Mokka & BTK Tutorial

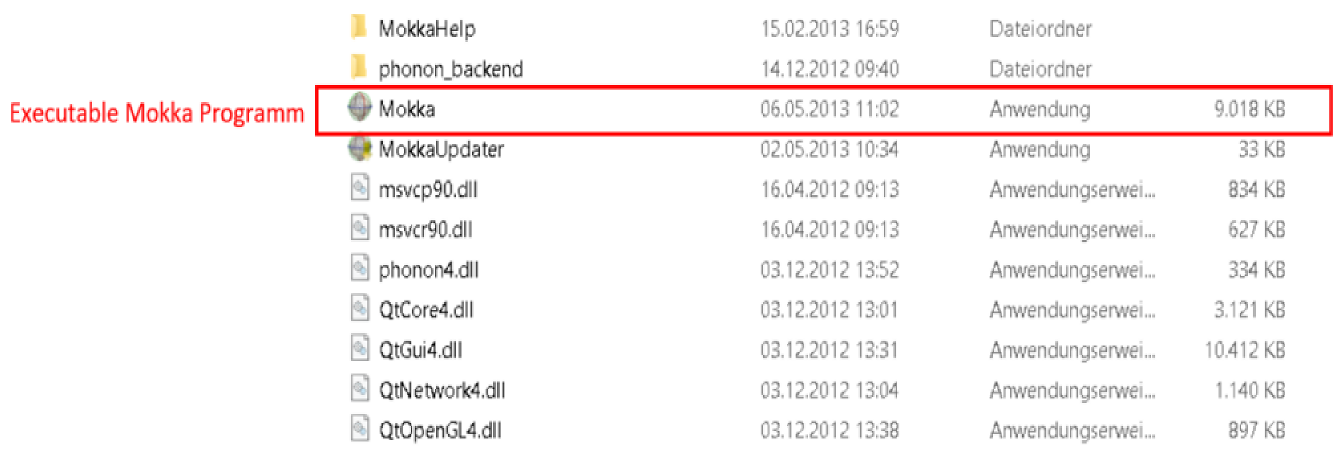
Description

In biomechanical research, we often collecting motion capture (marker trajectories) and force (analog data) data to calculate joint kinematics (joint angles and joint angular velocities) and joint kinetics (joint moments and joint power). Usually, this experimental data is collected with motion capturing software such as Qualysis Track Manager or Vicon Nexus, to name only a few. Regardless of the uses software to capture the biomechanical data the C3D file extension has been established as a standard for further data processing. Although C3D-files can be opened by the Motion Capturing Software, resource-saving and open source tools are available to visualize and manipulate C3D files.

C3D Visualization

Mokka is free to use and cross-platform software to easily analyze biomechanical data. Mokka can be downloaded from the website http://biomechanical-toolkit.github.io/ and requires no installation.

After downloading and unpacking the Mokka version for your operating system the folder should be looked like this:



MokkaHelp	15.02.2013 16:59	Dateiordner	
phonon_backend	14.12.2012 09:40	Dateiordner	
Mokka	06.05.2013 11:02	Anwendung	9.018 KB
MokkaUpdater	02.05.2013 10:34	Anwendung	33 KB
msvcp90.dll	16.04.2012 09:13	Anwendungserwei...	834 KB
msvcr90.dll	16.04.2012 09:13	Anwendungserwei...	627 KB
phonon4.dll	03.12.2012 13:52	Anwendungserwei...	334 KB
QtCore4.dll	03.12.2012 13:01	Anwendungserwei...	3.121 KB
QtGui4.dll	03.12.2012 13:31	Anwendungserwei...	10.412 KB
QtNetwork4.dll	03.12.2012 13:04	Anwendungserwei...	1.140 KB
QtOpenGL4.dll	03.12.2012 13:38	Anwendungserwei...	897 KB

Figure 1: Example Mokka for Windows

After opening Mokka.exe the graphical user interface (GUI) look like this:

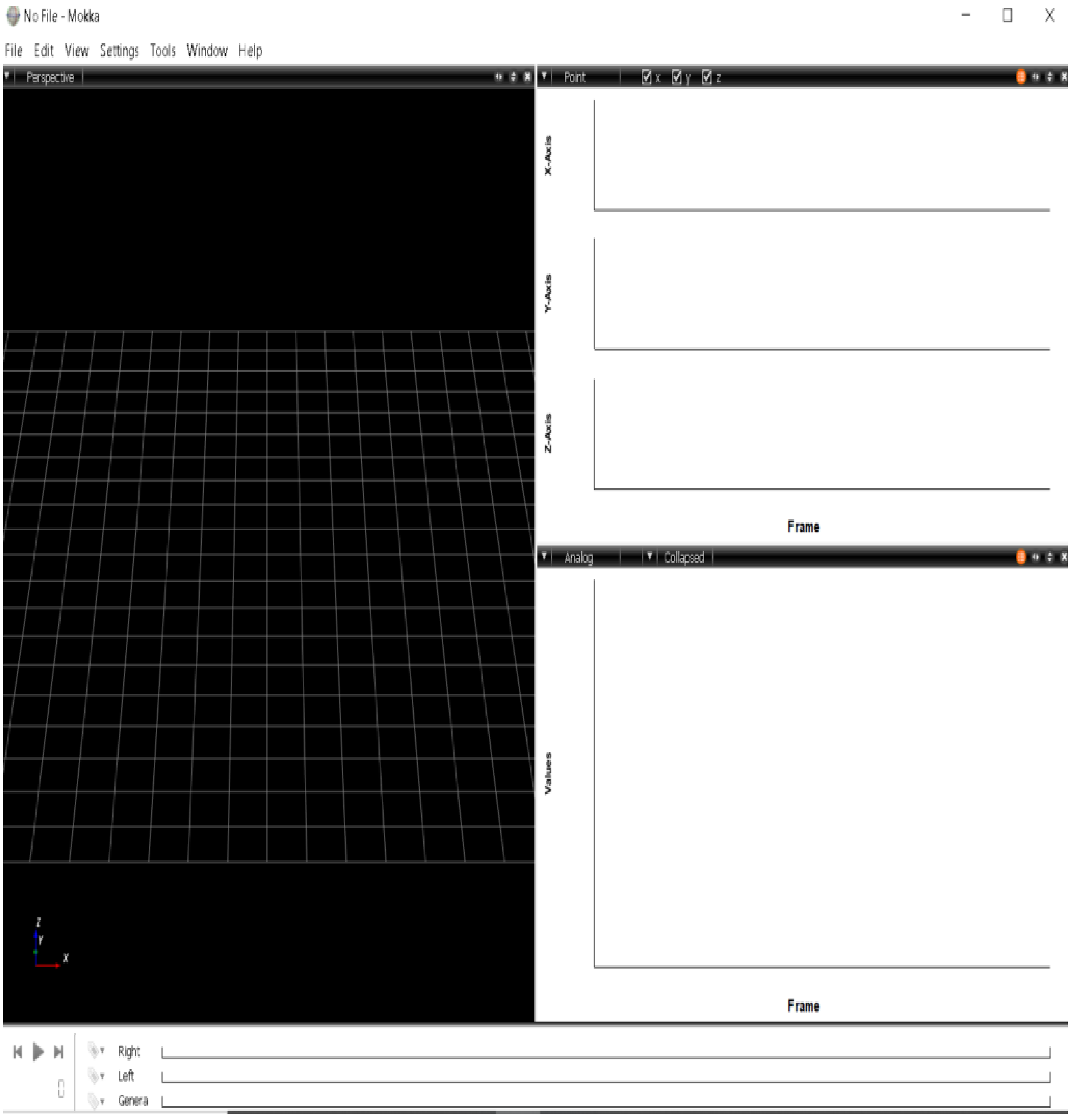


Figure 2: Grafical User Interface Mokka

1. Load the *Running_Example.c3d* file

Load a file by choosing File -> Open. After loading the *Running_Example.c3d* file the GUI should look **like** this:

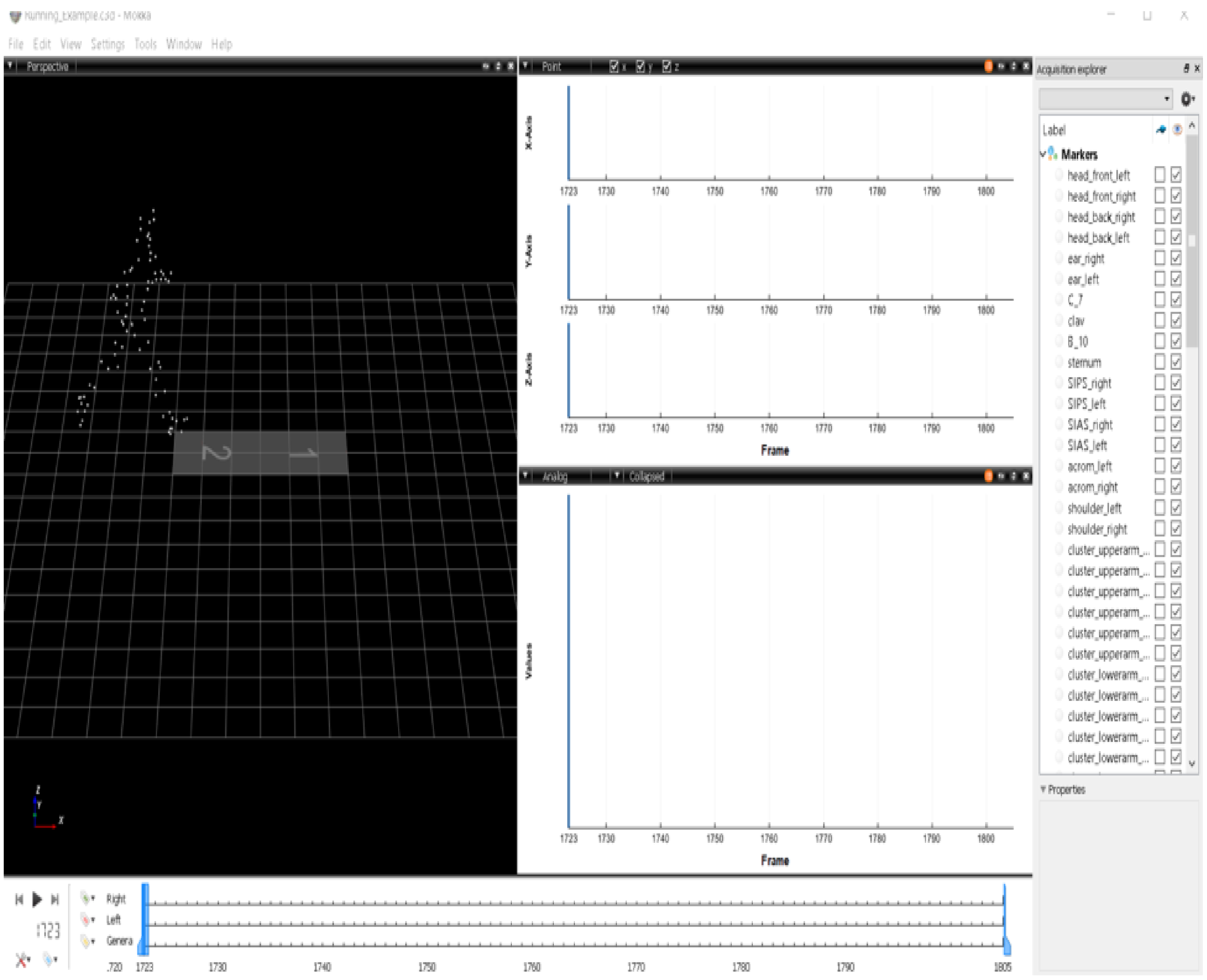


Figure 3: Mokka Graphical User Interface with loaded Experimental Data

Within the GUI you can explore the captured **3-dimensional experimental data**.

Each gray Point represents a spherical marker attached to the human body. By clicking on a specific marker, the name (often referred to as the marker label) is highlighted on the right-hand side **list box**.

2. Inspect the data

With the **play** button at the left lower corner of the GUI, you can see how the participant is running. Alternatively, you can use the **time bar** to inspect every frame (point in time) of your experimental data.

The **chart window** can provide further information about the single components of your data. Note if the chart window is not shown go to ***Window ->Layout -> 3D and Charts***. Now you can drag and drop labeled markers from the **List box** to the chart window, to see its X, Y, and Z coordinates over each instant in time.

Within the **list box**, you can also scroll down to the force platform data (if exist). For the provided example Force Platform #2 was used. You can expand the force platform #2 field and can visualize the force components (Forces and Moments) in the same way as the marker trajectories by drag and drop the component to the chart view.

However, when having many markers attached to your participant, visual inspection of your marker trajectories can become cumbersome. Therefore loading a model fitting to your data can be helpful.

