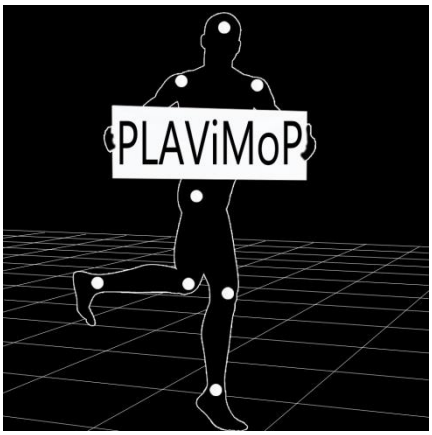
# PLAViMoP Software

User Manual

*Written by Arnaud DECATOIRE*

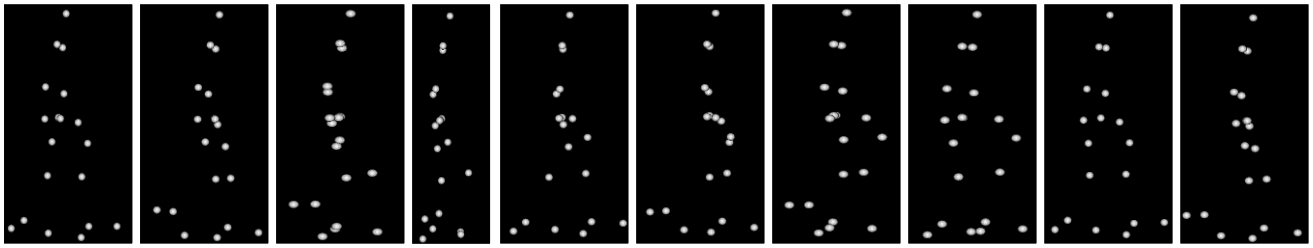
*Version 1 – 2018*



# CONTENTS

# 1. General presentation

PLAViMoP software is the acronym of **P**oint-**L**ight **A**ction **Vi**sualization and **Mo**dification **P**latform. It is composed of a Matlab graphical user interface interacting with the open source free software Mokka [1]. *The main goal of this platform is to propose a tool that allows to visualize and transform movements displayed as point-lights in a 3D scene. A point-light represents the 3D trajectory of a marker placed on a human, an animal, an object… The marker movement is generally tracked with motion capture system consisting in multiple optoelectronic cameras (Vicon, Motion, Qualisys, Optitrack …). Here is an example of a point-light sequence representing a walking human movement:*



# 2. Installations

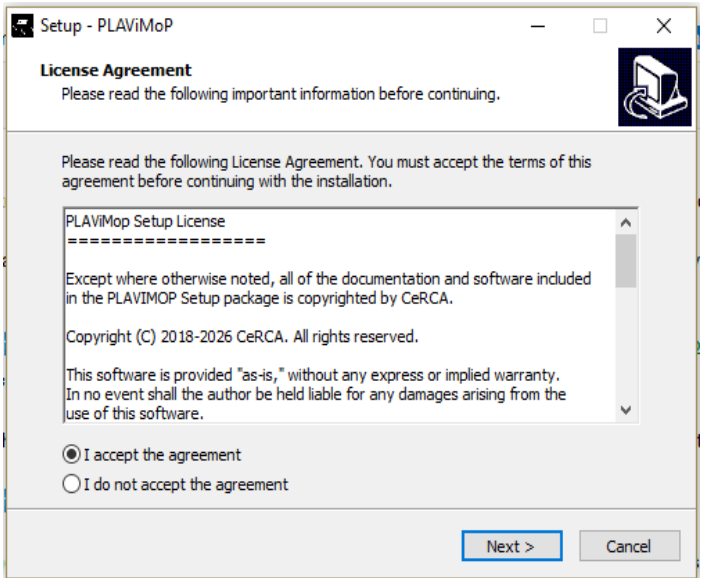
Both Matlab interface and Mokka software are required, as well as Windows 64 bits system and an internet connection for installation.

Matlab interface installation requires administrator rights.

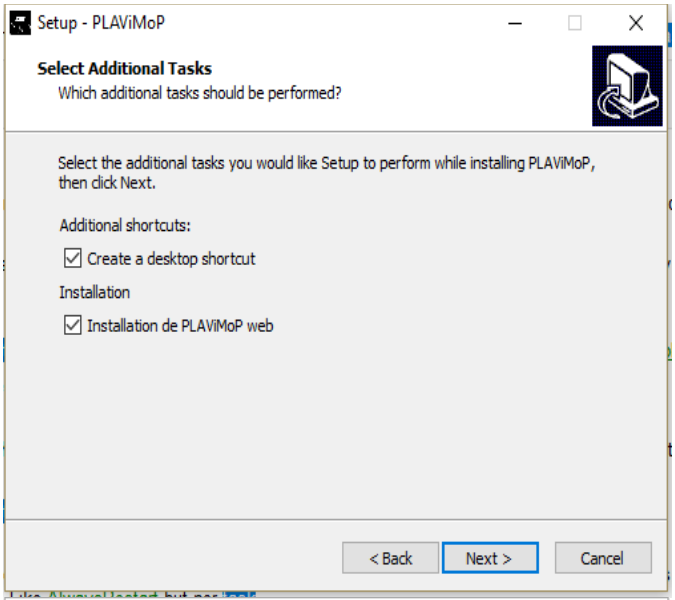
The minimal screen resolution is 1024 x 900 pixels. However, the application has been optimised for a 1920 x 1080 pixels screen resolution.

## 2.1. PLAViMoP installation

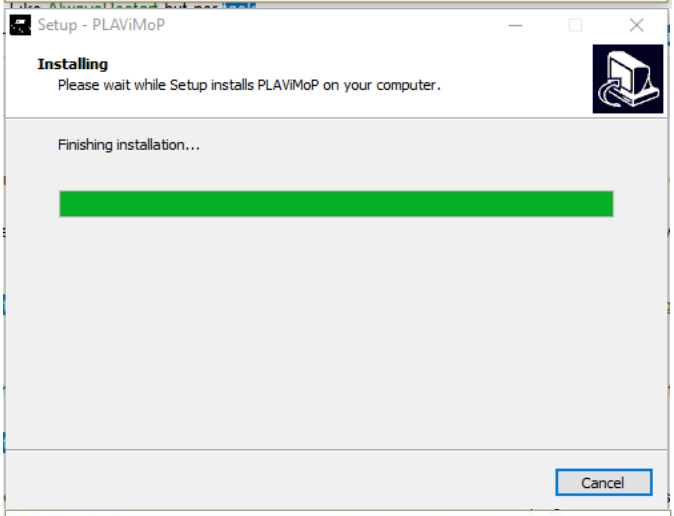
Go to http://plavimop.prd.fr and download the software installation package ***setupPlavimop.exe***



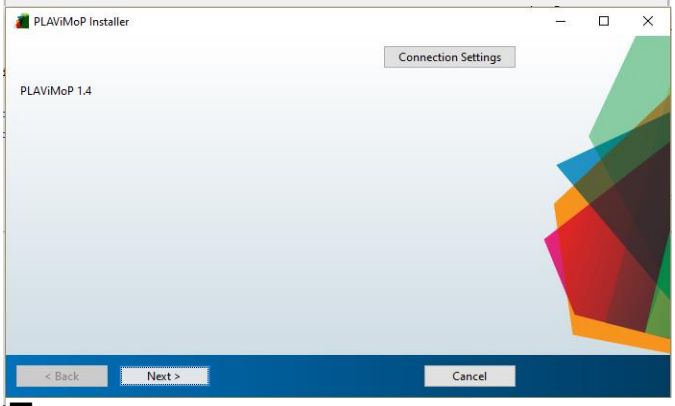

After launched the installation, you must accept the agreement.



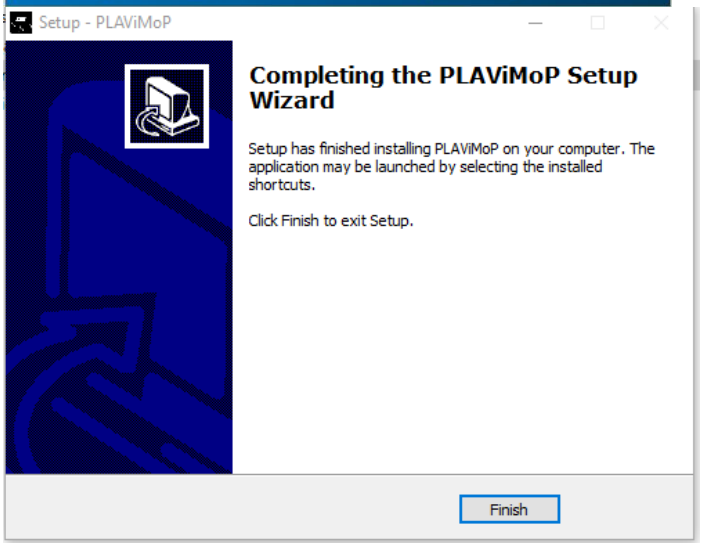

The installation of ***PLAViMoP web*** must be checked.
Press “next”.




The first part of the installation is ended. You need to wait time … perhaps long time.




The next ***MatLab*** part begin to install automatically.
The JRE *Matlab* will be installed if is not yet.
You press “next” to next all screens




Great ! !

If you see this screen, PLAViMoP software is succesfully installed.

Have enjoy !

After installation, a folder link “PLAViMoP ressources” have been created on your desktop.

# 3. Input / output formats

## 3.1. Input data format

Only the C3D format is supported by PLAViMoP software. This is the standard format of motion capture file. The file should contain only 3D trajectories of a set of markers (no force plate data, no analogical channel …). The X, Y and Z components are expressed in **millimetres** in a global reference frame (forward direction given by X-axis, vertical direction given by Z-Axis pointing upward and lateral direction given by Y-Axis pointing to the left of the subject/object). The number of markers is not limited, but for example, the common human motion set of markers is listed in the following table.

| **Markers names** | | **Locations / Descriptions** |
|---|---|---|
| **Right** | **Left** | |
| R_Heel | L_Heel | *Back of the right and left heels* |
| R_Toe | L_Toe | *Top of the right and left big toes* |
| R_Ankle | L_Ankle | *Middle of external and internal ankles markers* |
| R_Knee | L_Knee | *Middle of external and internal knees markers* |
| R_Hip | L_Hip | *Right and left hips centres of joint computed as* [5] |
| R_Shoulder | L_Shoulder | *Right and left acromion* |
| R_Elbow | L_Elbow | *Right and left lateral epicondyle* |
| R_Wrist | L_Wrist | *Right and left radial styloid* |
| R_Finger | L_Finger | *Right and left distal phalanx of the index fingers* |
| Head | | *Mean point of right and left front head markers and right and left back head markers* |

The number of frames of the C3D file is not limited as well as the frame rate. But, keep in mind that high number of frames and/or high frame rate will result in a time consuming process. For example, the provided C3D file are sampled at 100 Hz and contain around 200 frames.

If you do not have C3D files, you can create them from a .CSV file containing all information on the PLD (3D time histories of markers, names of marker components and a time column). For this, you can use the plug-in “CSVtoC3D” proposed in PLAViMoP (for more information on the use of plug-in, see here).

Another Mokka specific configuration is used to set the display preferences. This file creation is described here.

## 3.2. Output data formats

After the process, the data can be saved as C3D file as well as a video (*.avi file) via PLAViMoP. The operations are detailed here.

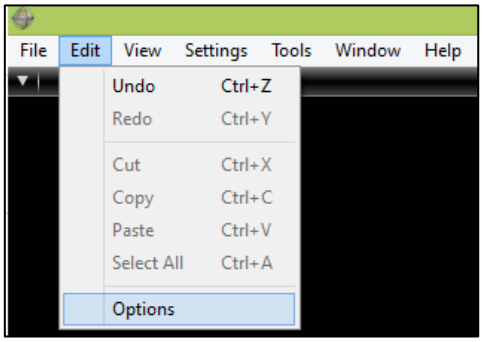
# 4. Mokka functionalities

Only the PLAViMoP specific Mokka functionalities are detailed in this section. The other functionalities can be found through the help menu of Mokka.

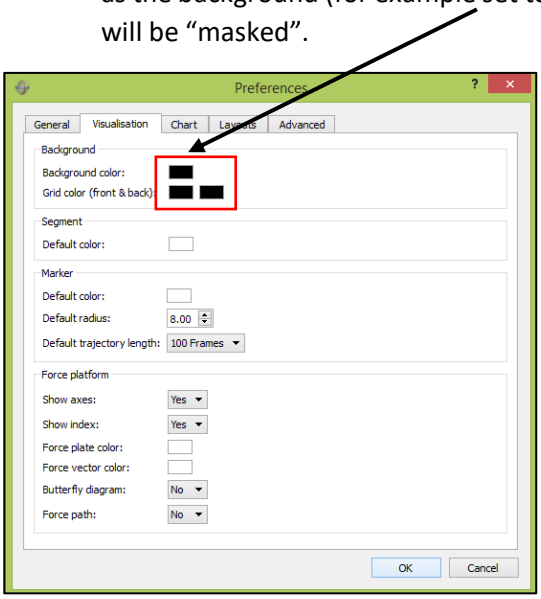
## 4.1. Mokka recommended settings

By default, Mokka display a 3D scene with a floor grid. To hide this grid, here are the steps to follow:

1) Go to Edit/Options menu of Mokka.




2) In the opening window, go to **Visualisation tab** and set the grid colours (front & back) to the same colour as the background (for example set to black), and click OK. The next time Mokka will be launched, the grid will be "masked".

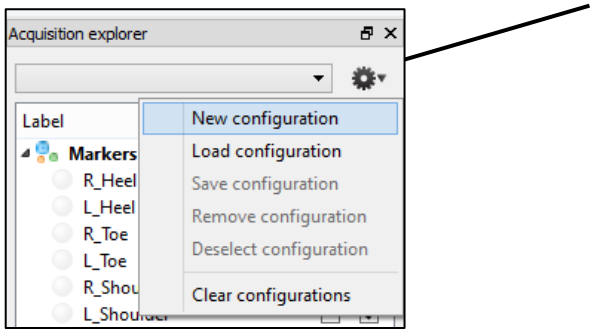



## 4.2. Mokka visualization configuration file

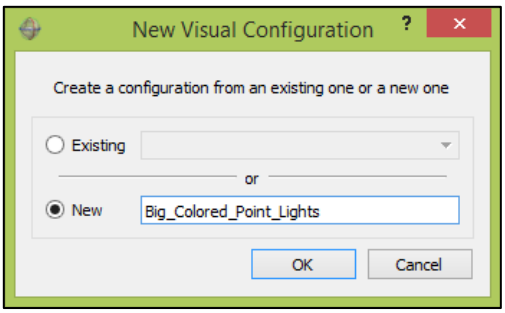
The way Mokka displays the content of a C3D file depends on a Mokka Visualization Configuration (*.mvc) file.

