GaitExtract Toolbox V1.6

User Manual

Musculoskeletal Biomechanics Laboratory
Department of Mechanical Engineering
The University of Melbourne

Compiled & Written by Tim Dorn (t.dorn@pgrad.unimelb.edu.au)

Table of Contents

1. Introduction

This toolbox is primarily used for extracting and processing experimental gait data from the Biomechanics lab in the Department of Mechanical Engineering at the University of Melbourne. It has been configured for this lab specifically in terms of the coordinate systems used, force plate positions, and the mathematical models used for analysis. Use of this toolbox in other configurations / laboratories will require changes to the coordinate parameters in ***loadLabels.m***. This toolbox will be continually updated as other useful scripts are produced and added.

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 benefit for engineering applications. The basic purpose of this toolbox is to provide a platform to extract gait data directly from C3D format into to a Matlab environment where higher levels of flexibility are available for data processing. Data can then be exported for use with musculoskeletal models in OpenSim.

The basic idea is to first record the kinematics, kinetics, and (optional) EMG of a dynamic activity through motion capture & analog inputs (i.e. Vicon). The resulting trial is a file with a C3D extension. A working directory can be created with a copy of the C3D file, and a subject Matlab m script, which defines the pre-processing commands that define the 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 the 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 acknowledge myself and the toolbox if used in publications.

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

This toolbox (V1.5) was tested with Matlab 7.4.0 (2007a).

3. Toolbox Installation

The installation requires the Matlab path to always point to the directory where you have decided to install this toolbox. It also requires the installation of Motion Labs$^{®}$ C3D Server (C3D Viewer is handy to install and a freeware copy is provided with the toolbox, but is not necessary). The installation m file provided will install this application 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

To test the installation has been successful, try going through an example by going into the EXAMPLE folder and running ***testsubject.m*** in Matlab (see Section 6).

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, and analog channel names. Just like Vicon needs information about the lab 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.

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.

<u>*DO NOT MODIFY*</u>

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*

<u>*USER DEFINED (NEED TO MODIFY)*</u>

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 4.4).*

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.*

Feel free to add other labels as needed. The benefit of storing these labels in one file is one of modularity and consistency.

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 and Kistler. While both of these plates measure ground reaction forces (GRF) directly, only the AMTI plates measure the ground reaction moment about its origin (GRMo) directly.

In fact, both of these plates actually measure different components of force because they use different types of sensors in the hardware. For example, the AMTI plates (http://www.amti.biz) incorporates strain gauges mounted on each corner of the plate to directly measure the main six components of force using six channels (*Fx, Fy, Fz, Mx, My, Mz*).