3. Load a model

For this tutorial, we have already generated a model for our marker setup. To load the model go to **Edit -> Options**. Check the box *Use* ***default configuration*** and click ***on Choose*** *â€¦*. Select the file we have provided named ***full_body_Model.mvc***. After the operation has been completed the 3D view should contain different colored markers and connected segments.

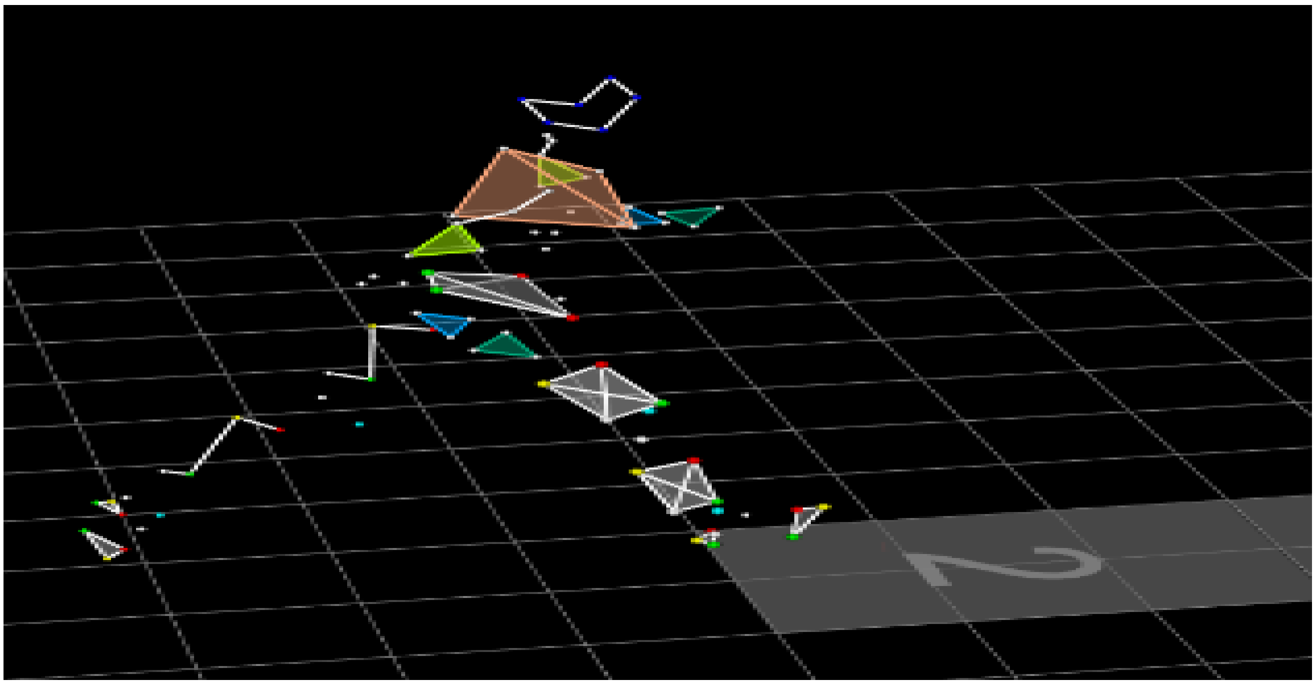


Figure 4: Experimental data with predefined segments and different colored markers.

Please note that the model has been adjusted to the used marker set. When using your experimental data with different marker names you need to define new segments according to your marker names. New Segments can be generated under ***Tools -> Model -> New Segment*****.**

4. Crop your data

For further data processing, it is sometimes useful to crop your C3D files. Mokka provides the ability to crop (or cut) your C3D files very easily. Within the **time bar**, you can use the **sliders** to the left and the right end of the time bar to crop the C3D file. The frames which are removed from the C3D file are marked in blue. When right-click anywhere in the **time bar** you can **select** ***Crop Frame of Interest*****.** Finally, you need to store the cropped file **(*File -> Save As or Save to overwrite the existing file*)**.

