximum mean EMG time window has been found)

'MEAN' → MVC = mean value (*default value if not specified*)

'RMS' → MVC = root mean square value

'MAX' → MVC = maximum value

**Outputs:** MVC = structure of MVC voltages (uV)

(MVC.[muscleLabel] = value)

MVCset = structure of cells containing information about the EMG labels

**Notes:** ---- EMG LABELS ARE CONTAINED IN: **loadLabels.m**

**References:**

ABC or EMG (page 30)

Bolgia, L. A. and T. L. Uhl (2007). "Reliability of electromyographic normalization methods for evaluating the hip musculature." J Electromyogr Kinesiol 17(1): 102-11.

## loadLabels()

---

**Description:** Loads label files used throughout the toolbox.

**Version:** User updated

**Inputs:** N/A

**Outputs:** N/A

**Notes:** See Sections 4.1 & 4.5 for label descriptions

[eVecGlob, EMGVecGlob] =

**multipleEMGprocess(C3DFile, emgSetName, emgProcessTasks, interval4Time)**

---

**Description:** Batch Process multiple stride EMG signals from a single C3D file

**Version:** July 2009

**Inputs:** c3dFile = the name of the C3D file

emgSetName: the label of EMG names contained in the EMG set  
(this must be defined in loadlabels.m as a cell: glab.[emgSetName])

emgProcessTasks: the EMG processing options in the order of execution  
(this must be defined in loadlabels.m as a cell: glab.[emgProcessTasks])

interval4Time = the interval number to set for the time vector

**Outputs:** eVecGlob = structure of processed EMG  
EMGVecGlob = structure of raw EMG

**Notes:** Ensure that events are places in the C3D file at the start & end of each interval. e.g. 4 events == 3 intervals. If we want to set the time vector to be the middle interval (between events 2&3), then interval4Time = 2.

e.g. multipleEMGprocess('myfile.C3D', 'emgset', 2)

The output file will contain the time column (from interval4Time) and the time normalized AVERAGE emg data over all intervals

eVec =

**processEMG**(EMGVec, {ProcessTask1, Value1, ProcessTask2, Value2, ...})

---

**Description:** Process raw EMG values

**Version:** June 2009

**Inputs:** EMGVec.data = Raw EMG data from getEMG.m

EMGVec.time = time vector (sec)

EMGVec.name = string of muscle label

ProcessTaskX = Processing task X

ValueX = Value for processing task X

C3Dkey\* = if this is given, the event lines will be added the processed EMG plot

**Outputs:** eVec = processed EMG structure (containing .time & .data)

**Notes:** Processing tasks are performed in the order they are given:

Task = 'REMDC' = Remove DC offsets Value = []

Task = 'RECT' = Full wave rectification Value = []

Task = 'REMDCRECT' = Remove DC offset & full wave rectification Value = []

Task = 'HPF' = High pass filter Value = [FilterOrder, Freq(Hz)]

Task = 'LPF' = Low pass filter Value = [FilterOrder, Freq(Hz)]

Task = 'BPF' = Band pass filter Value = [FilterOrder, FreqLow(Hz), FreqHigh(Hz)]

Task = 'TKE' = TKE filter Value = []

Task = 'NORM' = MVC Normalization Value = MVC (uV)

Task = 'NORM1' = Normalization to 1 Value = []

Task = 'ABOVEZERO' = Force EMG > 0	Value = []
Task = 'SQRT' = Squareroot EMG	Value = []
Task = 'MULTIPLY' = Multiply EMG	Value = MultipleNumber
Task = 'REMOVESPIKES' = Remove Spikes	Value = []
Task = 'PLOT' = Plot2Screen value	Value = 1(ON default), 0(OFF)
Task = 'SAVE' = Save plots to emf & fig	Value = [indices of process ops to not plot]
Task = 'HIDE' = Hide line indices	Value = [](OFF default), (ON)
Task = 'VERTLINES' = Plot event lines	Value = C3Dkey (default = []) only active when PLOT = 1

Note that the HPF and LPF options, a zero-phase Butterworth filter is used



[xmlString, fileName] =

writeXML(type, C3Dkey\*)

---

**Description:** Write XML files for use in OpenSim.

**Version:** Sept 2010

**Inputs:** type: the type of XML setup file to be created (case sensitive)

- type = 'scale' --> create scale XML file
- type = 'ik' --> create inverse kinematics XML setup file
- type = 'id' --> create inverse dynamics XML setup file
- type = 'so' --> create static optimization XML setup file
- type = 'jr' --> create joint reaction XML setup file
- type = 'rra' --> create residual reduction XML setup file
- type = 'cmc' --> create computed muscle control XML setup file
- type = 'pi' --> create pseudo inverse GRF decomposition XML setup file

C3Dkey\*: the C3D key frames structure from getEvents  
(if not given, default parameters are used. These can be then modified manually using an XML editor)

**Outputs:** xmlString: the generated XML string  
fileName: the generated file name

the XML file is saved in the current working directory as

*[C3Dkey.c3dFile]\_Setup\_[type].xml*

**Notes:** Note that these XML files are only templates. They may need to be fine tuned in terms of the paths of the models you are using, and any additional settings that the analyses offer. The XML files can be modified in Notepad++ or for the more advanced Matlab users, you can go into the **writeXML.m** file and modify some of the default XML output settings.

## 7. Example Subject Trial

The EXAMPLE directory contains two examples for your viewing. The first is a static and dynamic walking C3D file (captured using Vicon Workstation). The second is a static and dynamic downstairs walking C3D file (where the force plates are individually orientated in different positions). Inside each example directory, there is a high level m file that contains the pre-processing instructions required to output relevant kinematics, ground reactions, and EMG to a format suitable for OpenSim. The m files should be self explanatory if you have read Sections 1 to 5. Please note that conventional units are always used unless explicitly stated (i.e. kg for mass, meters for length).