To create a personal MVC file:

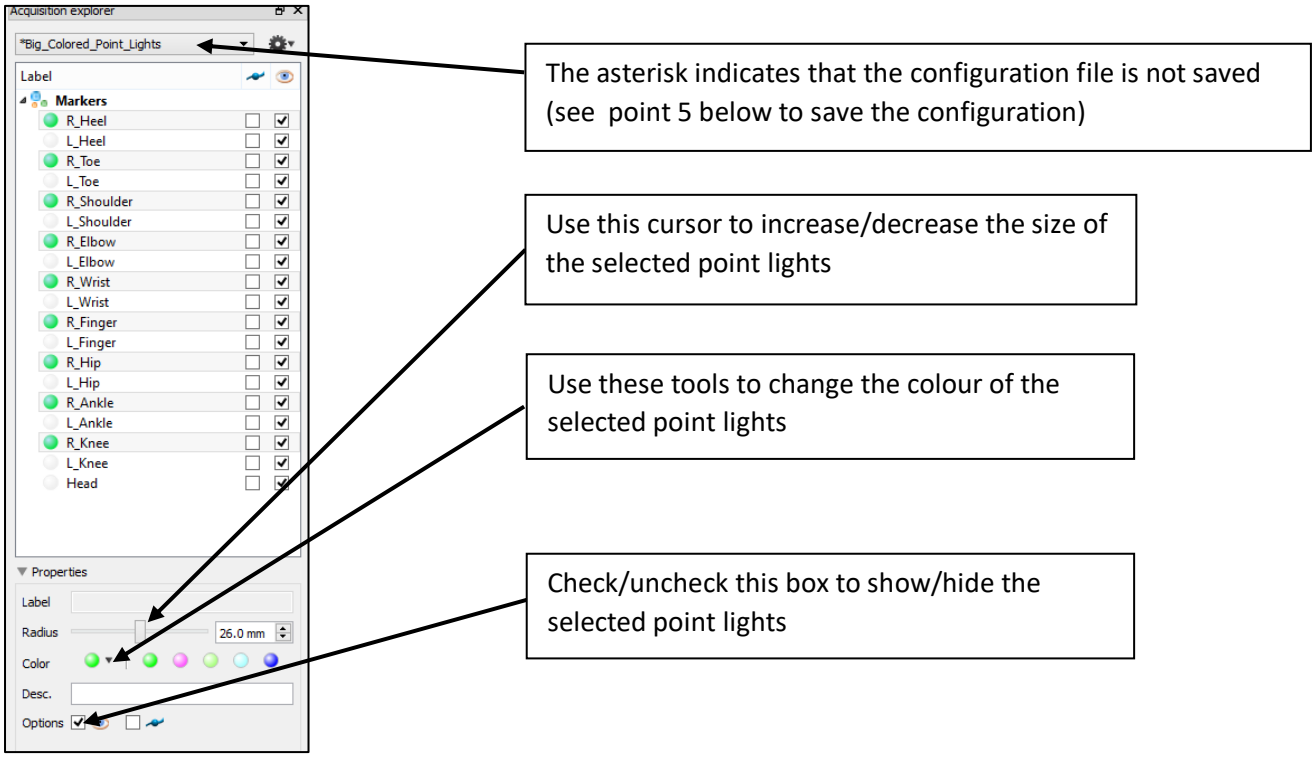
1) In **Acquisition explorer tab** of Mokka, click on gearing icon and then on **New configuration**.




2) In the opening window, give an explicit name for the new configuration and click **OK**.

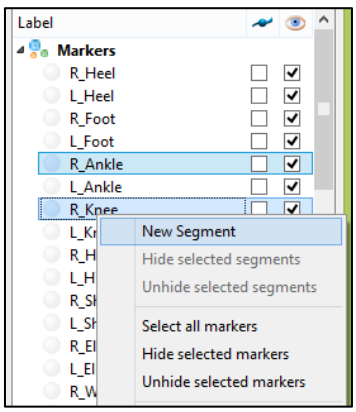


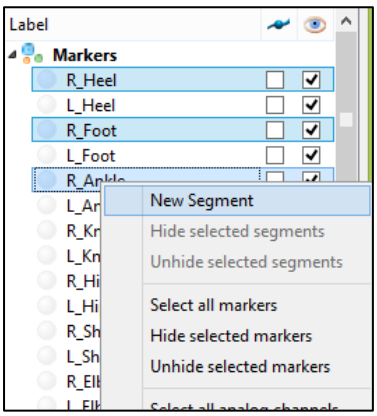

3) To modify the properties of point lights, select the marker(s) in the list and modify their properties.




4) Create links between Point lights

The way the point lights are linked together (it is not mandatory to link the point lights) is stored in the *.mvc file. By default, points are note linked together, however it is possible to connect them.

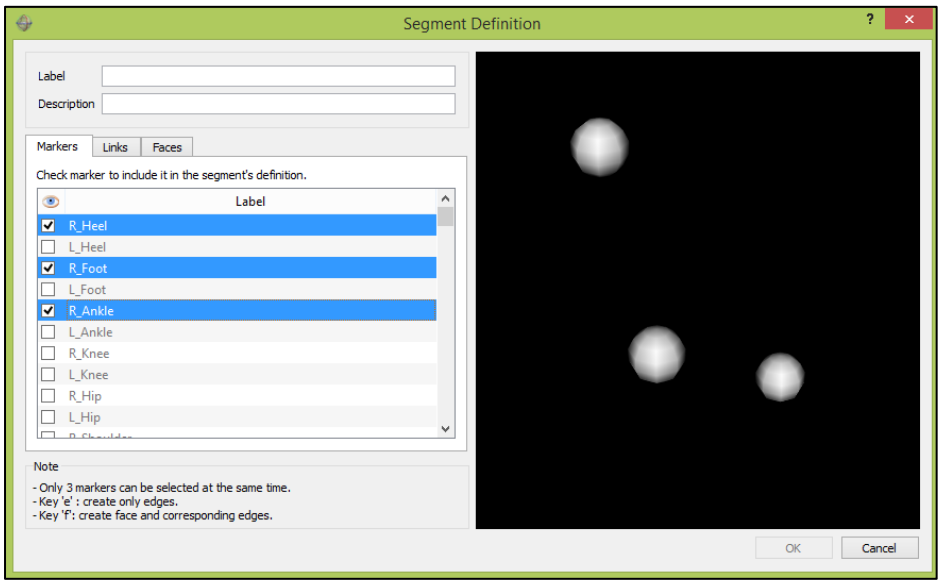


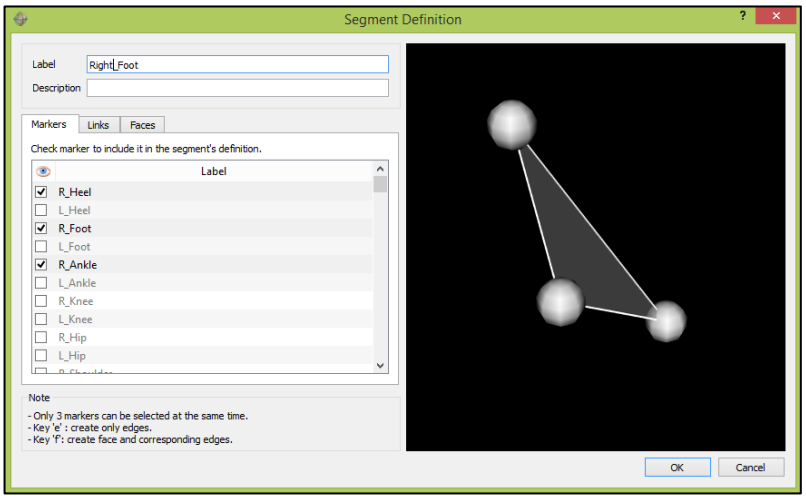

To create a link between 2 or 3 markers, first select the markers in the markers list (hold Ctrl + Left click).

Then right click and select **New Segment** in the floating menu.

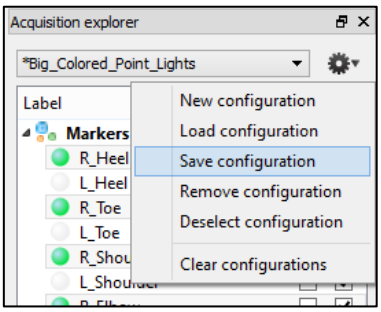
In the **Markers tab** of the new window, check and re-select (hold Ctrl + Left click) the 2 or 3 markers.




Then press “e” to create a stick between 2 markers and/or “f” key to create a face between 3 markers.
Give a segment name in the **Label field** and click **OK**.
Finally, choose a segment colour in the properties area.

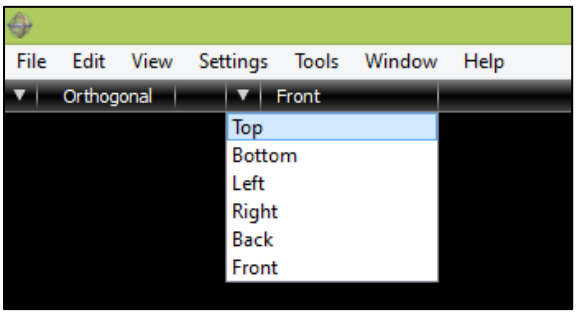


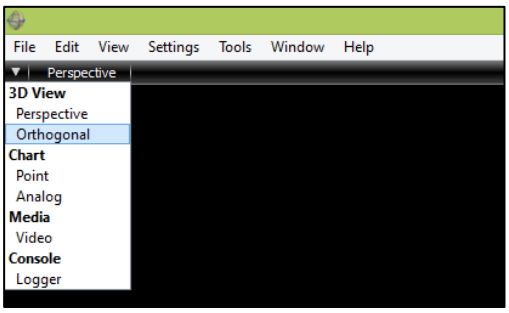

5) Don’t forget to save the modifications. It is recommended to save the *.mvc file in the **C3D Models** folder of the PLAViMoP application, so that it will be easier to load them later.




## 4.3. Set the point of view

You can switch between “orthogonal projection” and “perspective” view with the tool in the left-top corner.




The orthogonal projection mode is useful if you want to precisely set a side/front/top point of view.

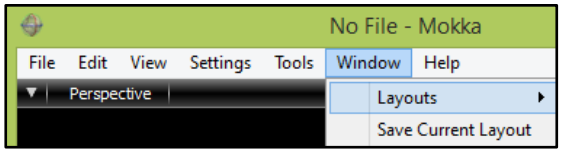
For both orthogonal projection and perspective modes:

- Use the mouse wheel to zoom in/out on the 3D scene. It works for both mode (hold alt gr then left click of the touchpad + move up/down).
- Click and hold the mouse wheel then move the mouse to grab the 3D scene (hold alt then left click of the touchpad + move).

For perspective mode only:

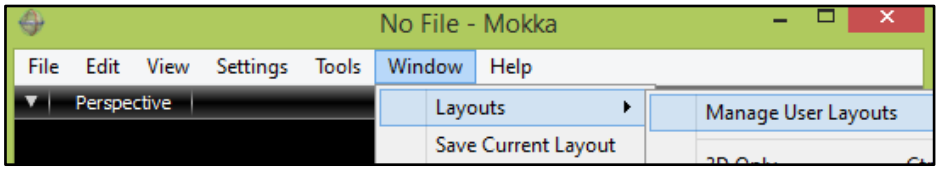
- Click and hold the left mouse button (or touchpad button) to rotate around the 3D scene.

The 3D view settings (view angle, zoom …) can be saved in a layout through the “Window\Save Current Layout” menu of Mokka application.




The layout can be re-loaded via the “Window\Layouts” menu.

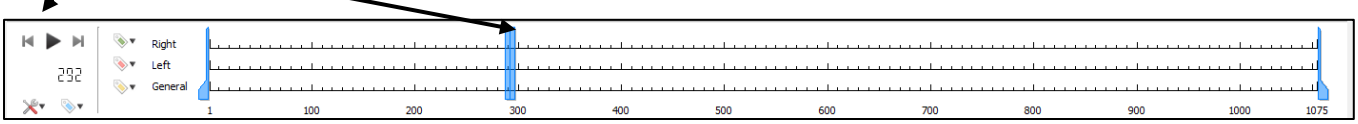
Note that if the 3D view doesn’t update, just press F5 on your keyboard.




## 4.4. Play the C3D file

To play the content of a C3D file:

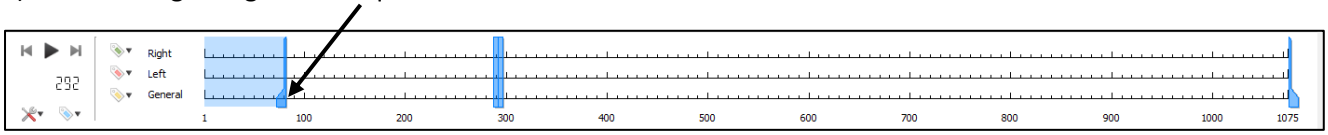
- Simply use the time bar buttons, or
- Move directly the cursor of the time bar.



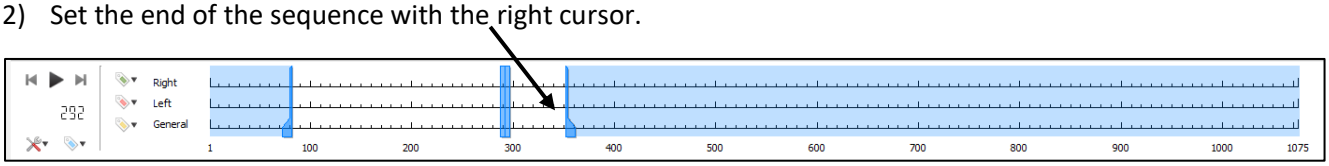

## 4.5. Crop the C3D file

If there is too much frames in the C3D file, it is possible to crop it as follow:

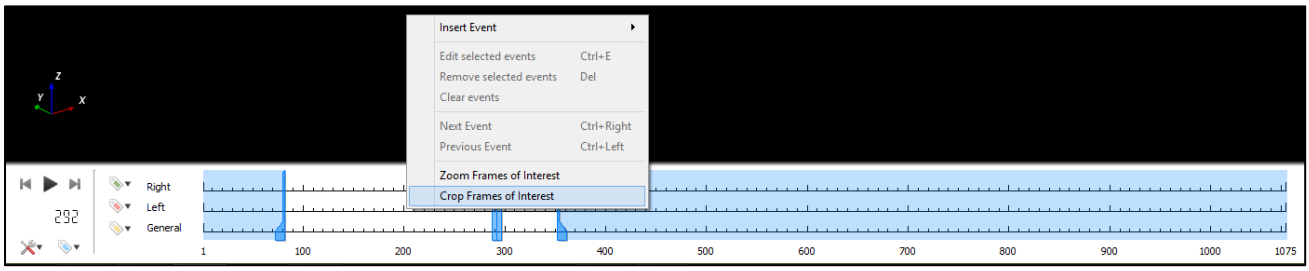
1) Set the beginning of the sequence with the left cursor.