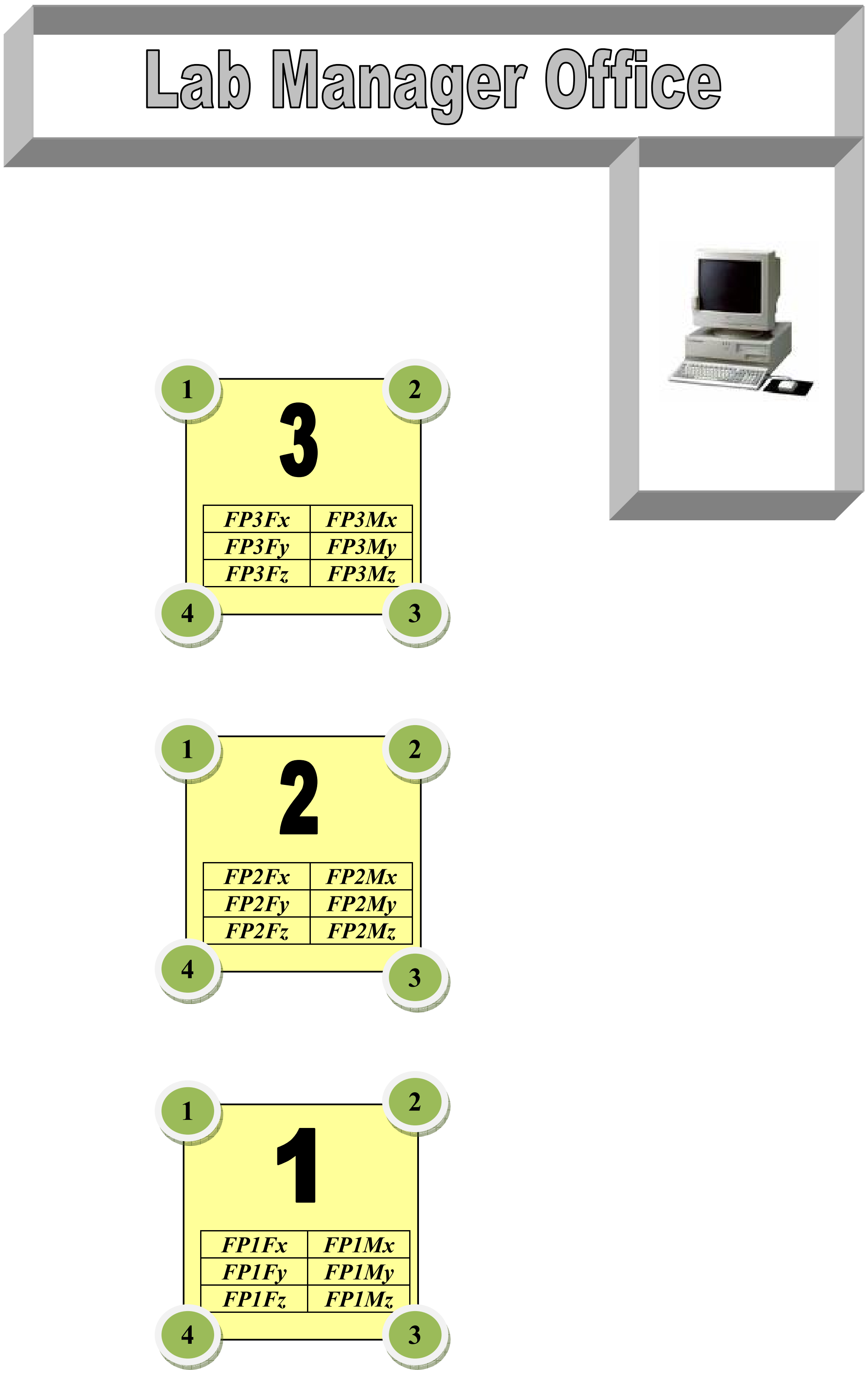
The Kistler plates (http://www.kistler.com) on the other hand consists of a base frame on which four piezoelectric crystal 3-component force sensors are mounted which use eight channels to output force (Fx(1+2), Fx(3+4), Fy(4+1), Fy(2+3), Fz1, Fz2, Fz3, Fz4). This clearly requires several equations to calculate our required six force components that the AMTI plates give us directly (Kistler say 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).

This toolbox assumes AMTI plates are in use such that the required force parameters (Fx, Fy, Fz, Mx, My, Mz) come directly from the c3d file. The equations used come from http://www.kwon3d.com/theory/grf/cop.html

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. The schematic in Section 4.3 illustrates the correct force plate setup in the gait lab.

Note also that 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).