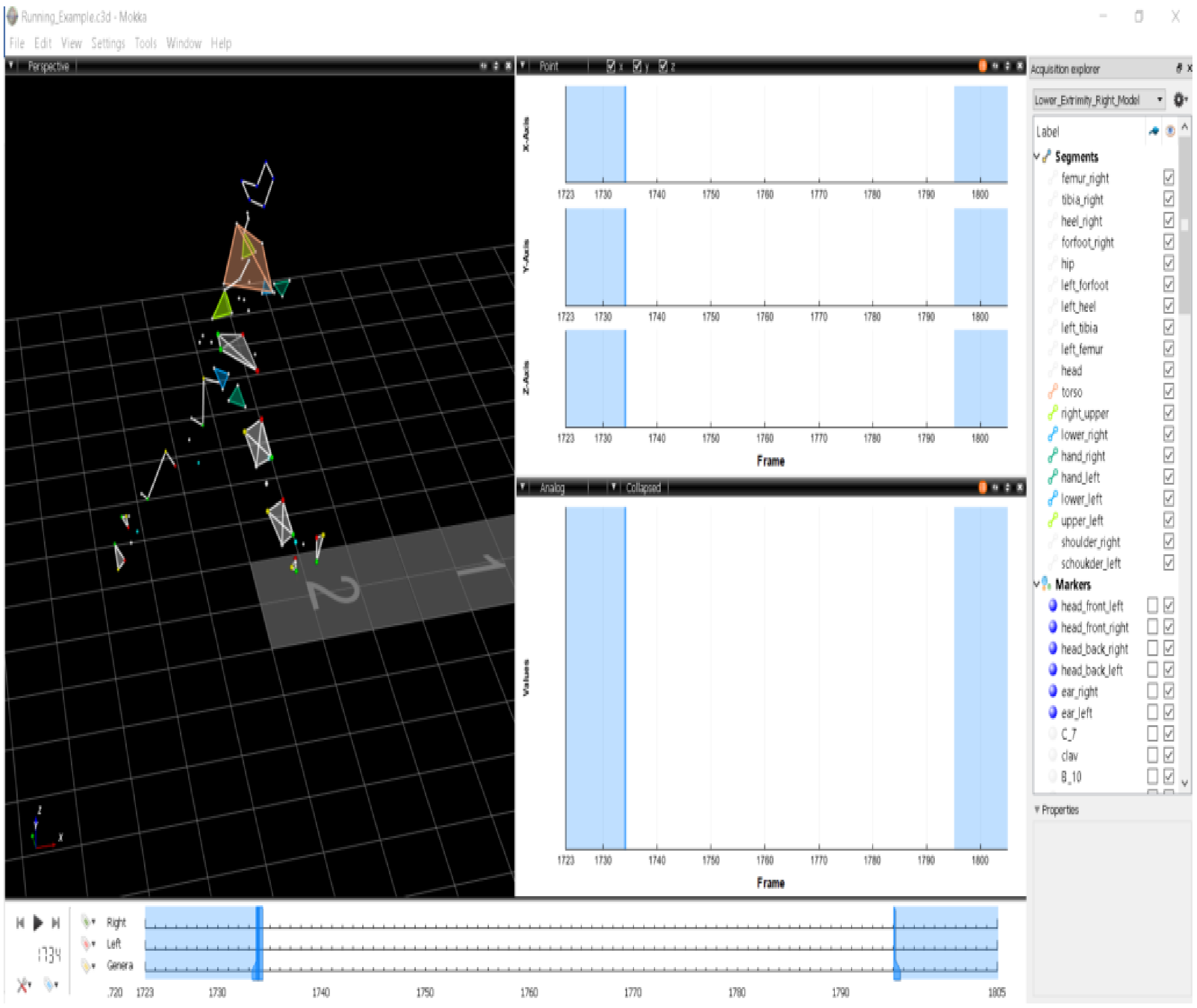


Figure 5: Cropping a C3D file within the Mokka Graphical User Interface

Although, Mokka is useful for visualization purpose it does not allow to calculate joint kinematics and kinetics. It becomes also extraordinary time consuming when much experiential data needs to be batch processed. Within the next step, we would like to show you how to read in C3D files into MatLab.

C3D File Loading with Matlab

For loading C3D files into your MatLab workspace you first need to download some additional, open-source functions. The **biomechanical toolkit (b-tk)** is a frequently used cross-platform toolkit for reading, writing, and manipulating C3D files.