2) Set the end of the sequence with the right cursor.




3) Right click on the time bar and in the floating menu, select **Crop Frames of Interest**.

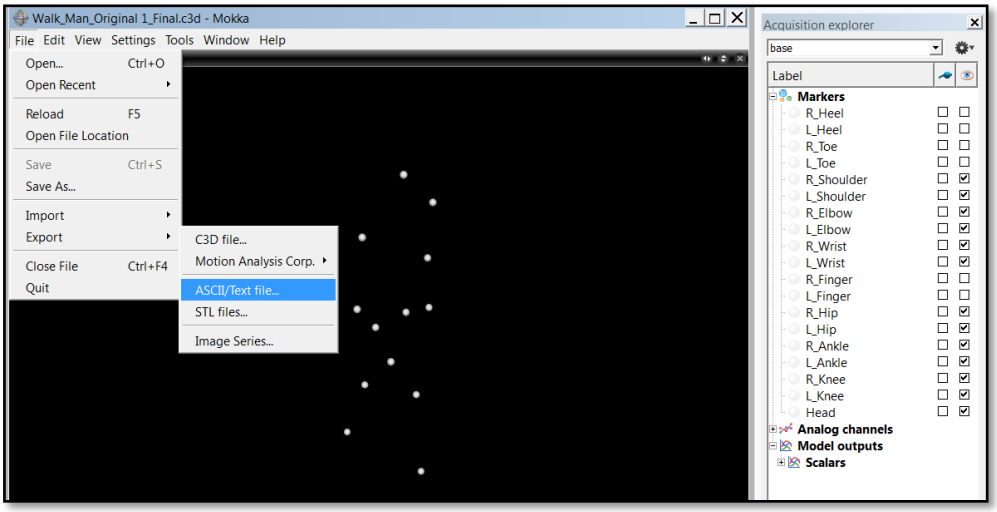



⁎ *This operation is a Mokka transformation. Use "File/Save as" menu of Mokka application to save this transformation and not the Matlab interface button as explained here.*

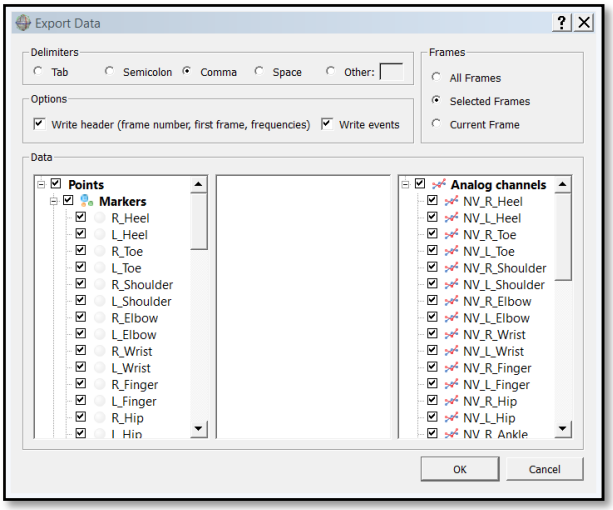
## 4.6. Save the coordinates of motion

Thanks to Mokka, it is possible to save in a text file the coordinates of each marker constituting the c3d. For this,

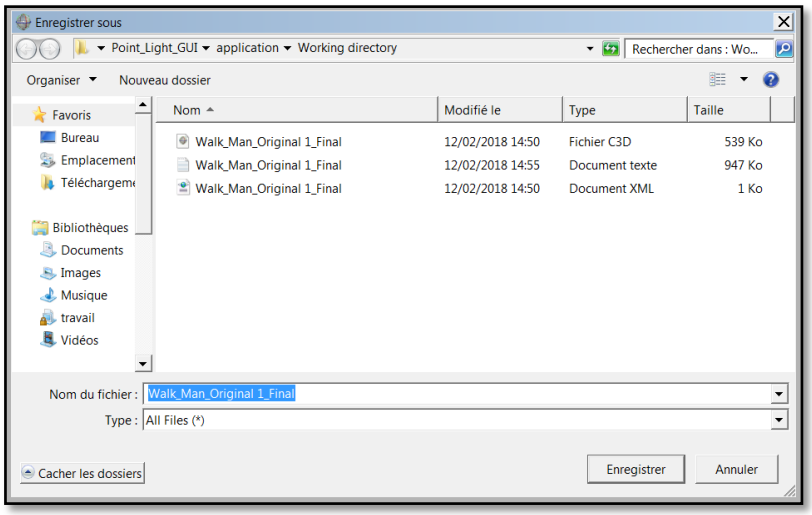
1) Go to file/Export Menu of Mokka
2) Select ASCII/Text file…



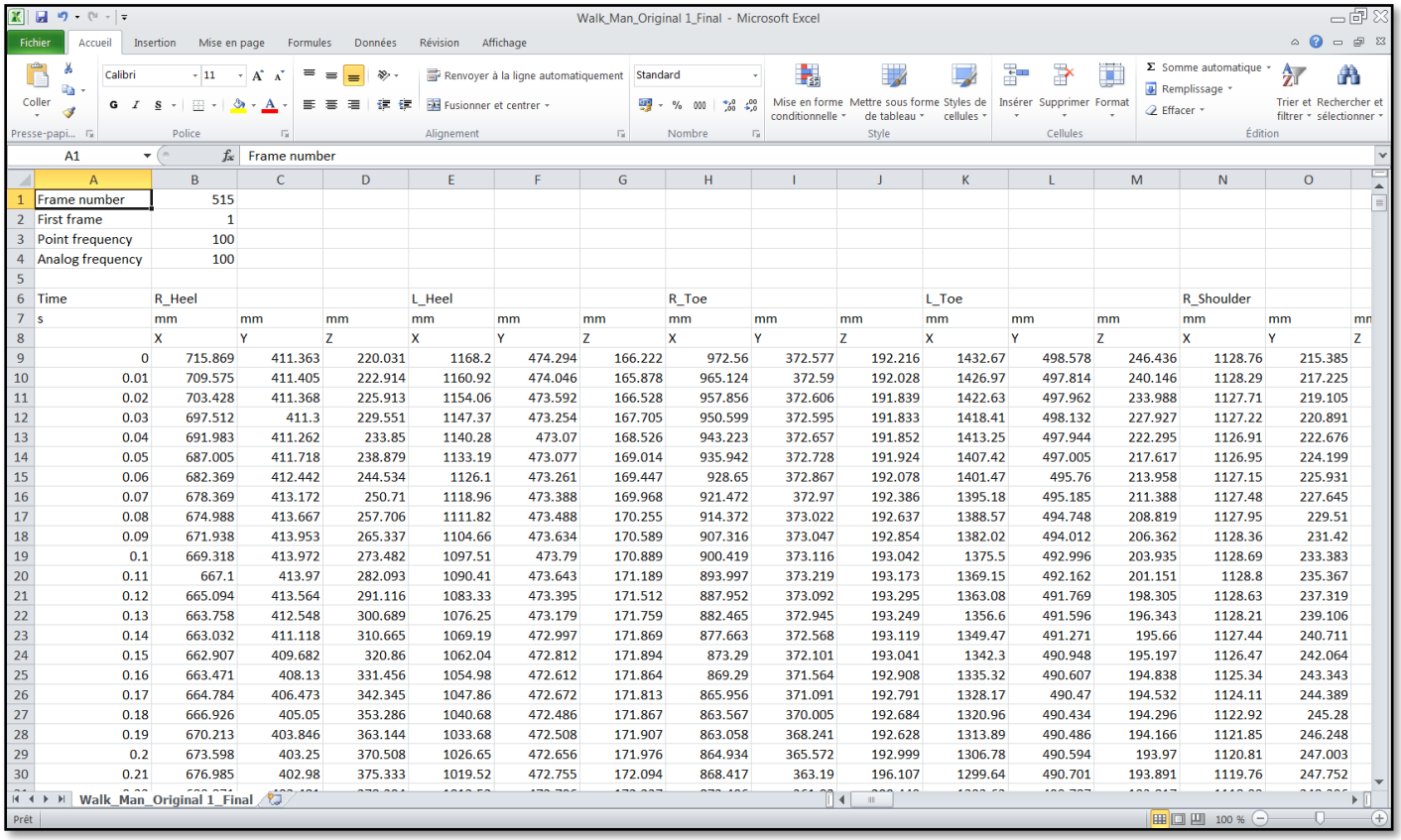

3) Select the characteristics of your exportation (delimiters, frames, options, markers, etc.). Don't forget to check the Analog channels box to export velocities and accelerations.




4) Choose a directory and a name to save your file




5) You can open your file with table program (as excel for example)




| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Frame number | 515 | | | | | | | | | | | | | |
| 2 | First frame | 1 | | | | | | | | | | | | | |
| 3 | Point frequency | 100 | | | | | | | | | | | | | |
| 4 | Analog frequency | 100 | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | |
| 6 | Time | R_Heel | | | L_Heel | | | R_Toe | | | L_Toe | | | R_Shoulder | |
| 7 | s | mm | mm | mm | mm | mm | mm | mm | mm | mm | mm | mm | mm | mm | mm |
| 8 | | X | Y | Z | X | Y | Z | X | Y | Z | X | Y | Z | X | Y |
| 9 | 0 | 715.869 | 411.363 | 220.031 | 1168.2 | 474.294 | 166.222 | 972.56 | 372.577 | 192.216 | 1432.67 | 498.578 | 246.436 | 1128.76 | 215.385 |
| 10 | 0.01 | 709.575 | 411.405 | 222.914 | 1160.92 | 474.046 | 165.878 | 965.124 | 372.59 | 192.028 | 1426.97 | 497.814 | 240.146 | 1128.29 | 217.225 |
| 11 | 0.02 | 703.428 | 411.368 | 225.913 | 1154.06 | 473.592 | 166.528 | 957.856 | 372.606 | 191.839 | 1422.63 | 497.962 | 233.988 | 1127.71 | 219.105 |
| 12 | 0.03 | 697.512 | 411.3 | 229.551 | 1147.37 | 473.254 | 167.705 | 950.599 | 372.595 | 191.833 | 1418.41 | 498.132 | 227.927 | 1127.22 | 220.891 |
| 13 | 0.04 | 691.983 | 411.262 | 233.85 | 1140.28 | 473.07 | 168.526 | 943.223 | 372.657 | 191.852 | 1413.25 | 497.944 | 222.295 | 1126.91 | 222.676 |
| 14 | 0.05 | 687.005 | 411.718 | 238.879 | 1133.19 | 473.077 | 169.014 | 935.942 | 372.728 | 191.924 | 1407.42 | 497.005 | 217.617 | 1126.95 | 224.199 |
| 15 | 0.06 | 682.369 | 412.442 | 244.534 | 1126.1 | 473.261 | 169.447 | 928.65 | 372.867 | 192.078 | 1401.47 | 495.76 | 213.958 | 1127.15 | 225.931 |
| 16 | 0.07 | 678.369 | 413.172 | 250.71 | 1118.96 | 473.388 | 169.968 | 921.472 | 372.97 | 192.386 | 1395.18 | 495.185 | 211.388 | 1127.48 | 227.645 |
| 17 | 0.08 | 674.988 | 413.667 | 257.706 | 1111.82 | 473.488 | 170.255 | 914.372 | 373.022 | 192.637 | 1388.57 | 494.748 | 208.819 | 1127.95 | 229.51 |
| 18 | 0.09 | 671.938 | 413.953 | 265.337 | 1104.66 | 473.634 | 170.589 | 907.316 | 373.047 | 192.854 | 1382.02 | 494.012 | 206.362 | 1128.36 | 231.42 |
| 19 | 0.1 | 669.318 | 413.972 | 273.482 | 1097.51 | 473.79 | 170.889 | 900.419 | 373.116 | 193.042 | 1375.5 | 492.996 | 203.935 | 1128.69 | 233.383 |
| 20 | 0.11 | 667.1 | 413.97 | 282.093 | 1090.41 | 473.643 | 171.189 | 893.997 | 373.219 | 193.173 | 1369.15 | 492.162 | 201.151 | 1128.8 | 235.367 |
| 21 | 0.12 | 665.094 | 413.564 | 291.116 | 1083.33 | 473.395 | 171.512 | 887.952 | 373.092 | 193.295 | 1363.08 | 491.769 | 198.305 | 1128.63 | 237.319 |
| 22 | 0.13 | 663.758 | 412.548 | 300.689 | 1076.25 | 473.179 | 171.759 | 882.465 | 372.945 | 193.249 | 1356.6 | 491.596 | 196.343 | 1128.21 | 239.106 |
| 23 | 0.14 | 663.032 | 411.118 | 310.665 | 1069.19 | 472.997 | 171.869 | 877.663 | 372.568 | 193.119 | 1349.47 | 491.271 | 195.66 | 1127.44 | 240.711 |
| 24 | 0.15 | 662.907 | 409.682 | 320.86 | 1062.04 | 472.812 | 171.894 | 873.29 | 372.101 | 193.041 | 1342.3 | 490.948 | 195.197 | 1126.47 | 242.064 |
| 25 | 0.16 | 663.471 | 408.13 | 331.456 | 1054.98 | 472.612 | 171.864 | 869.29 | 371.564 | 192.908 | 1335.32 | 490.607 | 194.838 | 1125.34 | 243.343 |
| 26 | 0.17 | 664.784 | 406.473 | 342.345 | 1047.86 | 472.672 | 171.813 | 865.956 | 371.091 | 192.791 | 1328.17 | 490.47 | 194.532 | 1124.11 | 244.389 |
| 27 | 0.18 | 666.926 | 405.05 | 353.286 | 1040.68 | 472.486 | 171.867 | 863.567 | 370.005 | 192.684 | 1320.96 | 490.434 | 194.296 | 1122.92 | 245.28 |
| 28 | 0.19 | 670.213 | 403.846 | 363.144 | 1033.68 | 472.508 | 171.907 | 863.058 | 368.241 | 192.628 | 1313.89 | 490.486 | 194.166 | 1121.85 | 246.248 |
| 29 | 0.2 | 673.598 | 403.25 | 370.508 | 1026.65 | 472.656 | 171.976 | 864.934 | 365.572 | 192.999 | 1306.78 | 490.594 | 193.97 | 1120.81 | 247.003 |
| 30 | 0.21 | 676.985 | 402.98 | 375.333 | 1019.52 | 472.755 | 172.094 | 868.417 | 363.19 | 196.107 | 1299.64 | 490.701 | 193.891 | 1119.76 | 247.752 |

Note that with this procedure you can access to the X, Y, Z coordinates of each marker and also to the velocity and acceleration for each axe and each marker.