4.3. Example force plate configuration



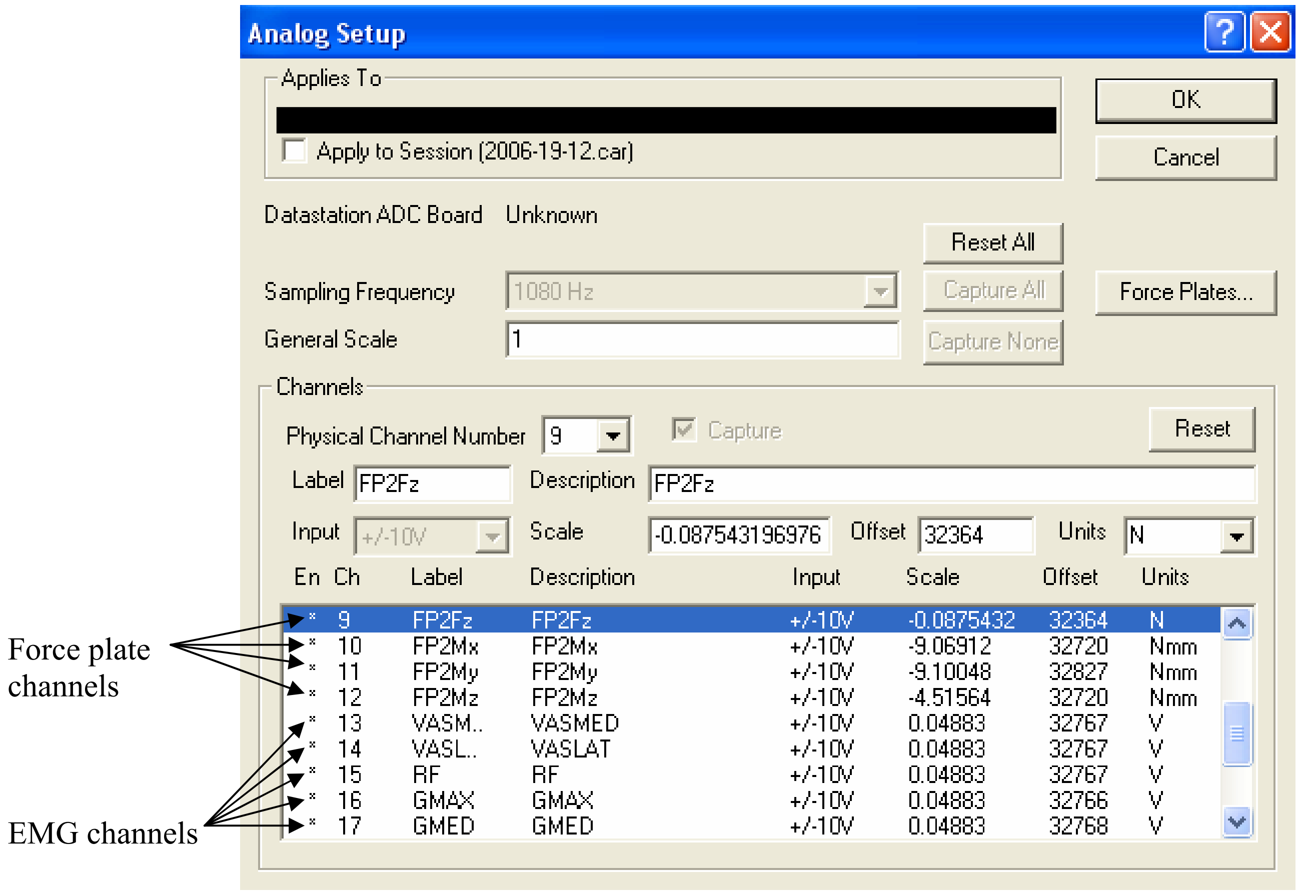
Values in the table denote the analog channel names
Values in the green circles denote the corner numbers

4.4. Analog EMG Configuration

For any analog setup, it is important that the channels are labeled correctly. The analog setup can be loaded in Vicon → System Menu → Analog Setup

Firstly, check that the force plate channel labels are correct from Section 4.3. 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 the illustration below. It is very important that EMG channel labels match the labels of glab.[emgSet] in **loadLabels.m** (See Section **Error! Reference source not found.**) for data extraction to occur correctly. Refer to the Vicon manual for help on setting up analog channels.



