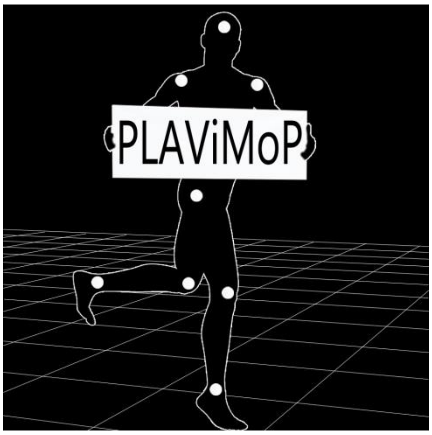
PLAViMoP 2 Software

User Manual

Written by Arnaud DECATOIRE & Christel Bidet-Ildei

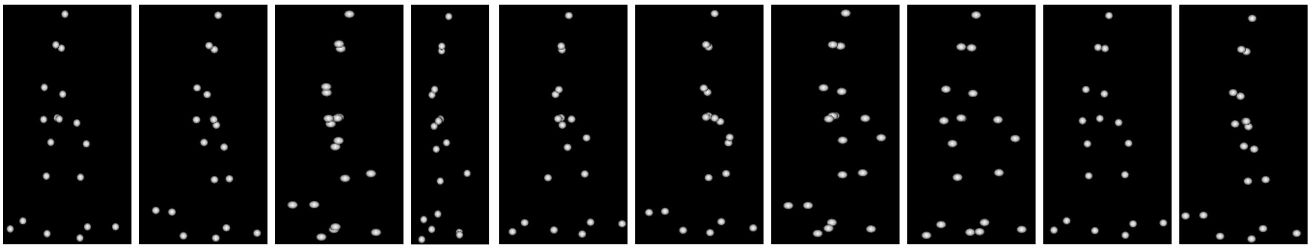
Version 1 – 2020



CONTENTS

1. General presentation

PLAViMoP 2 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 a window of visualization. *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

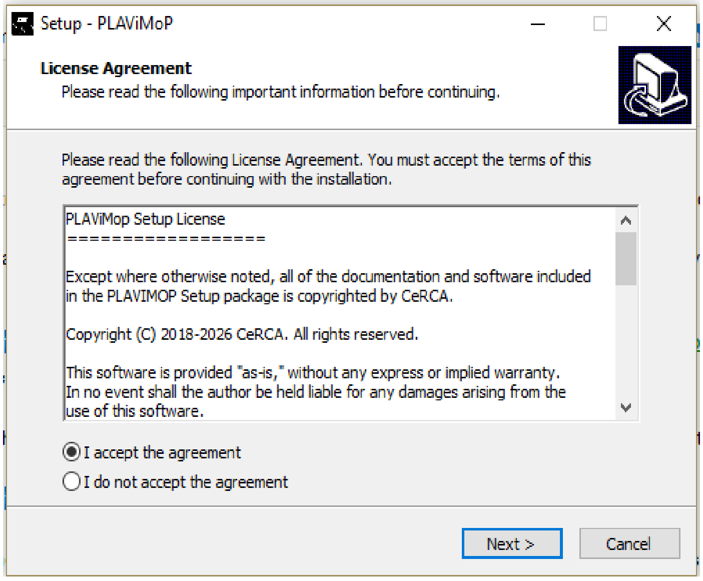
Both Matlab interface and Moveck software are required, as well as Windows 64 bits system and an internet connection for installation.

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.

Go to https://plavimop.prd.fr/en/contact and ask PLAViMoP Software release 2.

When you receive the link, download the software installation package ***setupPlavimop.exe***



Setup - PLAViMoP

License Agreement

Please read the following important information before continuing.