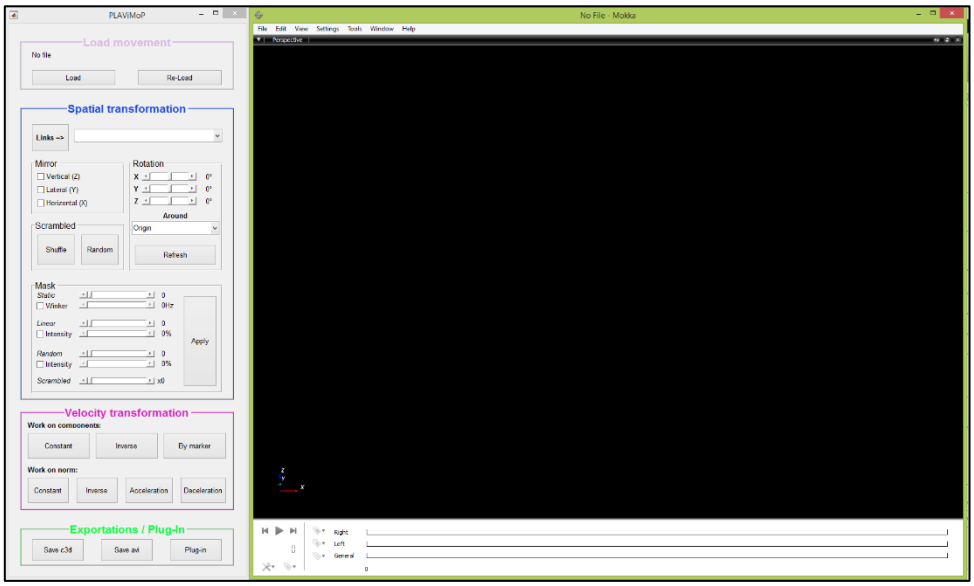
You can use this procedure for original and modified c3d files.

# 5. PLAViMoP functionalities

To start the application, double click on the PLAViMoP.exe file. It will launch the Matlab graphical user interface and Mokka application.

The user interface is divided in four zones:

- Load movement
- Spatial transformation
- Velocity transformation
- Exportations




## 5.1. Load a point-light sequence

To load a C3D file, click on the **Load** button. It will automatically open the C3D Files folder of the application. Even if the best practice is to put the C3D file in this folder, you can browse your disk to load a C3D file from another location. When a file is selected, the application loads it into Mokka with the Mokka Visualisation Configuration file (if specified). A copy of the original C3D file is automatically made in the Working directory folder of the application and all future transformations will be made on this copy so that to preserve the original file.

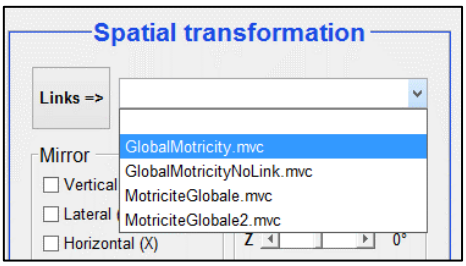
If you are “lost” in your transformation, you can easily reload the original C3D file with the **Re-Load** button.

Note #1: if another C3D file is loaded, all the content of the Working directory is deleted. So, be sure to save your previous work before (re)loading.

Note #2: to use your own c3d files or c3d files downloaded from PLAViMoP Database, put the files in the “C3D Files” directory created during the installation of PLAViMoP Software.

## 5.2. Load a configuration file

A C3D file can be loaded with a particular Mokka Visualisation Configuration file (*.mvc). For that, use the **popup menu** of the spatial transformation zone to choose the *.mvc file. The files listed here are the ones present in the C3D Models folder of the application.




If you have created a new *.mvc file since PLAViMoP have been started, you can refresh the list by clicking on the **Links** button.

## 5.3. Spatial transformation

This module enables to transform spatially the original motion. The different modifications are detailed below.

### *5.3.1. Mirror*

This transformation enables to create horizontal, lateral or vertical symmetry of the original motions.

To apply a mirror transformation to a point-light sequence, click the corresponding transformation on the interface. The result appears directly in the Mokka window.

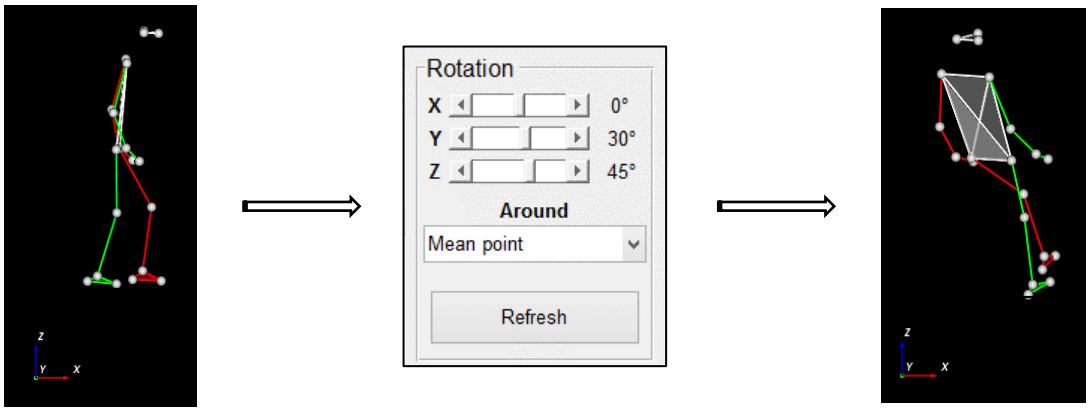
### *5.3.2. Rotation*

The rotation transformation enables to rotate about different axes (x, y, z) the original sequence of motion. The rotation point (origin, mean point or joints) and as well as the rotation angle (from –180° to 180°) can be specified by the user.

To apply this transformation, first choose the axes and angle of rotation and then set the point around which the rotation has to be made with the popup menu.

Finally, click on **Refresh** button to perform the following operation (note that the rotation sequence is X, Y, Z in this order).

For example, here is an illustration of a rotation of 30° around Y-axis followed by a rotation of 45° around Z-axis, rotations made about the mean point:




### *5.3.3. Scrambled*

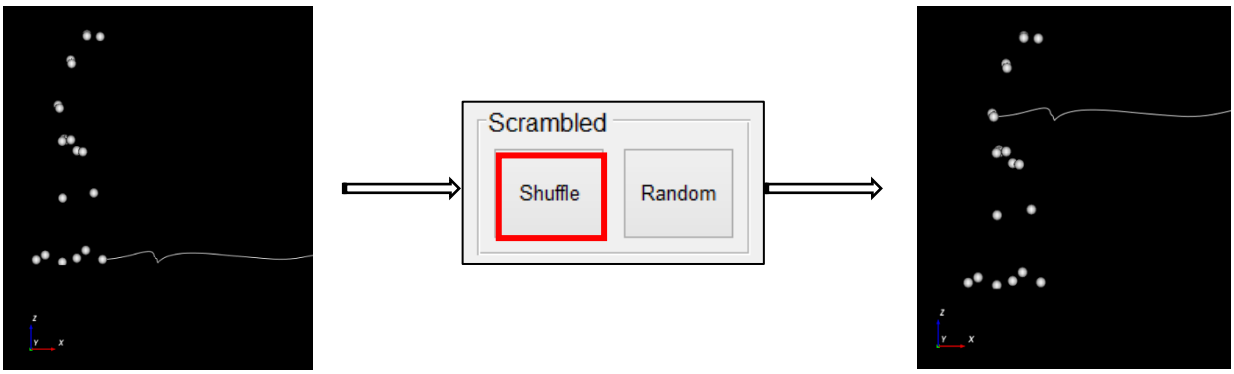
This transformation enables to scramble each point constituting the sequence.

There are two modes: **Shuffle** and **Random**

In **Shuffle** mode, each point takes the place of another but conserves its initial trajectory and dynamic.

To perform a shuffled scramble transformation, just click on the **Shuffle** button.

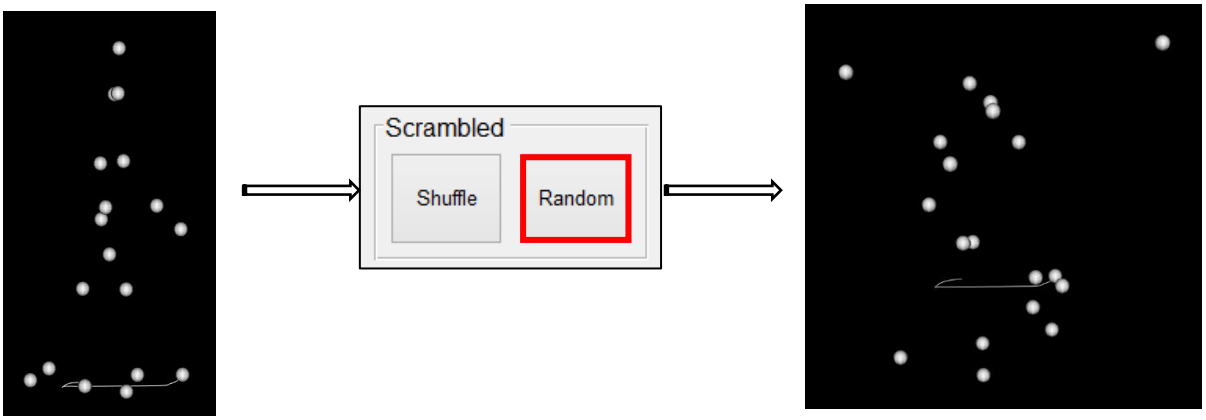
In the example presented below, the right elbow randomly inherited the trajectory and dynamic of the left foot marker.




In **Random** mode, each point takes a random place but conserves its initial trajectory and dynamic.

To perform a randomized scramble transformation, just click on the **Random** button.

In the example presented below, the left toes starting position moves randomly to another point inside the box defined to avoid the markers to leave the 3D scene volume during the movement.

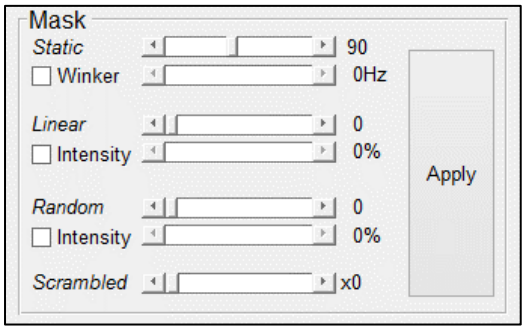



## 5.4. Masking PLDs

The masks are additional point lights that make more difficult the original movement recognition. There are four kinds of masks:

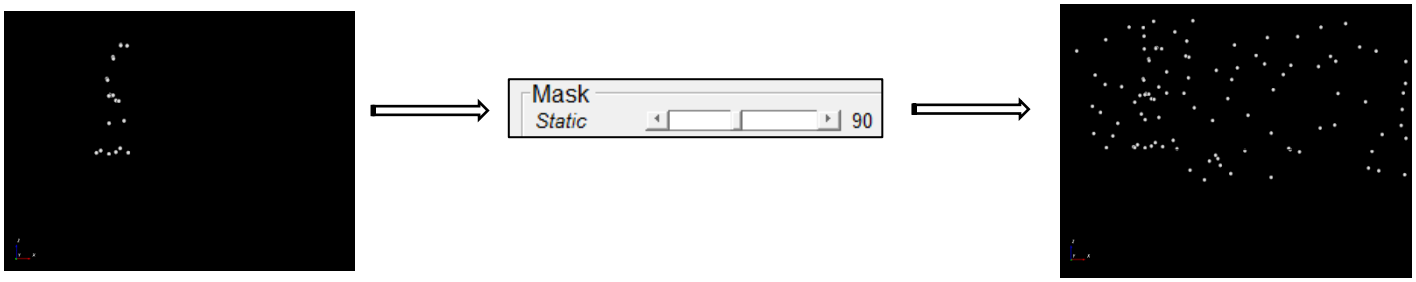
- Static masks
- Linear masks
- Random masks
- Scrambled masks

The first three have two different behaviours detailed below.

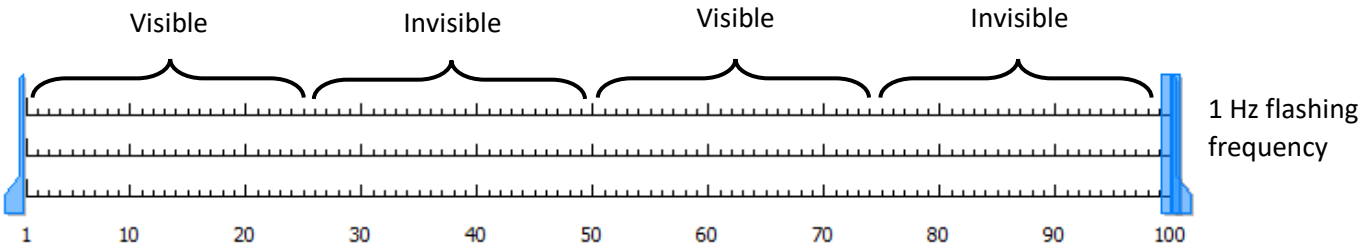



#### *5.4.1.1. Static masks*

The static mask is simply a not moving point light whose coordinates (randomly defined) lay in the limits of the bounding box. Up to 200 static masks can be added using the dedicated slider. Click then on **Apply** button to update the C3D file.