Applies To
[redacted]
☐ Apply to Session (2006-19-12.car)

Datastation ADC Board Unknown

Sampling Frequency: 1080 Hz

General Scale: 1

Channels

Physical Channel Number: 9 ☑ Capture

Label: FP2Fz Description: FP2Fz

Input: +/-10V Scale: -0.087543196976 Offset: 32364 Units: N

En	Ch	Label	Description	Input	Scale	Offset	Units
*	9	FP2Fz	FP2Fz	+/-10V	-0.0875432	32364	N
*	10	FP2Mx	FP2Mx	+/-10V	-9.06912	32720	Nmm
*	11	FP2My	FP2My	+/-10V	-9.10048	32827	Nmm
*	12	FP2Mz	FP2Mz	+/-10V	-4.51564	32720	Nmm
*	13	VASM..	VASMED	+/-10V	0.04883	32767	V
*	14	VASL..	VASLAT	+/-10V	0.04883	32767	V
*	15	RF	RF	+/-10V	0.04883	32767	V
*	16	GMAX	GMAX	+/-10V	0.04883	32766	V
*	17	GMED	GMED	+/-10V	0.04883	32768	V

4.5. 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 coordinate system
- Global space marker (VICON) coordinate system
- Model coordinate system

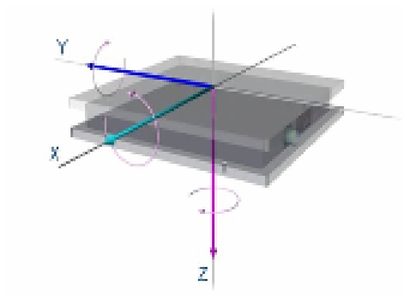
Converting from one system to another can get a little confusing and tedious, so the function **coordChange.m** exists (see Section 5) to perform rigid body transformations from one system to another after kinetics or kinematics have been extracted (kinetics are always extracted and outputted in the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

Note: the musculoskeletal model used with this toolbox is the Gait23xx_Simbody in OpenSim:

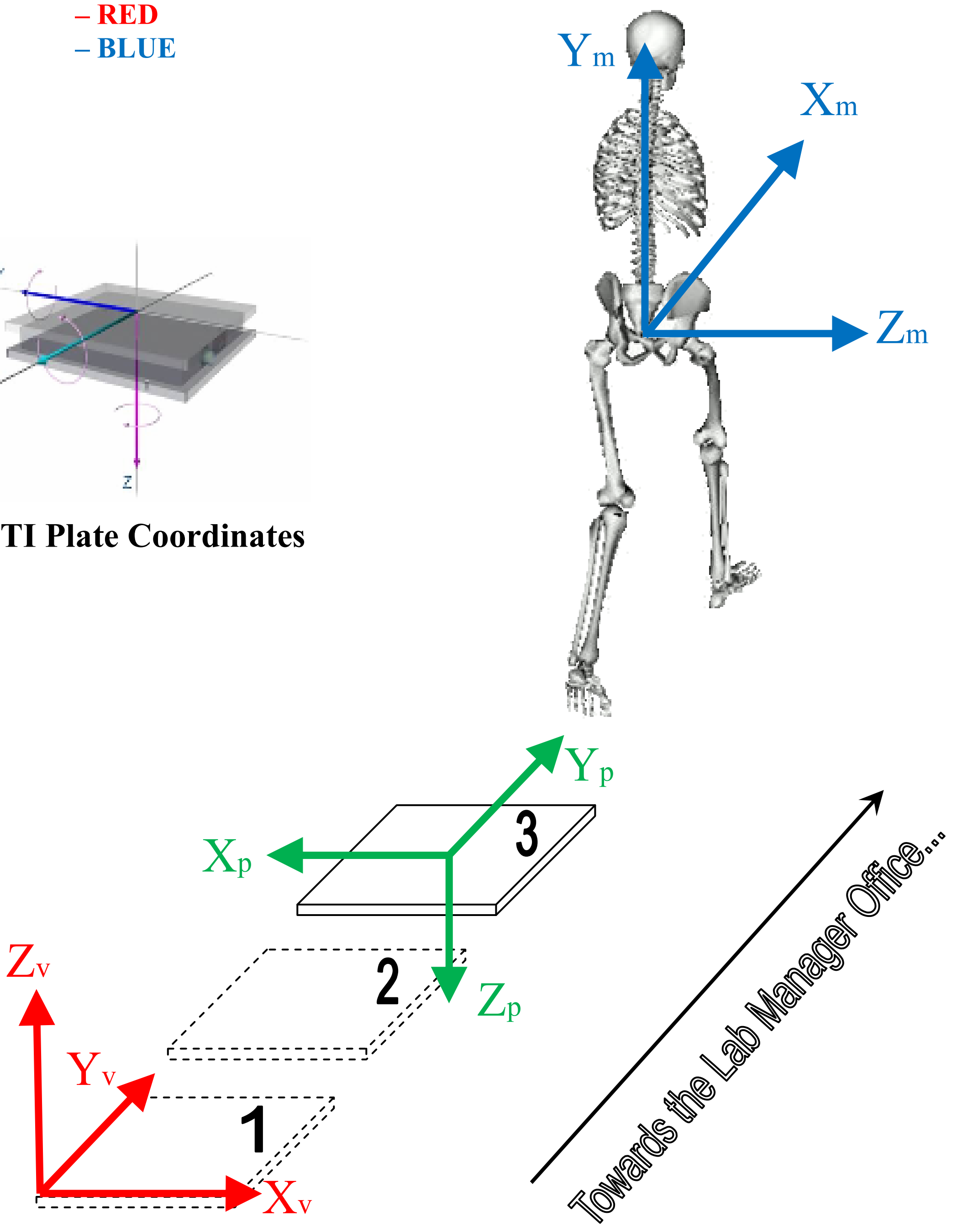
1. Anderson, F.C. and M.G. Pandy, *Dynamic optimization of human walking.* J Biomech Eng, 2001. **123**(5): p. 381-90.
2. Delp, S.L., et al., *An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures.* IEEE Trans Biomed Eng, 1990. **37**(8): p. 757-67.

For the current lab setup in the MechMelb gait lab, the illustration below outlines the default coordinate setup for each of the three systems.

Force plate	**– GREEN**
Vicon	**– RED**
Model	**– BLUE**



AMTI Plate Coordinates



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

I have attempted to cover everything that needs to change to set up this toolbox for alternate labs but I may have missed a few things, so feel free to email things I may have missed: t.dorn@pgrad.unimelb.edu.au. Keep in mind that this toolbox was originally designed for use in one lab, so it hasn't been tested in other labs. I would love to hear your comments & suggestions! Everything you need to change is located in one file: **loadLabels.m**

1) ***Set up force plate numbers:*** You need to set up the standard force plate orders for gait trials along different Vicon axes. If you don't give a *FP_order* input to **getEvents.m**, these standard force plate orders will be used. This requires a numbering system to the set of force plates in the laboratory. This option comes in handy if for example you are performing a running task where a force plate may be skipped due to a large stride length.

   EXAMPLE

   ```
   glab.FP.string = '%s%d%s';       % Prefix, PlateNum, Suffix

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

   glab.FP.verticalForceIndex = 3;
   ```

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

X force:	Fx*i*	X moment:	Mx*i*
Y force:	Fy*i*	Y moment:	My*i*
Z force:	Fz*i*	Z moment:	Mz*i*

   Z is the vertical direction (in force plate coordinates)

2) ***Set up coordinate systems:*** You need to define the rigid body transformations between the three main coordinate systems (Force plate, Vicon, and Model)

   EXAMPLE

   ```
   glab.dirVec.FPMODEL = [2, -3, -1];     % FP coords -> Model coords
   ```

   For example, the above transformation vector states that
   Model (X) = Force Plate (Y)
   Model (Y) = Force Plate (-Z)
   Model (Z) = Force Plate (-X)

   This transformation can easily be seen on the diagram on the previous page.
   This needs to be input for *FP* $\rightarrow$ *Model* and *Vicon* $\rightarrow$ *Model*

3) ***Set up marker offset:*** 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 is handy because global (Vicon) origins may differ from lab to lab and not necessarily correspond to the global origin in the OpenSim environment.