To run these examples, ensure that the toolbox is correctly installed and paths set correctly. In Matlab (V2007a or greater), go into the EXAMPLE folder, choose your example, and run *testWalk.m* or *testDownStair.m*. Note the Matlab outputs as well as the additional directories and files created from the execution of the script.

Once the examples have been run successfully, you can preview the motion and ground forces together in OpenSim, by selecting “Preview Motion Data” under the “File” menu. Then select the marker *exampleX\_\*.trc* file. Do the same for the kinetics file *exampleX\_\*\_kinetics.mot*. Synchronize the motions together and view the extracted data. Everything should be ready for further analysis in OpenSim!

This example is provided to outline the functionality of the toolbox. Trials are given for demonstration purposes only. Please refer to the OpenSim web site for further information on the specifics of muscle actuated simulations (<https://simtk.org/home/opensim>).

## 8. Revision Information

- |      |               |   |
|------|---------------|---|
| V1.0 | February 2007 | - <b>Initial release</b>  |
| V1.1 | March 2007    | <ul style="list-style-type: none"><li>- Fixed some installation path bugs</li><li>- <b>runinvKin.m</b> now also outputs marker positions (raw or filtered) in standard format</li><li>- <b>runinvKin.m</b> now also provides a user friendly way to copy the static kinematic files to the current location to perform inverse kinematics on multiple trials of the same patient (and hence same static trial kinematics)</li><li>- Added automatic detection of missing markers to ensure that all markers are present at all frame numbers <i>before</i> processing inverse kinematics</li><li>- Configured coordinate systems for both MECH and PHYSIO labs at Melbourne University (<i>\MatlabUtils\CoordChangeLabs</i>)</li><li>- <b>getKinetics.m</b> now also outputs data in BOTH video and analog frame rates rather than just at the video rate via the <code>outputFrameRate</code> variable inside the m file.</li><li>- Added small feature to check that directories exist before attempting to save plot images to them.</li></ul> |
| V1.2 | Jan/Feb 2008  | <ul style="list-style-type: none"><li>- Fixed bugs in the installation process</li><li>- Removed inverse kinematics / dynamics scripts from the toolbox as they should be treated separately since they are model dependant.</li><li>- Remade <b>getKinetics.m</b> for support for any number of force plates as well as automatic event detection from the C3D file rather than input relevant frame numbers.</li><li>- Modified the format of <b>loadlabels.m</b>.</li><li>- Made <b>trialPlot.m</b>, <b>normalizeCycle.m</b>, <b>getEMG.m</b> and <b>compareStance.m</b> more user-friendly and compatible with the new <b>keyEvent</b> auto detection.</li><li>- Renamed <b>filterData.m</b> to <b>filterDiffData.m</b> for clarity.</li></ul>  |

- Updated **exportc3d.m** function to be able to supply desired information such as offset, and scale parameters.
  - Added **getMarkers.m** function
  - Added **saveOpenSim.m** function
  - Added **writeXML.m** function
- V1.3 Mar-Jun 2008
- Added marker verification to the kinetic verification
  - **getKinetics.m** now supports any combination of plates for data extraction
  - Fixed GRMx calculation in **getKinetics.m**
  - Added **extractOpenSim.m** to extract and extend motion data
  - Remade the **getEMG** and **getMVC** functionality
  - Removed **getMVCauto.m**
  - Removed several other files to generalize the toolbox and make it inter-lab friendly.
- V1.5 June-Nov 2008
- added batch EMG processing tool (**batchEMGprocess.m**)
  - changed order of **writeXML**, and added generic variables to make it easier to configure.
  - Added **extractMotFile.m** for user friendly OpenSim motion file extraction / filtering / plotting & superimposing.
  - Refined the example file to extract and run a muscle actuated simulation using OpenSim command lines.
- V1.6 August 2009
- Fixed some bugs related to detection of force plate corners
  - Fixed plotting bugs in **extractMotFile.m**
  - Added **createEvents.m** to be able to create event labels in Matlab (non Vicon users)
  - Added **multipleEMGprocess.m** to extract multiple EMG cycles from a single C3D file
- V1.7 September 2010
- Fixed some bugs related to local force plate origins that would have affected trials where force plate origins are non zero in X and Z directions (usually are zero though but not always)
  - Added functionality for overriding the automatic detection of force plate sequence in **getEvents.m**.
  - Automatic detection of coordinate system from FP to VICON and FP to MODEL for *each* force plate, so now trials can be extracted where multiple force plates are individually orientated

- ***getKinetics.m*** now outputs an xml kinetics file (that links to the kinetics \*.mot file) to support OpenSim 2.0+ formats.
  - Updated ***writeXML.m*** to support OpenSim 2.0+ formats.
  - Additional options in ***extractMotFile.m*** (some are still in testing)
  - Added an additional example C3D file (down stair walking)
  - Reformatted and revised the user manual
  - Fixed situations where markers are not found in the markerset described in the ***loadLabels.m*** file. ***getMarkers.m*** now omits the marker altogether from the \*.trc file, rather than entering zeros in the marker location in the \*.trc file.
- V1.71    November    2010
- **IMPORTANT FIX:** Fixed some bugs related to center of pressure (CoP) calculation. In most lab setups, this fix will not be required and the correct CoP will be output. But there are a number of cases where slightly incorrect center of pressures were being reported. This version will correct the CoP calculation.
  - I have also added new section in the user manual detailing the process by which the center of pressure is calculated. The foot detection algorithm is also outlined in the manual.
  - Added flexibility to change the Butterworth filter order for filtering ground reaction data in ***loadlabels.m*** under the optional tag: `glab.FP.filterOrder`