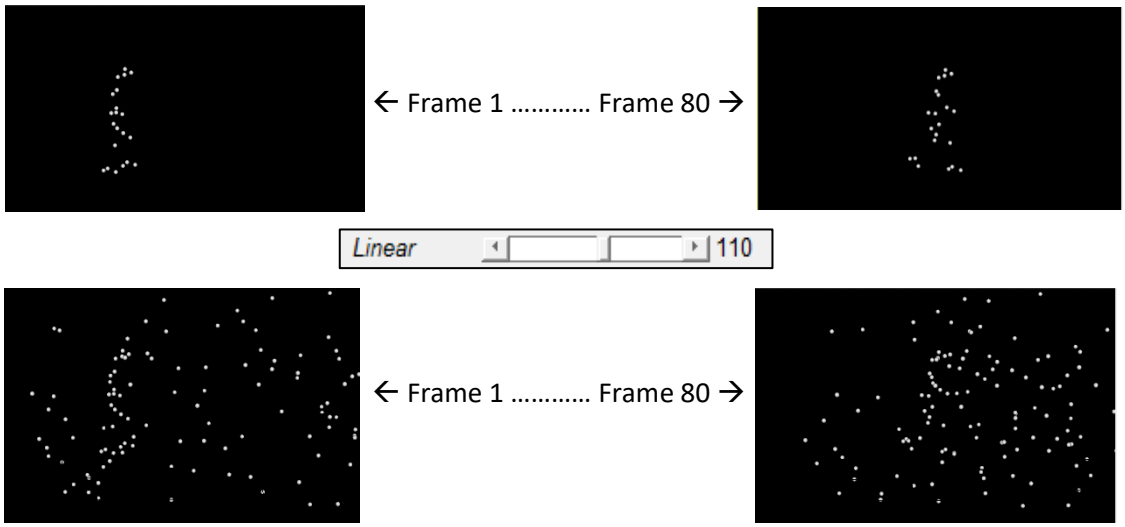


The second behaviour occurs if the **Winker** button is checked. The associated **slider** becomes enable. Use it to specify the flashing frequency (from 1 to 25 Hz). As the C3D file frame rate is 100 Hz, a flashing frequency of 1 Hz causes a mask being visible/invisible alternatively during 25 consecutive frames while a flashing frequency of 25 Hz causes a mask being visible/invisible alternatively during 2 consecutive frames.




#### *5.4.1.2. Linear masks*

The linear mask is a moving point light with constant velocity. Linear masks only move along X-axis (in positive or negative direction). Their initial positions are randomly chosen. According to their initial positions and the duration of the C3D file, a maximal velocity is computed in order to keep the masks in the limits of the bounding box. Then a random percentage of this velocity is chosen to compute the trajectory. Up to 200 static masks can be added using the dedicated slider. Click then on **Apply** button to update the C3D file.




The second behaviours occurs if the **Intensity** button is checked. The associated **slider** becomes enable. Use it to specify the common percentage of maximal velocity assigned to each mask (from 0 to 100 %). As there are two possible directions for the displacement of linear masks, masks can be divided into two groups. All the markers of the same group will have the same velocity. A 0 % intensity causes static masks, while a 100 % intensity ensures that all markers of the group stay in the limits of the bounding box.

#### 5.4.1.3. Random masks

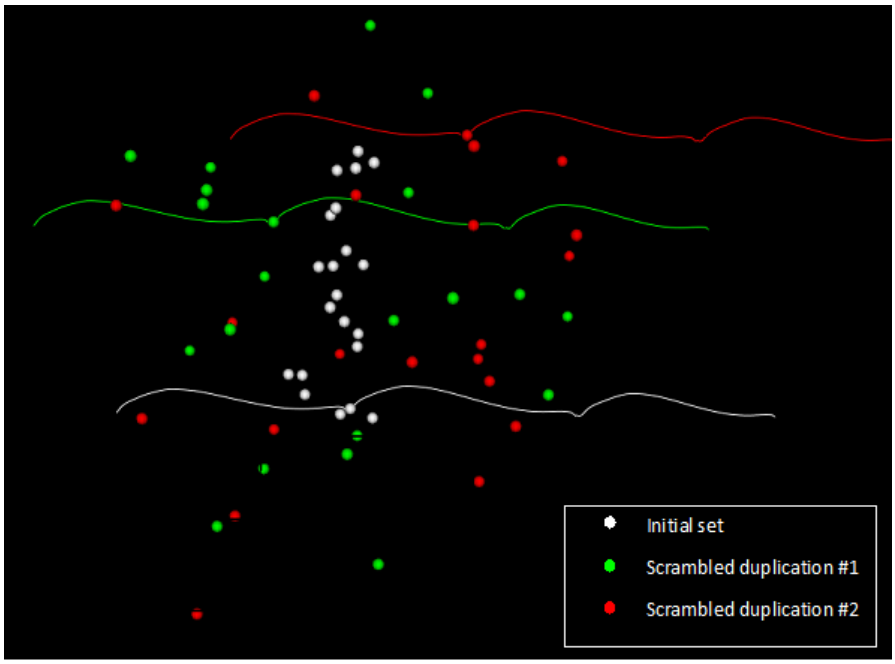
The random mask is a point light with a randomly defined trajectory. Both initial position and instantaneous acceleration are randomly chosen. Masks move along the three axes. A control loop ensures that the mask stays in the bounding box limits (rebound).

The second behaviour occurs if the **Intensity** button is checked. The associated **slider** becomes enable. Use it to specify the common percentage of maximal velocity (arbitrary fixed to 10 m/s) assigned to each mask (from 0 to 100 %).

#### 5.4.1.4. Scrambled masks

The scrambled mask is a set of point light with the same trajectory of initial point light set. Only their starting positions are defined randomly. A control loop ensures that the mask stay in the bounding box limits. The number of scrambled mask (k) is proportional with the number of point light (n) in the initial set.

Note that k is limited by the relation: k x n <200.




Please see [2] for the use of scrambled masks.

## 5.5. Velocity transformation

This series of tools aims to modify the dynamic of point light displacement. There are two different type of transformation:

- Modifications based on changes in the norm of the velocity: in this case, the norm of the velocity of a point light is modified in order to keep the original point light path (4 kinds of modification).
- Modifications based on changes in the components of the velocity: in this case, both norm, components and path are modified (3 kinds of modification).

### 5.5.1. Transformations applied to the norm of the velocity

The norm of the velocity of a given point light is classically computed at each frame with:

$$\|V\| = \sqrt{V_X^2 + V_Y^2 + V_Z^2}$$

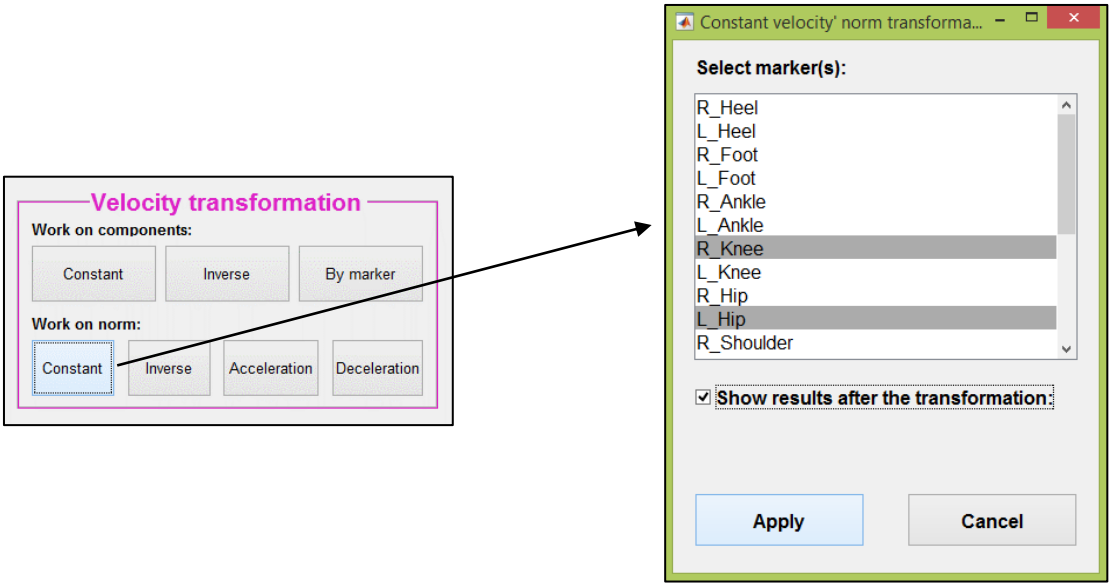
All transformations detailed below enable to modify the dynamic of the original sequence but in keeping the original trajectory and movement duration.

#### *5.5.1.1. Constant norm*

For this transformation, the components of a given point light velocity are modified in order to:

1) Keep the original point light path
2) Keep the original movement duration
3) Keep a constant norm of the given point light velocity throughout the movement

In PLAViMoP, to process this kind of transformation, just click on the **Constant** button of the "Work on norm" section to open the point light selection window.

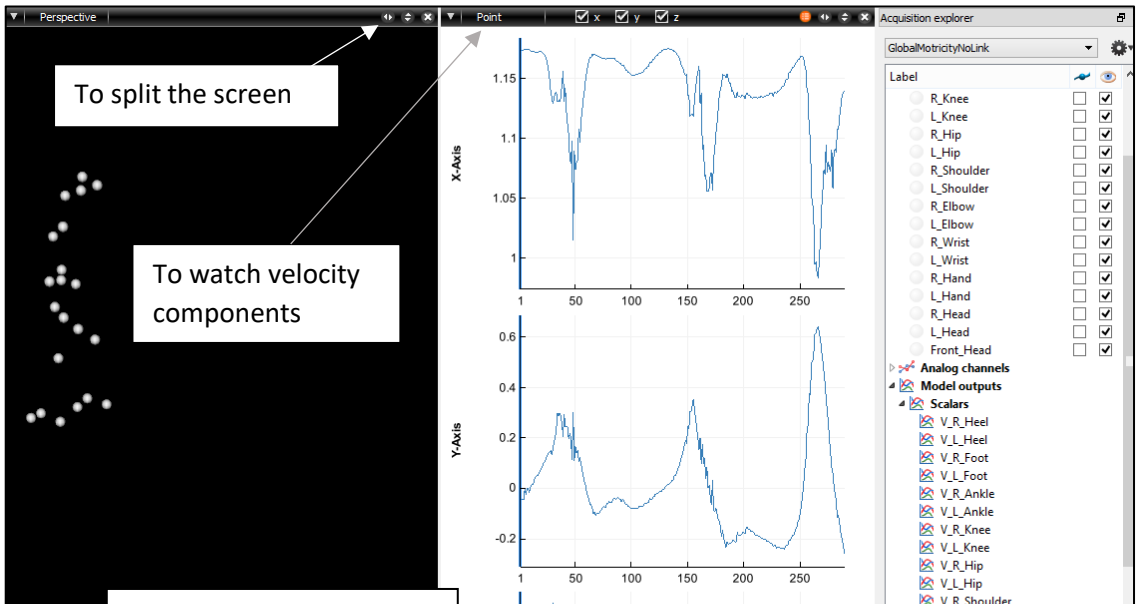



You can select several markers at the same time by holding Ctrl key.

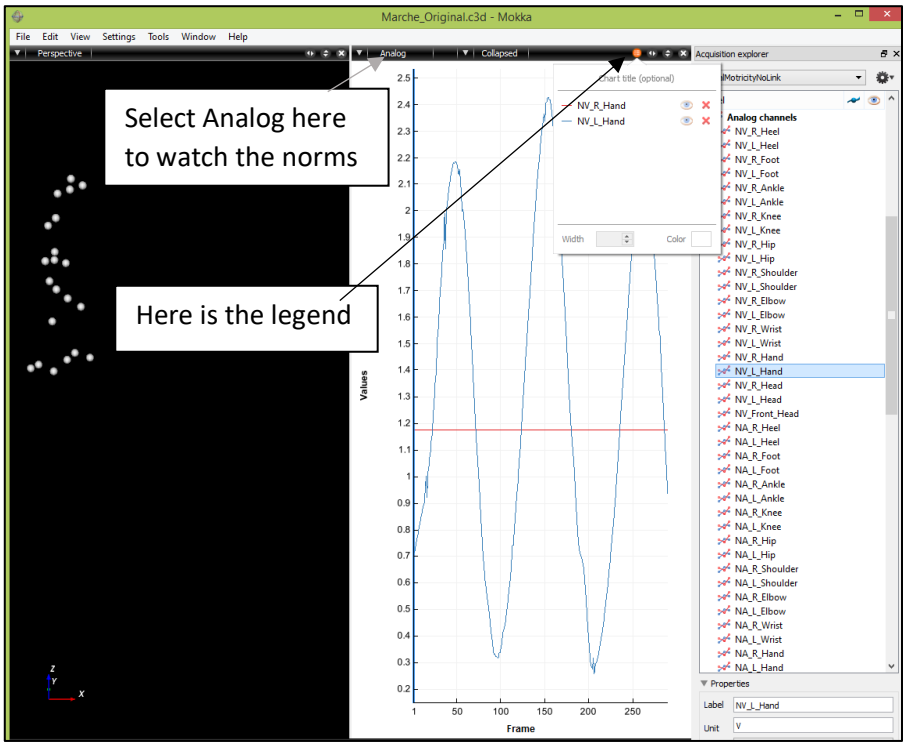
If you want to see the detailed process (figures presented in Annexes), check the box "Show results after transformation".

By clicking on **Apply** button, the computations are made and the C3D file is updated.

The velocity (V) and acceleration (A) of each point light are added in C3D file under the **Scalars** section of Mokka.



The norm of point light velocity and acceleration are also listed in the **Analog channels** section of Mokka.




### *5.5.1.2. Inverse norm*

For this transformation, the components of a given point light velocity are modified in order to:

1) Keep the original point light path
2) Keep the original movement duration
3) Get a norm of the given point light velocity inverted with respect to the mean norm original velocity.

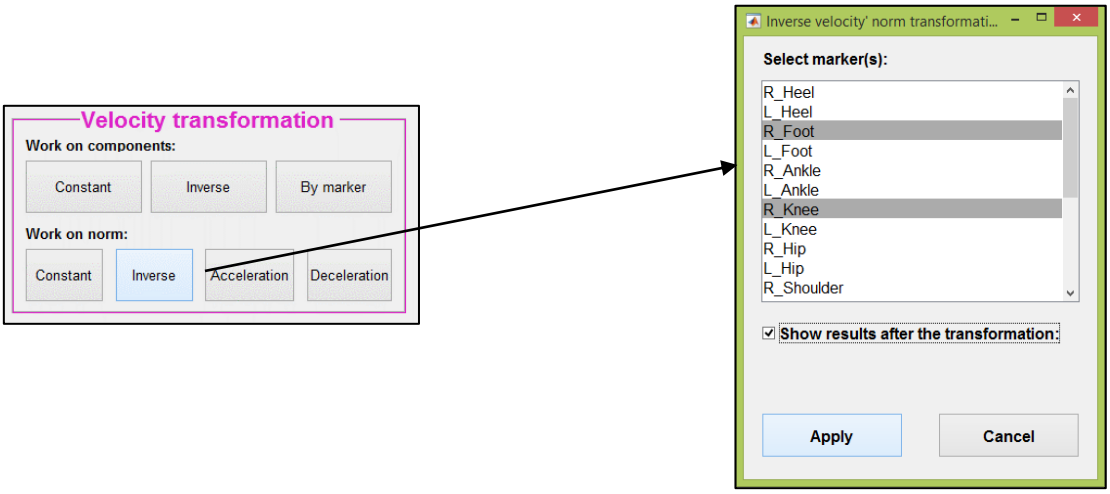
In PLAViMoP, to process this kind of transformation, just click on the **Inverse** button of the “Work on norm” section to open the point light selection window.