4) ***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.

5) ***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)

   EXAMPLE

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

6) ***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 **Error! Reference source not found.** 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. Detailed Function List

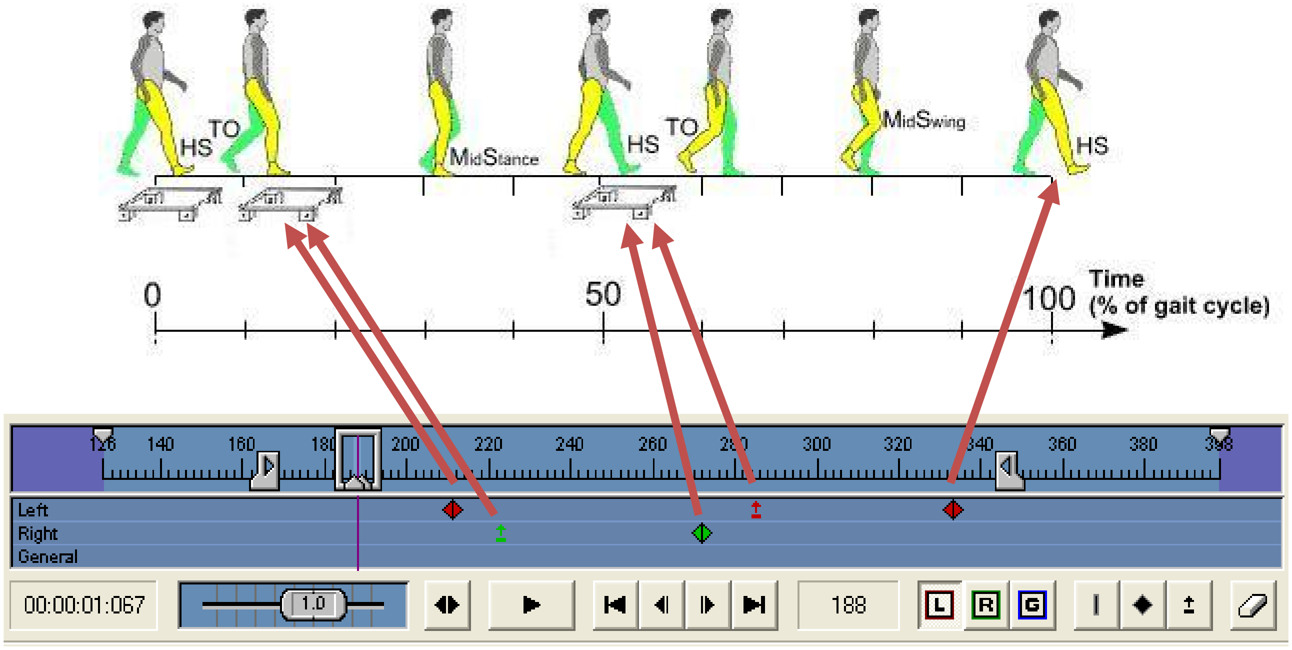
The following pages will explain in some detail the functions contained within the toolbox. This section will be continually updated as more functions are created. If anyone has made and tested their own function, it would be appreciated if you could give it to Tim to add to this ever-growing list to maintain the toolbox.

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

For the purposes of this toolbox, a gait cycle is defined from heel strike to heel strike (see diagram below), and it is assumed that the whole gait cycle is captured on three force plates (this is optimal but not necessary) and events should be labeled accordingly in Vicon (i.e. all heel strikes and toe offs should be labeled because the toolbox uses these events to determine the boundaries of the gait cycle).