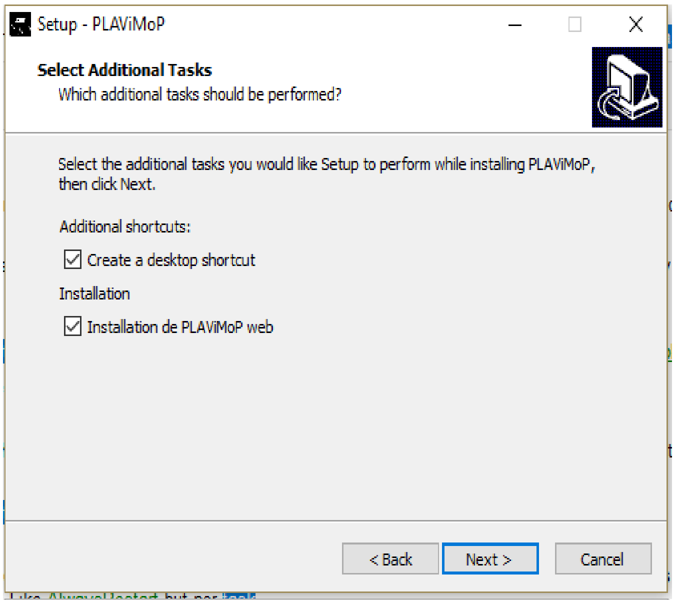
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

```
PLAViMop Setup License
==================

Except where otherwise noted, all of the documentation and software included
in the PLAVIMOP Setup package is copyrighted by CeRCA.

Copyright (C) 2018-2026 CeRCA. All rights reserved.

This software is provided "as-is," without any express or implied warranty.
In no event shall the author be held liable for any damages arising from the
use of this software.
```

● I accept the agreement

○ I do not accept the agreement

After launched the installation, you must accept the agreement.



Setup - PLAViMoP

Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing PLAViMoP, then click Next.

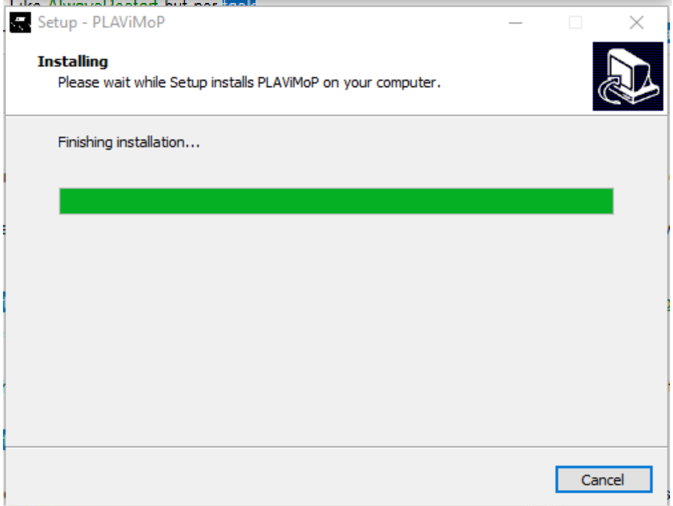
Additional shortcuts:

☑ Create a desktop shortcut

Installation

☑ Installation de PLAViMoP web

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



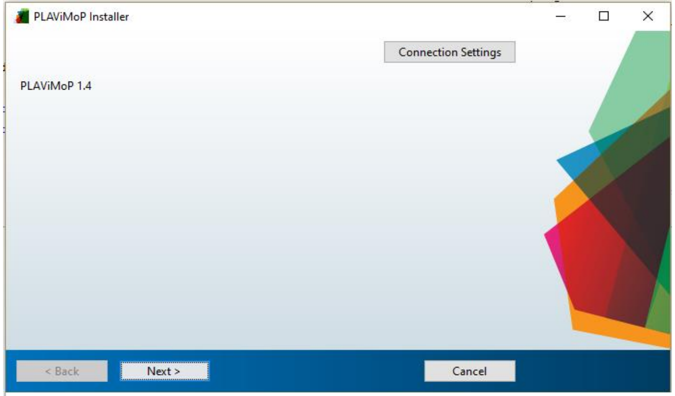
Setup - PLAViMoP

Installing

Please wait while Setup installs PLAViMoP on your computer.

Finishing installation...