You can select several markers at the same time by holding Ctrl key.

If you want to see the detailed process (figures presented in annexes), check the box “Show results after transformation”.

By clicking on **Apply** button, the computations are made and the C3D file is updated.

The velocity (V) and acceleration (A) of each point light are added in C3D file under the **Scalars** section of Mokka (see here for more details).




### *5.5.1.3. Accelerated norm*

For this transformation, the components of a given point light velocity are modified in order to:

1) Keep the original point light path
2) Keep the original movement duration
3) Get an uniformly accelerated motion

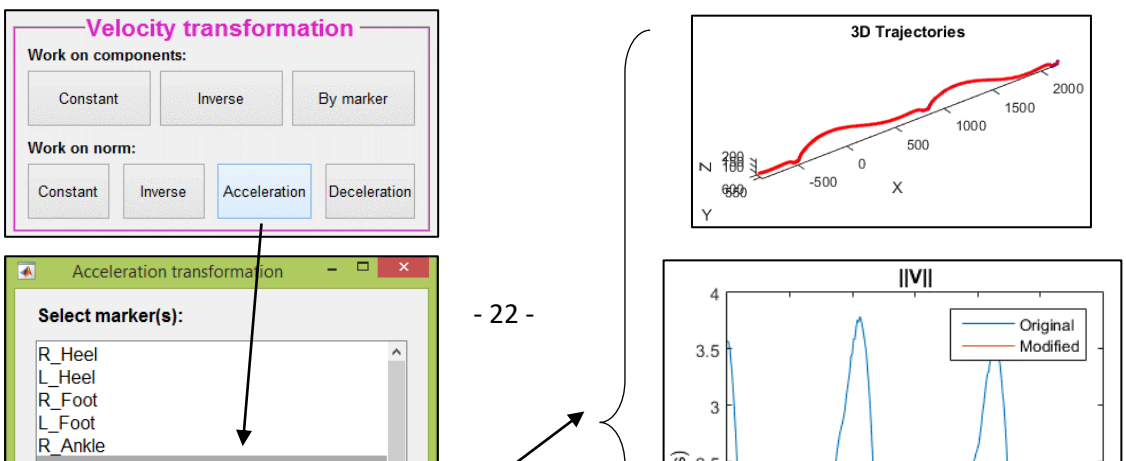
In PLAViMoP, to process this kind of transformation, just click on the **Acceleration** button of the “Work on norm” section to open the point light selection window.

You can select several markers at the same time by holding Ctrl key.

If you want to see the detailed process, check the box “Show results after transformation”.

By clicking on **Apply** button, the computations are made and the C3D file is updated.

The velocity (V) and acceleration (A) of each point light are added in C3D file under the **Scalars** section of Mokka (see here for more details).



#### 5.5.1.4. Decelerated norm

For this transformation, the components of a given point light velocity are modified in order to:

1) Keep the original point light path
2) Keep the original movement duration
3) Get an uniformly decelerated motion

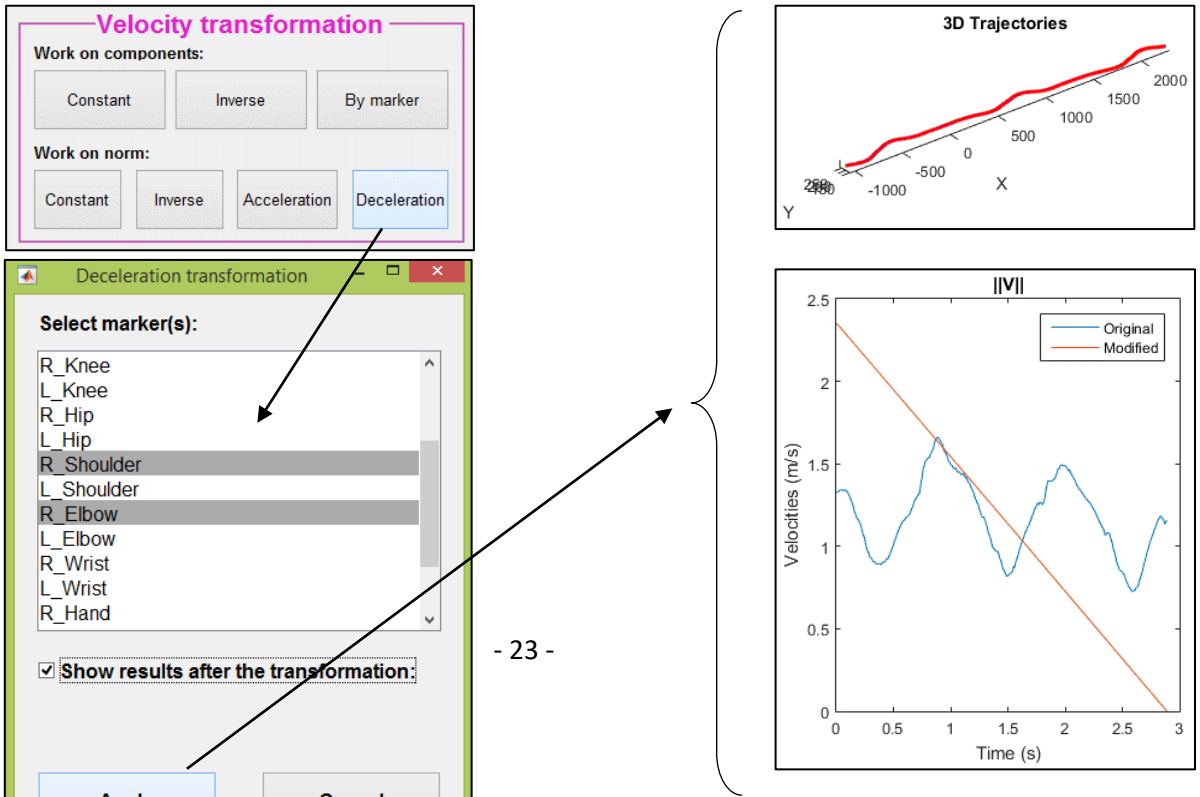
In PLAViMoP, to process this kind of transformation, just click on the **Deceleration** button of the "Work on norm" section to open the point light selection window.

You can select several markers at the same time by holding Ctrl key.

If you want to see the detailed process, check the box "Show results after transformation".

By clicking on **Apply** button, the computations are made and the C3D file is updated.

The velocity (V) and acceleration (A) of each point light are added in C3D file under the **Scalars** section of Mokka (see here for more details).



Please see [3] and [4] for illustrations of these transformations.

### 5.5.2. Transformations applied to each component of the velocity

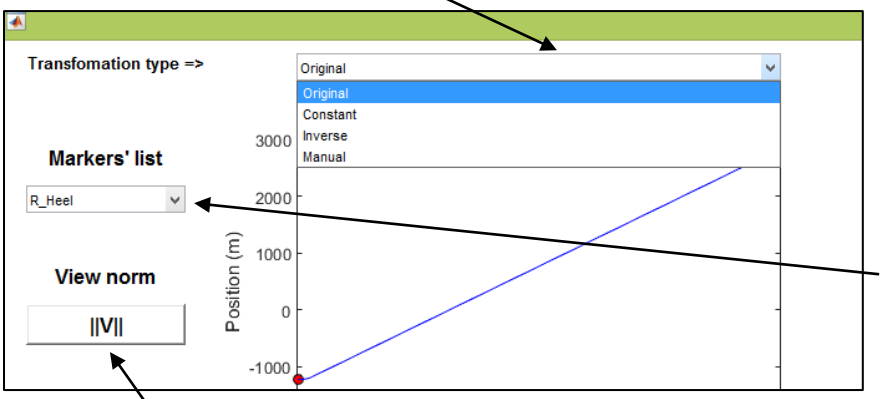
The main difference between this series of transformations and the previous ones is that the initial trajectory of point light is not maintained after the transformation. The modifications are applied to one, two or three components of the velocity for a given point light and it results in a new velocity norm and a new path for the point light.

#### *5.5.2.1. Manual setting of velocity components*

By clicking on **By marker** button of PLAViMoP application, you can access to three kind of velocity components transformation: constant, inverse and manual. The transformations can be applied components by components and point light by point light. Once a transformation is chosen for a point light and a velocity component, the new acceleration and coordinates are automatically computed.

With the **manual transformation**, it is possible to redefine the shape of the velocity component curve. For C3D file of more than 20 frames, 19 movable points (green circles) are added to the velocity curve. Just left click, hold and vertically drag the circle to move the checkpoint. When the left button is released, the velocity and acceleration components as well as the point light coordinates are re-computed. A shape-preserving piecewise cubic interpolation is performed between the clicked circle and the previous green (or red if any) circle, and between the clicked circle and the next green (or red if any) circle.

Use the popup menu on top of column of graphs to switch between original, constant, inverse and manual transformation modes.




Use the popup menu of markers' list to select the point light whose velocity has to be modified.

At any moment, it is possible to have a look at the effects of the transformations on the norm of the velocity by clicking on the push button **View norm**.

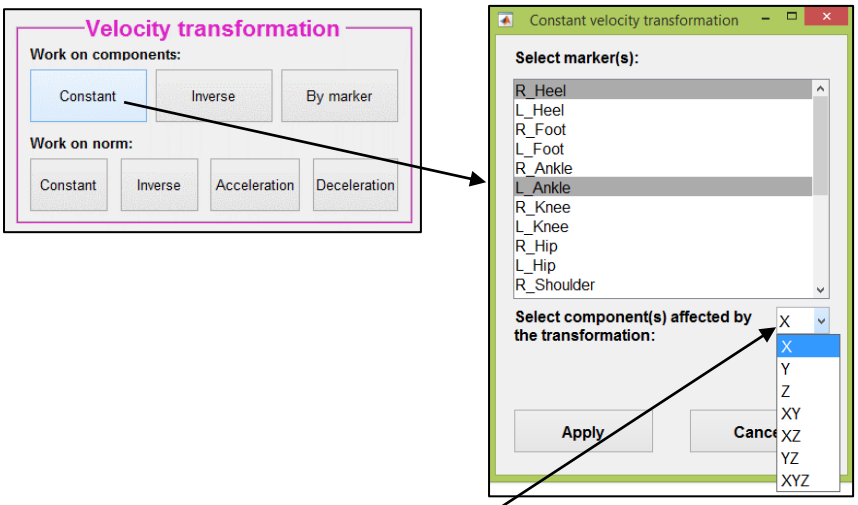
All the transformations are memorized and are definitively applied to the C3D file when closing the window. So it is not necessary to close the window after each point light transformation.

As for the transformations applied to the norm of the velocity, when the C3D file is updated, the new velocity and acceleration components and norms are written and are available as described here.

#### *5.5.2.2. Apply the same transformation to a group of point lights*

##### *5.5.2.2.1. Constant velocity components*

It is possible to apply the same constant velocity component transformation (as described here) to a set of point light by clicking on the **Constant** button of the "work on component" section.



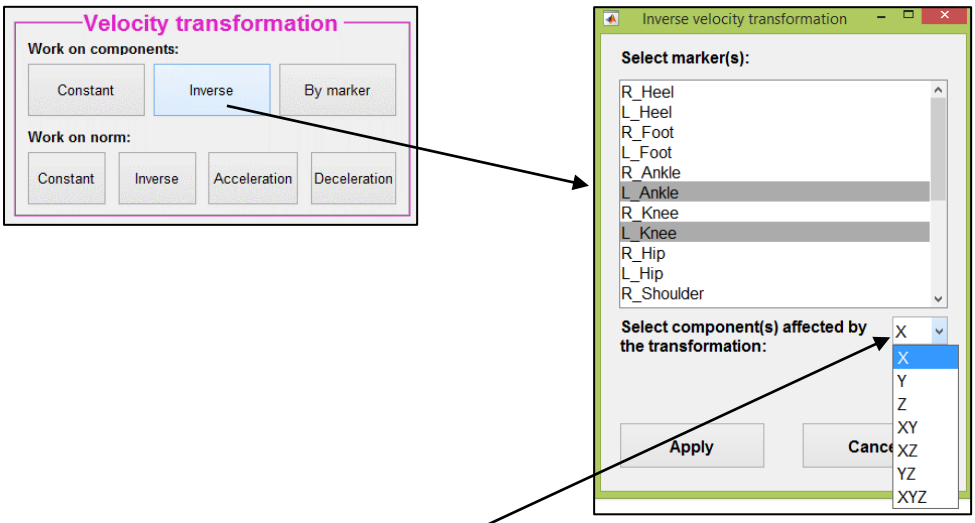

In the opening window, use the popup menu to select the component(s) that will be affected by the transformation.

You can select several markers at the same time by holding Ctrl key.

Click on **Apply** button to finalize the transformations. The C3D file will be updated, the new velocity and acceleration components and norms will be written and available as described here.

##### *5.5.2.2.2. Inverse velocity components*

It is possible to apply the same inverse velocity component transformation (as described here) to a set of point light by clicking on the **Inverse** button of the "work on component" section.




In the opening window, use the popup menu to select the component(s) that will be affected by the transformation.

You can select several markers at the same by holding Ctrl key.

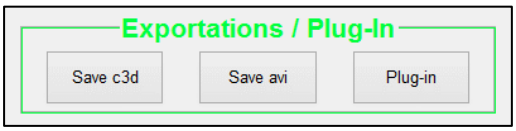
Click on **Apply** button to finalize the transformations. The C3D file will be updated, the new velocity and acceleration components and norms will be written and available as described here.

## 5.6. History transformation file

When loading a C3D file, a transformation history file (xml format) is created in the Working directory of the application. After each transformation made with the Matlab application, this file is updated. It allows to have a monitoring of the transformations processes. Here is an example of *.xml file illustrating the syntax of each kind of transformation.

## 5.7. Exportations / Plug-in

This section contains tools to export your work and to use your own transformations routines (plugin).




### 5.7.1. Save as C3D

There are two ways for saving your work as a C3D file depending on the application you use to make the transformations:

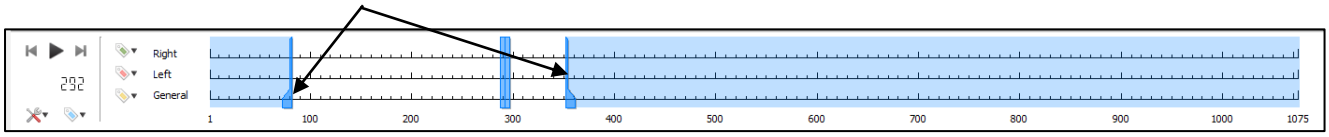
- If the transformations are made with the Matlab interface tool, you have to use the **Save c3d** button of the “Exportations” section. The history transformation file (*.xml) will be automatically saved in the same directory and with the same name as you give to the C3D file.