For consistent labeling in Vicon, the following guidelines may be useful:
FootStrike (FS) – defined as the frame just before the GRF vector appears on the foot
FootOff (FO) – defined as the frame just after the GRF vector disappears from the foot

Please report any suggestions / bugs to Tim Dorn: t.dorn@pgrad.unimelb.edu.au

[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.

C3Dkey =

getEvents(c3dFile, direction*, FP_order*)

Description: Extract events and general trial details from C3D files to a Matlab structure (used by other functions).

Version: May 2009

Inputs: c3dFile = the name of the c3d file

direction = direction of gait / direction that the subject is facing
IN GLOBAL VICON COORDINATES
(1 = X -1 = -X)
(2 = Y -2 = -Y)

FP_order* = Vector containing the order of force plate numbers stepped on during the trial. This parameter can be neglected if the trial is NOT a dynamic general trial, but otherwise it must be included.
e.g. Walking on FP3, FP2, FP1 → *FP_order = [3 2 1]*
e.g. Running on FP1, FP3 (FP2 is skipped due to the large stride length → *FP_order = [1 3]*

Note: direction and FP_order are required here if you will be processing kinetics using this toolbox

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.dirVec.FPMODEL: direction vector from FP coords → Model coords
C3Dkey.dirVec.VICMODEL: direction vector from Vicon coords → Model coords
C3Dkey.dirVec.FPVICON: direction vector from FP coords → Vicon coords

C3Dkey.dirVec.MODELFP: direction vector from Model coords → FP coords
C3Dkey.dirVec.MODELVIC: direction vector from Model coords → Vicon coords
C3Dkey.dirVec.VICONFP: direction vector from Vicon coords → FP coords

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 (HS or TO) 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 toe off, right heel 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 4.6 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:	Jan 2009

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
- 2= plot kinetic graphs & kinetic verification (model coords)
- 3 = plot kinetic graphs & kinetic verification (vicon coords)
- 4 = plot kinetic verification only (model coords)
- 5 = plot kinetic verification only (vicon coords)

filterFreq* = filter frequency for GFR, GRMo (optional)
< 0 means no filtering is done
Should be used with caution.