First, you need to download, b-tk from the website (https://code.google.com/archive/p/b-tk/downloads). Please ensure to download the correct version for your operating system. For us, the version btk-0.3.0_Win7_MatlabR2009b_64bit.zip works best. After downloading the toolkit you need to unpack the archive by using a standard archive converter.

Now you need to tell MatLab where all the b-tk functions are located. Open MatLab and go to **Home Set Path Add with Subfoldersâ€¦** and selected the folder (in our example btk-0.3.0_Win7_MatlabR2009b_64bit, note use the unpacked data folder and not the compressed archive).

You can check if the Path is set correctly when typing **v = btkGetVersion** into your MatLab command window.

If MatLab outputs a version number (for example ‘0.3.0’) the path is set correctly.

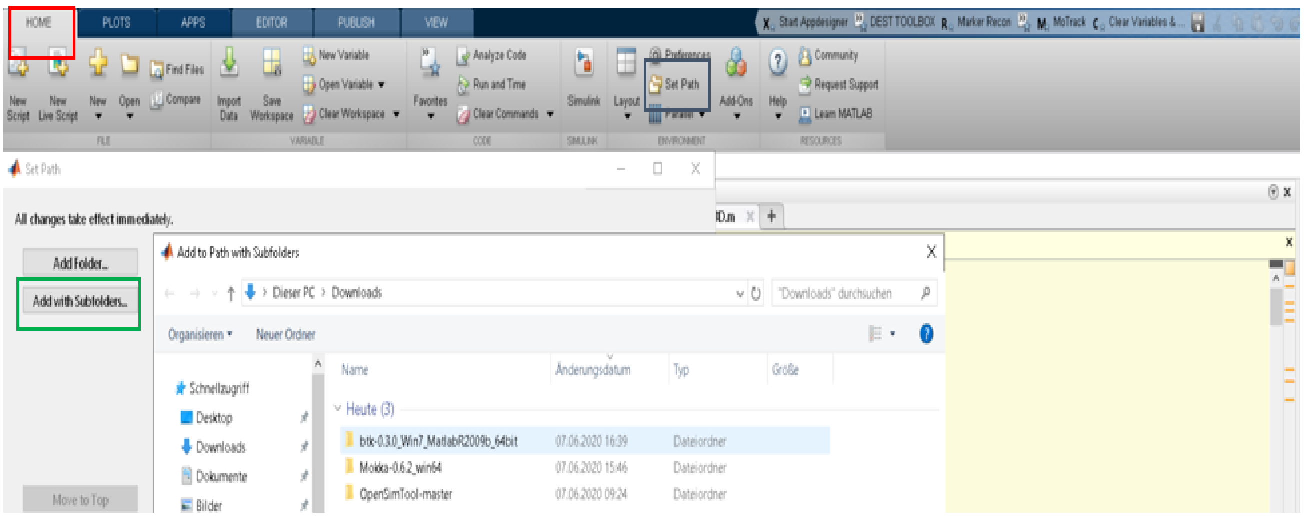


Figure 6: Add the b-tk functions to the MatLab path.

Working with BTK

You can now read, manipulate, and write C3D files with MatLab. For the full list of the b-tk function please refer to the following webpage http://biomechanical-toolkit.github.io/docs/Wrapping/Matlab/annotated.html

Here is just a simple Code the get started with b-tk and MatLab.

The code will show how to read and add Virtual Markers to a C3D file (static trail).