- If the transformations are made with Mokka, use the File/Save as menu of Mokka application. It is typically the case when cropping the C3D file.

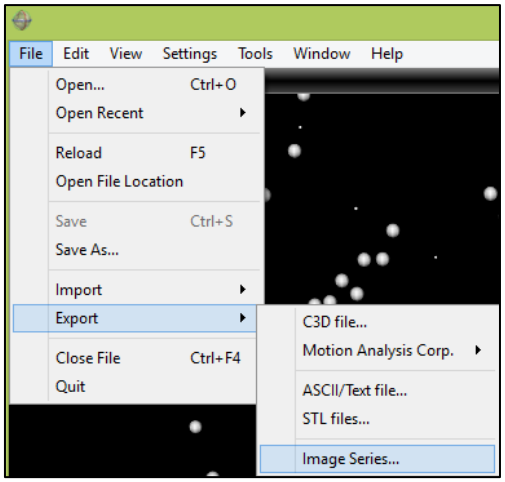
### 5.7.2. Save as AVI

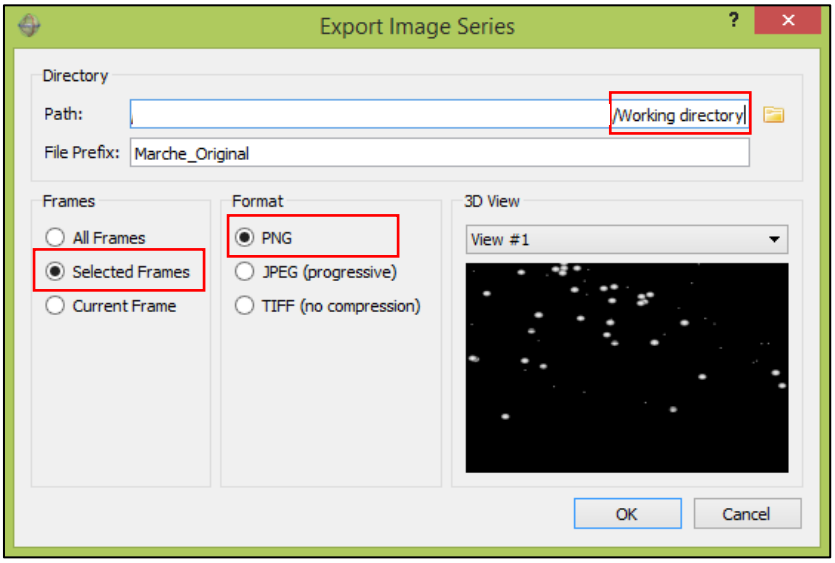
You can create a video (*.avi format) by following these 3 steps:

1) Set the beginning and the end of the sequence to export using left and right cropping cursor of the time bar in Mokka application (Note that it is not necessary to crop the sequence).




2) Create an image series via File \ Export \ Image series … menu of Mokka.

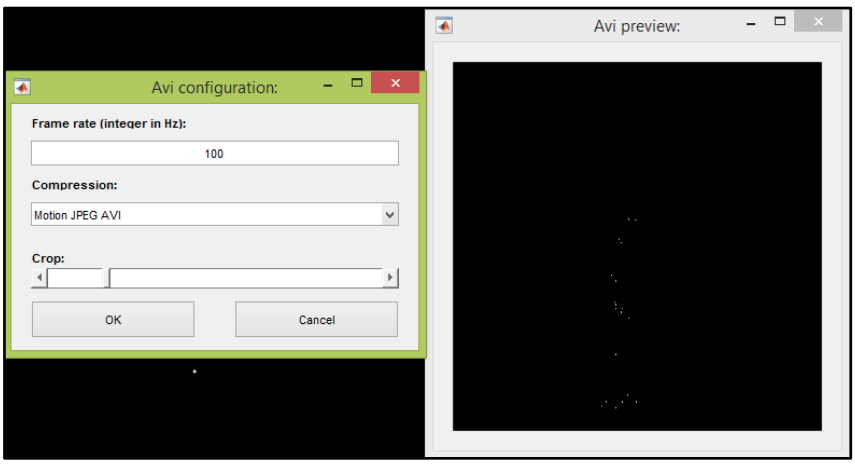



Check that Selected Frames and PNG are selected and that the path is pointing to the Working directory of the PLAViMoP Software and then click **OK**.

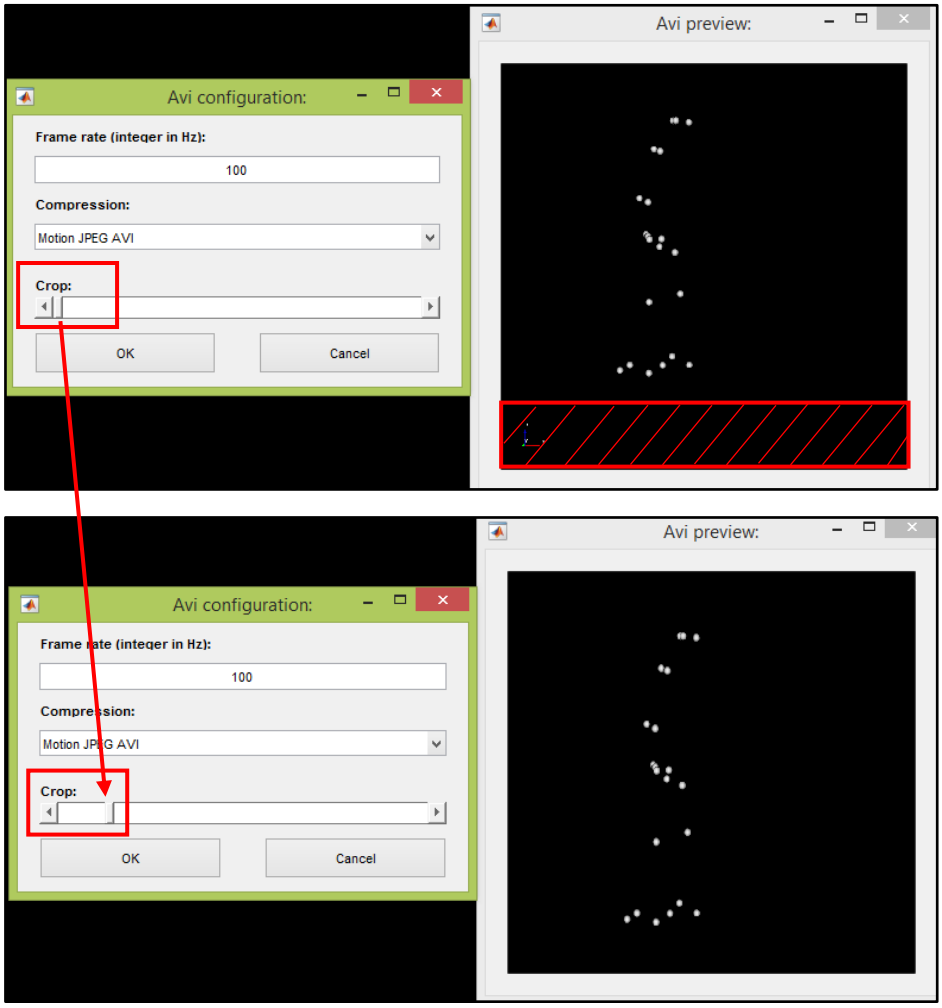
3) Then click on **Save avi** button of PLAViMoP Software, give a name to the video and click on **Save**.

4) Select the frame rate (default is the original C3D frame rate) and specify if the video needs to be compressed or not.



Note that an inferior frame rate than the original ones induces a slow motion effect

5) If you want to hide the global reference frame displayed in the bottom left corner of Mokka 3D view, you can use the “crop” slider to remove the bottom line(s) of pixels on each frame during the video creation.

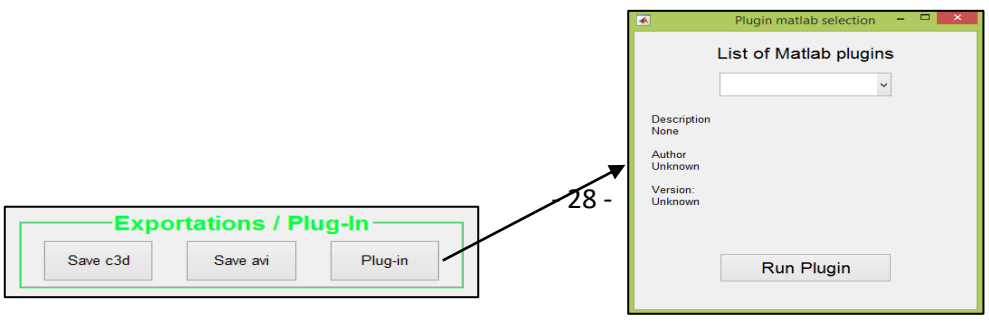



6) Click **OK** button to finalize the video creation. The image series will automatically remove to preserve disk space.

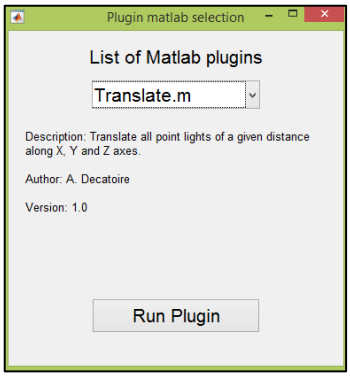
## 5.7.3. Use Plug-in

### *5.7.3.1. General considerations*

The **Plug-in** button is a tool that allows to extent the functionalities of PLAViMoP. In fact, your own Matlab script can be added to the Plugins Matlab folder of the application, and then it can be executed to perform the transformation you develop. When clicking on **Plug-in** button, a floating window opens. The available plug-ins (stored in the Plugins Matlab folder of the application) are listed in the popup menu.



The selection of a plug-in updates the description (if provided) of the plug-in, whose format is detailed here.




Finally, click on **Run Plugin** button to execute the plug-in. The available operations for a plugin are:

- Modifications of coordinates of existing point lights
- Modifications of velocity components and norm of existing point lights
- Modifications of acceleration components and norm of existing point lights
- Adding new point-lights…

At the end of the execution of the plug-in, the C3D is updated with regard to the contents of the Outputs variable.

#### *5.7.3.2. Developers considerations*

A complete plug-in is composed of a valid, commented Matlab script (*.m file) and a description file (*.txt). The script and the associated text file should have the save name (ex: Translate.m and Translate.txt).

The description file contents is organized in three parts:

- Description of the functionalities of the plug-in
- Author name (optionally with contact information)
- Version of the plug-in.

Here is an example of a valid description file.

# 6. References


[1] *Barré A, Armand S.* Biomechanical ToolKit: Open-source framework to visualize and process biomechanical data. Comput Methods Programs Biomedecine 2014; 80–87

[2] *Bidet-Ildei C, Chauvin A, Coello Y.* Observing or producing a motor action improves later perception of biological motion: Evidence for a gender effect. Acta Psychol (Amst) 2010; 134: 215–224

[3] *Bidet-Ildei C, Méary D, Orliaguet J-P.* Visual preference for isochronic movement does not necessarily emerge from movement kinematics: A challenge for the motor simulation theory. Neurosci Lett 2008; 430: 236–240

[4] *Martel L, Bidet-Ildei C, Coello Y.* Anticipating the terminal position of an observed action: Effect of kinematic, structural, and identity information. Vis Cogn 2011; 19: 785–798

[5] *Weinhandl JT, O'Connor KM.* Assessment of a greater trochanter-based method of locating the hip joint center. J Biomech 2010; 43: 2633–2636

# 7. Annexes

## 7.1. History transformation file

```
<?xml version="1.0" encoding="utf-8"?>
<Transformations_history Point_Light="Version 1.3">
  <Original_File_Informations>
    <Name>C:\~\Working directory\Marche_Original.c3d</Name>
    <Rate>100 Hz</Rate>
    <Number_of_Markers>21</Number_of_Markers>
    <Number_of_Frames>290</Number_of_Frames>
  </Original_File_Informations>
  <Spatial_transformation>
    <Model>GlobalMotricityNoLink.mvc</Model>
  </Spatial_transformation>
  <Spatial_transformation>
    <Mirror_Z>1</Mirror_Z>
  </Spatial_transformation>
  <Spatial_transformation>
    <Rotation>
      <Units>Degrees</Units>
      <About>R_Heel</About>
      <Rx>0</Rx>
      <Ry>40</Ry>
      <Rz>0</Rz>
    </Rotation>
  </Spatial_transformation>
  <Spatial_transformation>
    <Scrambled-Shuffle>
      <Scrambled-1>L_Ankle became R_Heel</Scrambled-1>
      <Scrambled-2>R_Foot became L_Heel</Scrambled-2>
      <Scrambled-3>L_Wrist became R_Foot</Scrambled-3>
      <Scrambled-4>R_Shoulder became L_Foot</Scrambled-4>
      <Scrambled-5>R_Knee became R_Ankle</Scrambled-5>
      <Scrambled-6>R_Hand became L_Ankle</Scrambled-6>
      <Scrambled-7>L_Elbow became R_Knee</Scrambled-7>
      <Scrambled-8>L_Knee became L_Knee</Scrambled-8>
      <Scrambled-9>R_Ankle became R_Hip</Scrambled-9>
      <Scrambled-10>Front_Head became L_Hip</Scrambled-10>
      <Scrambled-11>R_Head became R_Shoulder</Scrambled-11>
      <Scrambled-12>R_Wrist became L_Shoulder</Scrambled-12>
      <Scrambled-13>R_Heel became R_Elbow</Scrambled-13>
      <Scrambled-14>L_Heel became L_Elbow</Scrambled-14>
      <Scrambled-15>L_Foot became R_Wrist</Scrambled-15>
      <Scrambled-16>L_Hand became L_Wrist</Scrambled-16>
      <Scrambled-17>R_Elbow became R_Hand</Scrambled-17>
      <Scrambled-18>R_Hip became L_Hand</Scrambled-18>
      <Scrambled-19>L_Head became R_Head</Scrambled-19>
      <Scrambled-20>L_Hip became L_Head</Scrambled-20>
      <Scrambled-21>L_Shoulder became Front_Head</Scrambled-21>
    </Scrambled-Shuffle>
  </Spatial_transformation>
  <Spatial_transformation>
    <Winkers_masks>
      <Quantity>50</Quantity>
      <Rate>11</Rate>
    </Winkers_masks>
  </Spatial_transformation>
  <Spatial_transformation>
    <Linear_masks>
      <Quantity>60</Quantity>
      <Intensity>Random</Intensity>
    </Linear_masks>
  </Spatial_transformation>
  <Spatial_transformation>
    <Random_masks>
      <Quantity>40</Quantity>
      <Intensity>Random</Intensity>
    </Random_masks>
  </Spatial_transformation>
  <Spatial_transformation>
    <Scrambled_masks>
      <Quantity>x2</Quantity>
    </Scrambled_masks>
  </Spatial_transformation>
  <Velocity_transformation>
    <Components_Constant>
      <Component>X</Component>
      <Component>Z</Component>
      <Marker-1>R_Heel</Marker-1>
    </Components_Constant>
  </Velocity_transformation>
  <Velocity_transformation>
    <Components_Inverse>
      <Component>Y</Component>
      <Marker-1>R_Foot</Marker-1>
    </Components_Inverse>
  </Velocity_transformation>
  <Velocity_transformation>
    <By_marker>
      <Marker>R_Knee</Marker>
      <Component>X</Component>
      <Type>Constant</Type>
    </By_marker>
  </Velocity_transformation>
  <Velocity_transformation>
    <By_marker>
      <Marker>R_Knee</Marker>
      <Component>Y</Component>
      <Type>Inverse</Type>
    </By_marker>
  </Velocity_transformation>
  <Velocity_transformation>
    <By_marker>
      <Marker>R_Knee</Marker>
      <Component>Z</Component>
      <Type>Manual</Type>
    </By_marker>
  </Velocity_transformation>
  <Velocity_transformation>
    <Norm_Constant>
      <Marker-1>R_Hip</Marker-1>
    </Norm_Constant>
  </Velocity_transformation>
  <Velocity_transformation>
    <Norm_Inverse>
      <Marker-1>L_Hip</Marker-1>
    </Norm_Inverse>
  </Velocity_transformation>
  <Velocity_transformation>
    <Norm_Acceleration>
      <Marker-1>R_Shoulder</Marker-1>
    </Norm_Acceleration>
  </Velocity_transformation>
  <Velocity_transformation>
    <Norm_Deceleration>
      <Marker-1>R_Elbow</Marker-1>
    </Norm_Deceleration>
  </Velocity_transformation>
</Transformations_history>
```