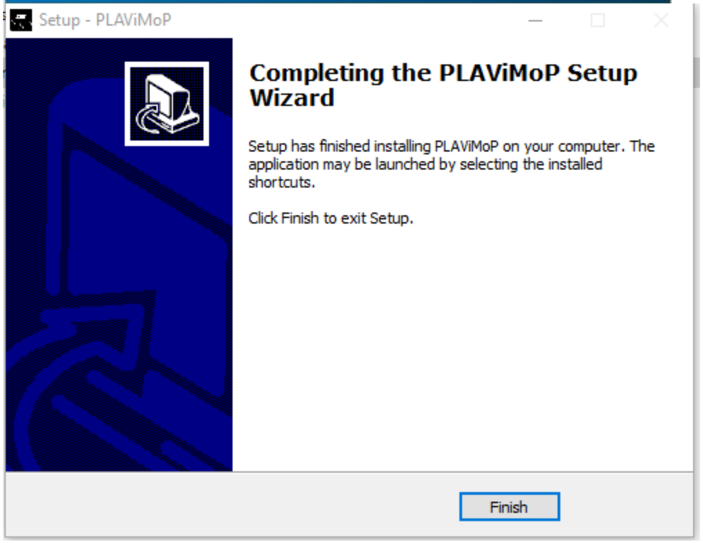
The first part of the installation is ended. You need to wait time … perhaps long time.



PLAViMoP Installer

PLAViMoP 1.4

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



Setup - PLAViMoP

Completing the PLAViMoP Setup Wizard

Setup has finished installing PLAViMoP on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

Great ! !

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

Have enjoy !

After installation, a folder link “PLAViMoP 2 ressources” have been created on your desktop.

3. Input / output formats

3.1. Input data format

Only the C3D format is supported by PLAViMoP 2 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.

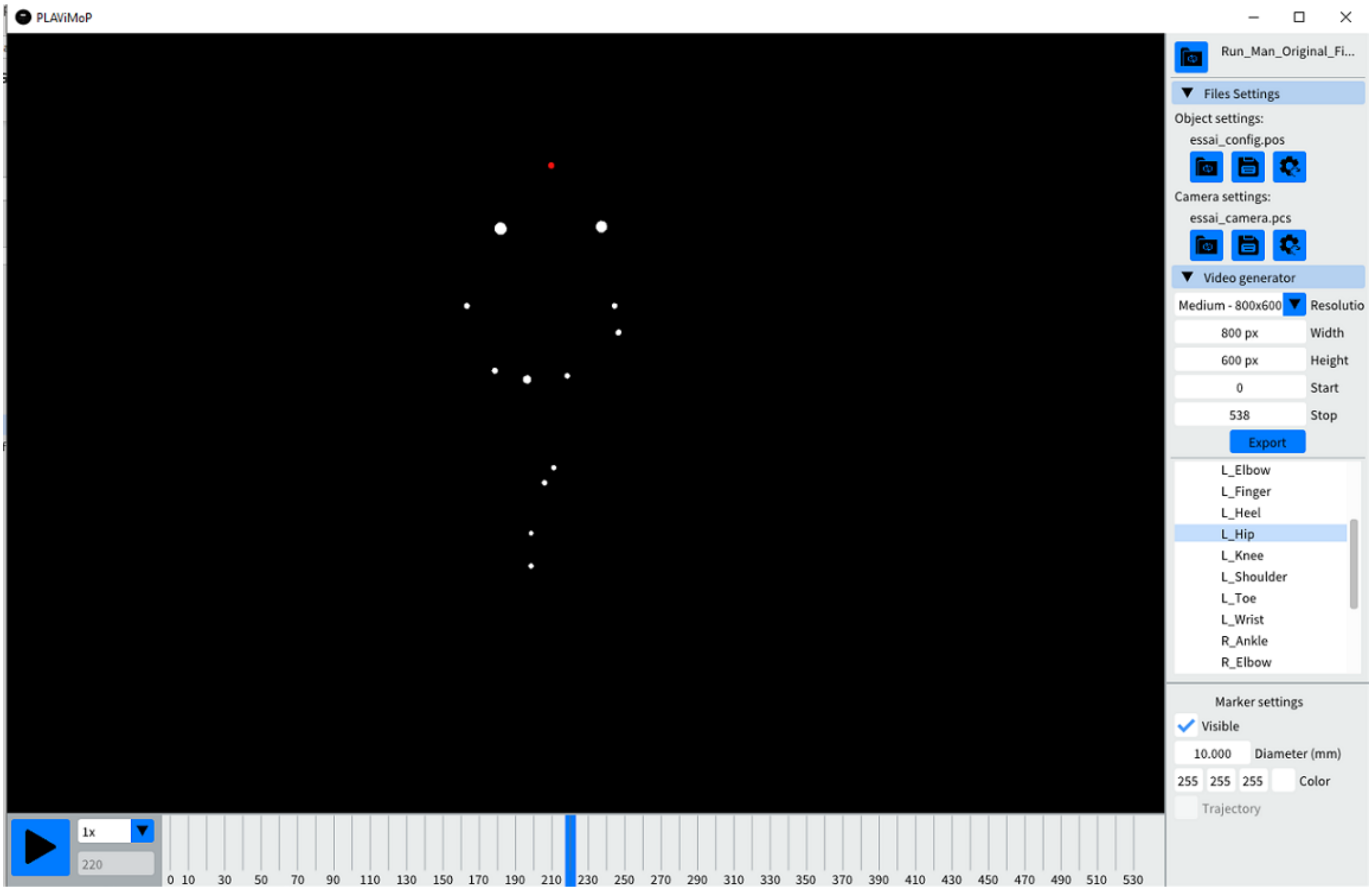
If you do not have C3D files, you can find some files in the PLAViMoP Database https://plavimop.prd.fr/en/motions or you can create them from a .CSV file containing all information on the PLD (3D time histories of markers, names of marker components and a time column). For this, you can use the plug-in “CSVtoC3D” proposed in PLAViMoP 2 (for more information on the use of plug-in, see here).