Furthermore, the code will show how to calculate stance time based on Force data and add touchdown and toe-off events to a C3D (rung trail). An additional Matlab script is provided (btkandmatlabexample.m):

```
%% some basic start commands
clc %clear the command window
clear all %clear the workspace
close all %close all open figures
v = btkGetVersion % get the current BTK version

%% Add the Joint Center of the right Knee to the Static Reference Trail as a v
%add a marker to the static reference trail
filename= ('Static_Example.c3d'); %the filename of the Static reference
```

```
acqstatic = btkReadAcquisition(filename); % read the static c3d file
[markersStatic, markersInfoStatic, markersResidualStatic] = btkGetMarkers(acqs
%the output for example markersStatic is a maatlab structure with all
%labeled markers and its X Y and Z componets
%The knee joint center is often calculated as the midpoint between the
%medial and the lateral epicondylus marker
            x1=markersStatic.epi_med_right(:,1); % x componets medial epicondy
            x2=markersStatic.epi_lat_right(:,1); % x componets lateral epicond
            y1=markersStatic.epi_med_right(:,2); % y componets medial epicondy
            y2=markersStatic.epi_lat_right(:,2); % y componets lateral epicond
            z1=markersStatic.epi_med_right(:,3); % z componets medial epicondy
            z2 =markersStatic.epi_lat_right(:,3);% z componets lateral epicond
            M_p = [(x1+x2)/2, (y1+y2)/2, (z1+z2)/2]; %calculate the midpoint h

[points, pointsInfo] = btkAppendPoint(acqstatic, 'marker' , 'V_Knee_Joint_Cent
%append a point to the Acquisition with the name
%'V_Knee_Joint_Center_right' and the coordinates from the Matrix M_p,
%markersResidualStatic.epi_lat_right are the residuals of the new virtual mark
%can simply be used from an existing marker
btkWriteAcquisition(acqstatic, 'Static_Example_with_Knee_Joint_Center_right.c3
%close the previously opened C3D file
btkCloseAcquisition(acqstatic)

%% Load a example file for Running and calculate Stance Time of the right leg
%% add Toutchdown and Toe off events to the C3D
%load a example file
filename= ('Running_Example.c3d'); %the filename of the running c3d
acqrunning = btkReadAcquisition(filename);  % read the running c3d file
%% get the marker data
[markers, markersInfo, markersResidual] = btkGetMarkers(acqrunning)
%markers contains all marker with its X Y and Z coordinates
%% plot a marker trajectory
figure(1)
plot(markers.C_7) % plot the marker coordinates of the C_7 marker
legend ('X', 'Y', 'Z')
legend boxoff
box off
ylabel ('Coordinates in mm')
xlabel ('Frame Number Marker Data')
%% get the raw force data
[forceplates, forceplatesInfo] = btkGetForcePlatforms(acqrunning)
% get the Ground Reaction Forces
grw = btkGetGroundReactionWrenches(acqrunning)
%% plot the vertical Ground Reaction Force
figure(2)
plot(grw(2).F(:,end)) % in this example the participant hits the force plate 2
box off
xlabel ('Frame Number Force Data')
ylabel ('Vertical Ground Reaction Force')
%% calculate stance time based on ground reaction force
vGRF = grw(2).F(:,end);
%use a force threshold to determine the stance time
thresholdforce = 20% N when 20 N of the vGRF
[stance_frames_Analog_based]=find (vGRF>= thresholdforce) %stance_frames conta
stance_number_of_frames_Analog_based = length (stance_frames_Analog_based) %th
analog_frequency = btkGetAnalogFrequency(acqrunning) % get the analof frequenc
stance_time_Analog_based = stance_number_of_frames_Analog_based/analog_frequen

%% add Touchdown and Toe off as events into your C3d Files
%this must be based on the point frequency (the frequency of the markers)
btkClearEvents(acqrunning) %clear events if exist
%fet the first frame of the aqquistion
first_frame= btkGetFirstFrame(acqrunning)

af = btkGetAnalogFrequency(acqrunning); %analog frequency
pf = btkGetPointFrequency(acqrunning);  %marker frequency
fkratio = af/pf;
Down_sampled_Force = downsample(grw(2).F(:,end),fkratio); %The analoag data do

[stance_frames]=find (Down_sampled_Force >= thresholdforce) %stance_frames con
stance_number_of_frames = length (stance_frames) %the total number of frames w
analog_frequency = btkGetAnalogFrequency(acqrunning) % get the analof frequenc
stance_time = stance_number_of_frames/pf % stance time in seconds
TD = stance_frames(1,1)-2;
TO = stance_frames(end,1);

dt = 1 / pf;
ff = btkGetFirstFrame(acqrunning);
td = ((ff+TD)*dt);%in c3d time format
to = ((ff+TO)*dt);%in c3d time format

btkAppendEvent(acqrunning, 'TD', td, 'Right');
btkAppendEvent(acqrunning, 'TO', to, 'Right');
btkWriteAcquisition(acqrunning, 'Running_Example_with_Events.c3d') % Please ch
%% close the previously opened C3D file
btkCloseAcquisition(acqrunning)

%% have fun :)
S(1) = load('gong');
sound(S(1).y,S(1).Fs)
```

Example Data

- BTK and MOKKA Tutorial Example Data839.02 KB
  2023-09-01
  Downloads: 736
  BTK and MOKKA Tutorial
  Download