General information about the C3D file

Model used

Vertical mirror applied

Rotation around Y-axis of 40° about R_Heel point light

List of the exchanges performed during a shuffled scramble transformation

50 masks flashing at 11 Hz added

60 linear masks with random velocity added

40 random masks with random velocity added

2 scrambled duplications of the initial markers set

X and Z components of the R_Heel velocity set as

Y component of the R_Foot velocity set as inversed

Manual transformations are listed as successive transformations:

X component of the R_Knee velocity set as constant

Y component of the R_Knee velocity set as inversed

Z component of the R_Knee velocity manually set

Norm constant transformation for R_Hip

Norm inversed transformation for L_Hip

Norm accelerated transformation for R_Shoulder

Norm decelerated transformation for R_Elbow

## 7.2. Use Plugin

*// Description*
*Translate all point lights of a given distance along X, Y and Z axes.*
*// Author*
*A. Decatoire*
*// Version*
*1.0*

The red parts are mandatory..

The script file should be written as follow (red parts are mandatory):

- The first line syntax is:

  *function Outputs=Name_Of_Plugin(Inputs)*

  Choose a valid name for your plugin (i.e. without space and forbidden characters)

- The last line syntax is:

  *end % function*

- There is no particular restriction for the main body of the script except comments and Matlab automatic message at the end of line. However, comments are welcome to make the analysis of the plugin easier.

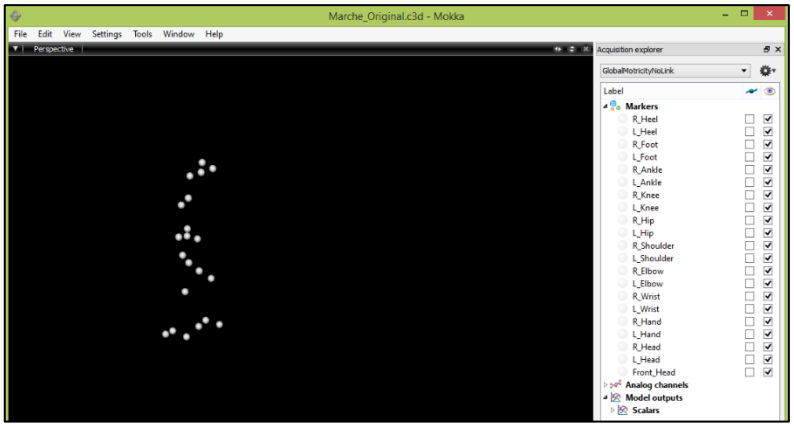
```
% Check validity of settings (numerical values)
ok=1;
str={'X';'Y';'Z'};
msg={''};
for j=1:3
    if isempty(str2double(answer{j})) | isnan(str2double(answer{j})) %#ok<OR2>
        ok=0;   % Updat output
        default(j)=0;
```

There are two kinds of plug-in:

- Plug-in that makes transformations on one or several C3D files loaded into the plug-in itself: in this case, the Inputs variable is an empty matrix

- Plug-in that makes transformations on a previously loaded C3D files in PLAViMoP: in this case, the Inputs variable is a structure with the following fields:
  - Markers (structure)
  - Scalars (structure)
  - Analogs (structure)
  - Rate : frame rate of the C3D file (1 x 1 matrix)

Markers, Scalars and Analogs fields have fields too, whose labels are the name of the markers, scalars and analogs data displayed in Mokka.



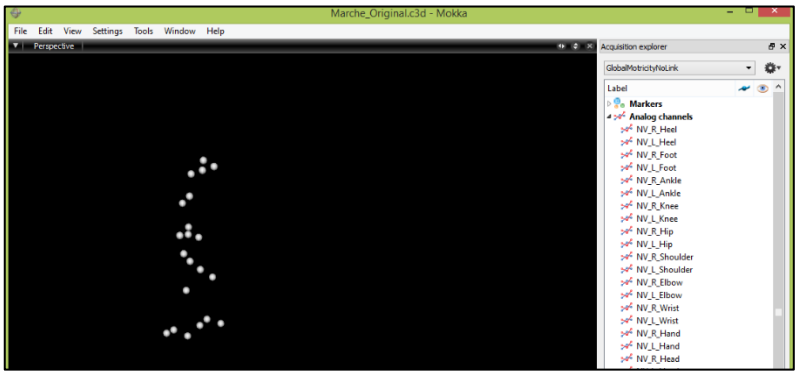

These data are stored in the Inputs variable and send to the plug-in function as n x 3 matrix (where n is the number of frames):

Inputs.Markers.R_Heel

Inputs.Markers.L_Heel

Inputs.Markers.R_Foot

…



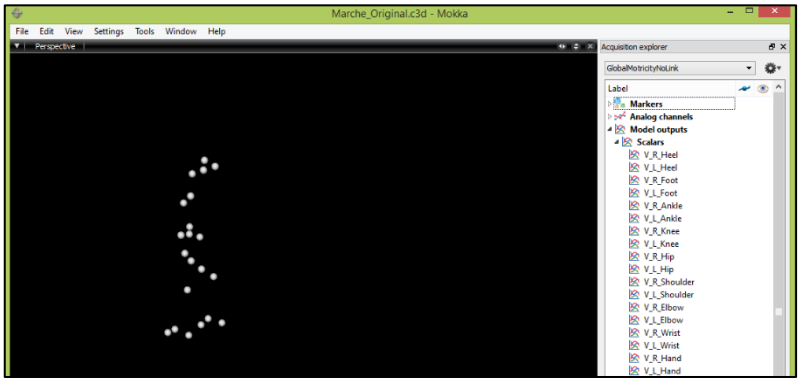

These data are stored in the Inputs variable and send to the plug-in function as n x 1 matrix (where n is the number of frames):

Inputs.Analogs.NV_R_Heel

Inputs.Analogs.NV_L_Heel

Inputs.Analogs.NV_R_Foot

…




These data are stored in the Inputs variable and send to the plug-in function as n x 3 matrix (where n is the number of frames):

Inputs.Scalars.V_R_Heel

Inputs.Scalars.V_L_Heel

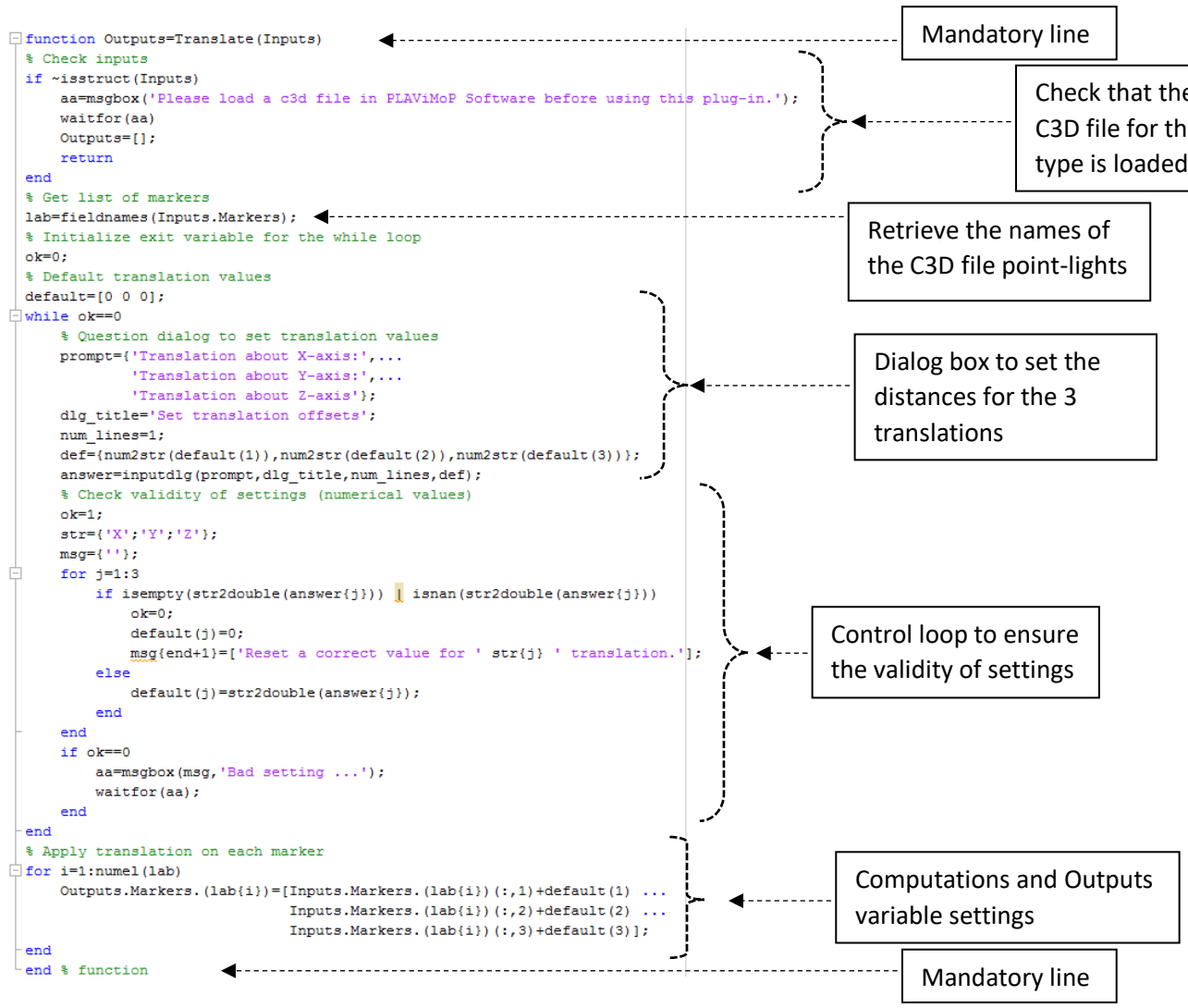
Inputs.Scalars.V_R_Foot

…

The Outputs variable should be organized in the same way, i.e. with Markers &/or Scalars &/or Analogs fields.

Here is a Matlab plug-in script example allowing to translate the point-lights of a C3D file along X, Y and Z-axis. The distances are set by the user through an input dialog box.

Example of plug-in that needs a C3D file loaded in PLAViMoP Software:



Example of plug-in that asks for C3D file(s) into the plug-in itself:

```
function Outputs=MixC3D(Inputs)
% Initialize Outputs
Outputs=[];
% Get files
[FileName1,PathName1]=uigetfile({'*.c3d'},'File Selector: C3D #1');
if FileName1==0
    return
end
[FileName2,PathName2]=uigetfile({'*.c3d'},'File Selector: C3D #2');
if FileName2==0
    return
end
% Get list of markers
c3d1=btkReadAcquisition([PathName1 FileName1]);
c3d2=btkReadAcquisition([PathName2 FileName2]);
markers1=btkGetMarkers(c3d1);
markers2=btkGetMarkers(c3d2);
lab1=fieldnames(markers1);
lab2=fieldnames(markers2);
nFrames1=size(markers1.(lab1{1}),1);
nFrames2=size(markers2.(lab2{1}),1);
FrameRate1=btkGetPointFrequency(c3d1);
FrameRate2=btkGetPointFrequency(c3d2);
% New C3D
max([nFrames1 nFrames2])
c3d=btkNewAcquisition(0,max([nFrames1 nFrames2]));
btkSetFrequency(c3d,max([FrameRate1 FrameRate2]));
for i=1:numel(lab1)
    if nFrames1>nFrames2
        btkAppendPoint(c3d,'marker',lab1{i},markers1.(lab1{i}));
    else
        btkAppendPoint(c3d,'marker',lab1{i},[markers1.(lab1{i});ones(nFrames2-nFrames1,3).*NaN]);
    end
end
for i=1:numel(lab2)
    if nFrames2>nFrames1
        if ismember(lab2{i},lab1)
            btkAppendPoint(c3d,'marker',[lab2{i} '_2'],markers2.(lab2{i}));
        else
            btkAppendPoint(c3d,'marker',lab2{i},markers2.(lab2{i}));
        end
    else
        if ismember(lab2{i},lab1)
            btkAppendPoint(c3d,'marker',[lab2{i} '_2'],[markers2.(lab2{i});ones(nFrames1-nFrames2,3).*NaN]);
        else
            btkAppendPoint(c3d,'marker',lab2{i},[markers2.(lab2{i});ones(nFrames1-nFrames2,3).*NaN]);
        end
    end
end
btkWriteAcquisition(c3d,[path file]);
aa=msgbox({[file ' created sucessfully in:'];path},'Informations');
waitfor(aa)
end % function
```

Mandatory line

No outputs send to PLAViMoP at the end of the plug-in

Load C3D file(s) into the plug-in

Read contents of the C3D files

Create a new C3D file

Save the new C3D file

Mandatory line