3.2. Output data formats

After the process, the data can be saved as C3D file as well as a video (*.mov file) or a data table (*.csv file) via PLAViMoP 2.

4. Moveck functionalities

4.1. Presentation of Moveck Interface



Run_Man_Original_Fi...

Object settings: essai_config.pos

Camera settings: essai_camera.pcs

Medium - 800x600 Resolutio

800 px Width

600 px Height

0 Start

538 Stop

☑ Visible

10.000 Diameter (mm)

255 255 255 Color

Figure 1: Presentation of the Moveck interface. On the left, a 3D view displaying the content of the C3D file. On the right, you have access to some tools (e.g., specify the object settings, the camera settings, generate a video).

4.2. Moveck visualization configuration files

The way Moveck displays the content of a C3D file depends on the object (*.pos) and the camera (*.pcs) settings.

4.2.1. Object settings (*.pos): Markers appearance

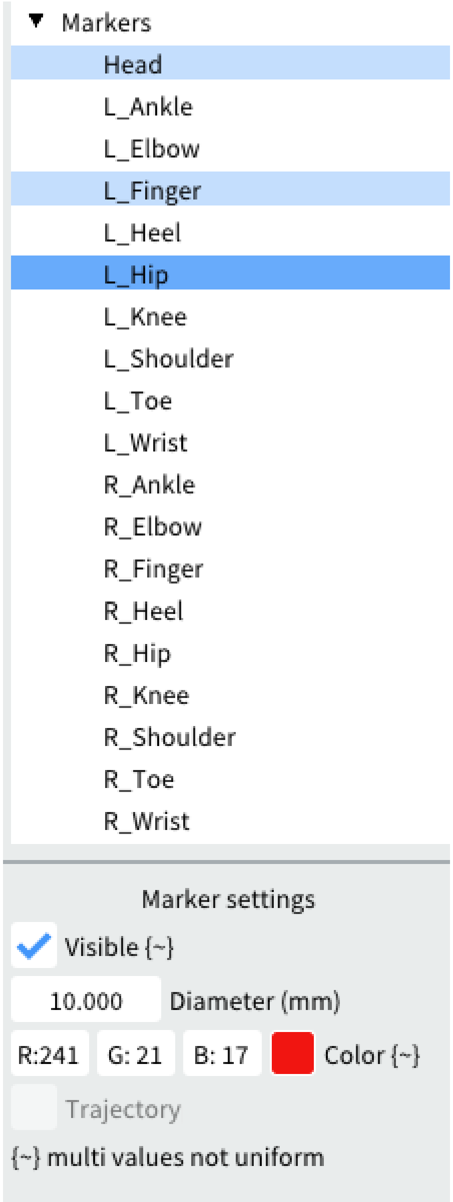
For each marker, you can choose to show or hide it and you can change its size and its colour.