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	CoP(1,:) Right Foot --> CoP X
GRF(2,:) Right Foot --> GRF Y	CoP(2,:) Right Foot --> CoP Y
GRF(3,:) Right Foot --> GRF Z	CoP(3,:) Right Foot --> CoP Z
GRF(4,:) Left Foot --> GRF X	CoP(4,:) Left Foot --> CoP X
GRF(5,:) Left Foot --> GRF Y	CoP(5,:) Left Foot --> CoP Y
GRF(6,:) Left Foot --> GRF Z	CoP(6,:) Left Foot --> CoP Z
GRMo(1,:) RF -> GRM X about FP origin	GRMx(1,:) RF -> GRM X about CoP
GRMo(2,:) RF -> GRM Y about FP origin	GRMx(2,:) RF -> GRM Y about CoP
GRMo(3,:) RF -> GRM Z about FP origin	GRMx(3,:) RF -> GRM Z about CoP
GRMo(4,:) LF -> GRM X about FP origin	GRMx(4,:) LF -> GRM X about CoP
GRMo(5,:) LF -> GRM Y about FP origin	GRMx(5,:) LF -> GRM Y about CoP
GRMo(6,:) LF -> GRM Z about FP origin	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
titlsize = 13;              % Title font size
```

markers = getMarkers(C3Dkey, markerSetName, filter*)

Description: Extract marker position data from a c3d file.

Version: Apr 2008

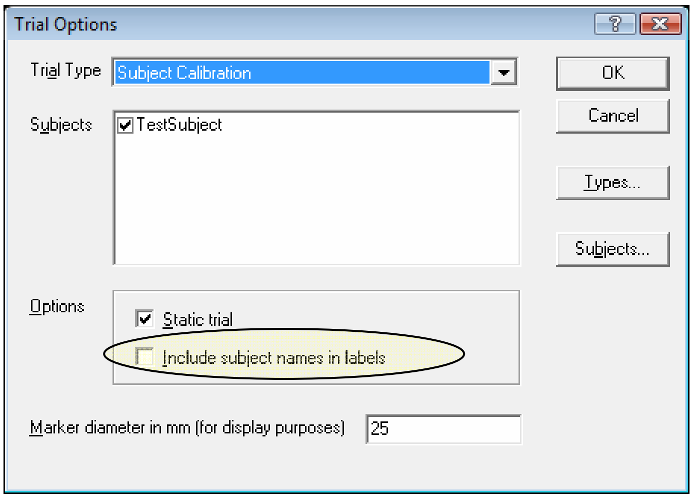
Inputs: C3Dkey: the C3D key frames structure from getEvents

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

Notes: It is vital 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.


Trial Options
Trial Type: Subject Calibration
Subjects: [✓] TestSubject
Options: [✓] Static trial
[ ] Include subject names in labels
Marker diameter in mm (for display purposes): 25

Output is saved as a *.trc file used by OpenSim

[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.[muscLabel] = value)

MVCset = structure of cells containing information about the EMG labels

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

References:

ABC or EMG (page 30)

Bolgla, 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 Section **Error! Reference source not found.** & 4.6 for label descriptions

[eVecGlob, EMGVecGob] =

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 defualt), 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 =

writeXML(type, C3Dkey*)

Description: Write XML files for use in OpenSim.

Version: Nov 2008

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 = 'rra' --> create residual reduction (pass 1) XML setup file
type = 'cmc' --> create computed muscle control XML setup file
type = 'forward' --> create forward dynamics 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

the XML file is saved in the current working directory as
[C3Dkey.c3dFile]_Setup_[type].xml

Notes: The output files are templates only. They may need to be altered to conform to the precise simulation (i.e. paths, filtering frequencies, input files, etc.

6. Example Subject Trial

The EXAMPLE directory contains a static and dynamic walking C3D file (captured from Vicon motion capture) and a C3D file of a full walking cycle. It also includes an OpenSim model file as well as several refined setting XML files.

testWalkSimulation.m provides an example of the raw data extraction process. The M file should be self explanatory. Please note that conventional units are always used unless explicitly stated (i.e. kg for mass, m for distance).

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

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 aspects of muscle actuated simulations (https://simtk.org/home/opensim).

Vicon C3D Files

testStatic.c3d:	Static trial in natural pose
testWalking.c3d:	Self selected level walking gait

7. Revision Information

V1.0	February 2007	- Initial release
V1.1	March 2007	- Fixed some installation path bugs - **runinvKin.m** now also outputs marker positions (raw or filtered) in standard format - **runinvKin.m** 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) - Added automatic detection of missing markers to ensure that all markers are present at all frame numbers *before* processing inverse kinematics - Configured coordinate systems for both MECH and PHYSIO labs at Melbourne University *(\MatlabUtils\CoordChangeLabs)* - **getKinetics.m** now also outputs data in BOTH video and analog frame rates rather than just at the video rate via the outputFrameRate variable inside the m file. - Added small feature to check that directories exist before attempting to save plot images to them.
V1.2	Jan/Feb 2008	- Fixed bugs in the installation process - Removed inverse kinematics / dynamics scripts from the toolbox as they should be treated separately since they are model dependant. - Remade **getKinetics.m** for support for any number of force plates as well as automatic event detection from the c3d file rather than input relevant frame numbers. - Modified the format of **loadlabels.m**. - Made **trialPlot.m, normalizeCycle.m**, **getEMG.m** and **compareStance.m** more user-friendly and compatible with the new **keyEvent** auto detection. - Renamed **filterData.m** to **filterDiffData.m** for clarity. - 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