For that, click on a marker’s label listed under the “Markers” node and check/uncheck “visible” in “Markers settings” area to show/hide it, change the diameter of the selected markers and click on the coloured square to modify the colour.

Notes:

- If you want to apply the same modification to several markers, just select them by maintaining the CRTL key before changing the markers’ properties.

- As the size of the markers are their real sizes and as the markers can be more or less distant from the camera (in the perspective mode), two markers with the same size can appear as two circles with different size.



▼ Markers

Head
L_Ankle
L_Elbow
L_Finger
L_Heel
L_Hip
L_Knee
L_Shoulder
L_Toe
L_Wrist
R_Ankle
R_Elbow
R_Finger
R_Heel
R_Hip
R_Knee
R_Shoulder
R_Toe
R_Wrist

Marker settings

☑ Visible {~}

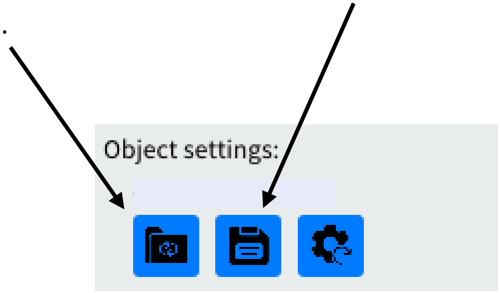
10.000 Diameter (mm)

R:241 G: 21 B: 17 🟥 Color {~}

□ Trajectory

{~} multi values not uniform

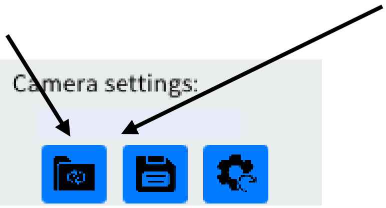
All the characteristics chosen can be saved by using “save Json configuration file” in the object settings at the right top of the interface and reloaded.



Object settings:

4.2.2. Camera settings (*.pcs): 3D scene appearance

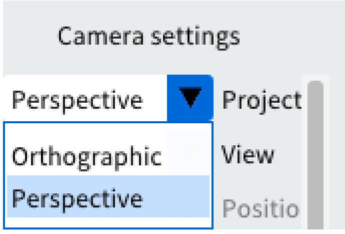
All the following characteristics can be saved by using “save Json configuration file” in the camera settings at the right top of the interface and reloaded

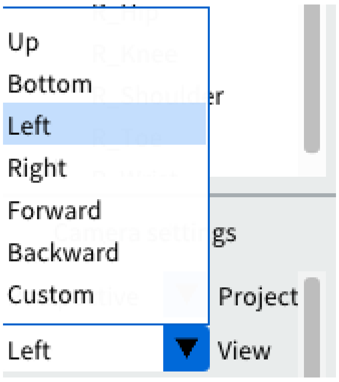


Camera settings:

4.2.2.1. The camera tools

In the camera settings (click on “Camera” to display them), you can choose the projection mode and the view of visualization. You can switch between “orthogonal“” and “perspective” projection and between up, bottom, left, right, forward, backward and Custom view with the tool in the right corner.





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

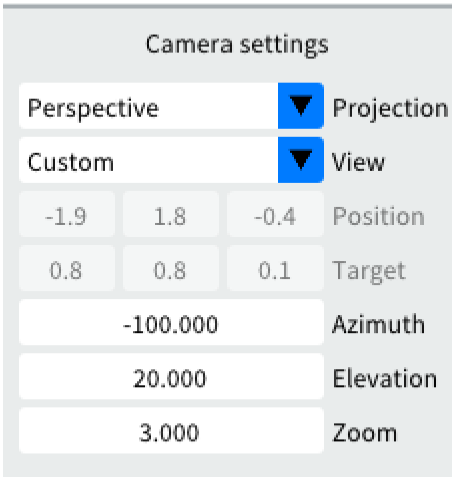
For both orthogonal and perspective projections:

- 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.

In this setting, you can also directly fix the position of the camera (elevation and azimuth) and the zoom of the C3D.

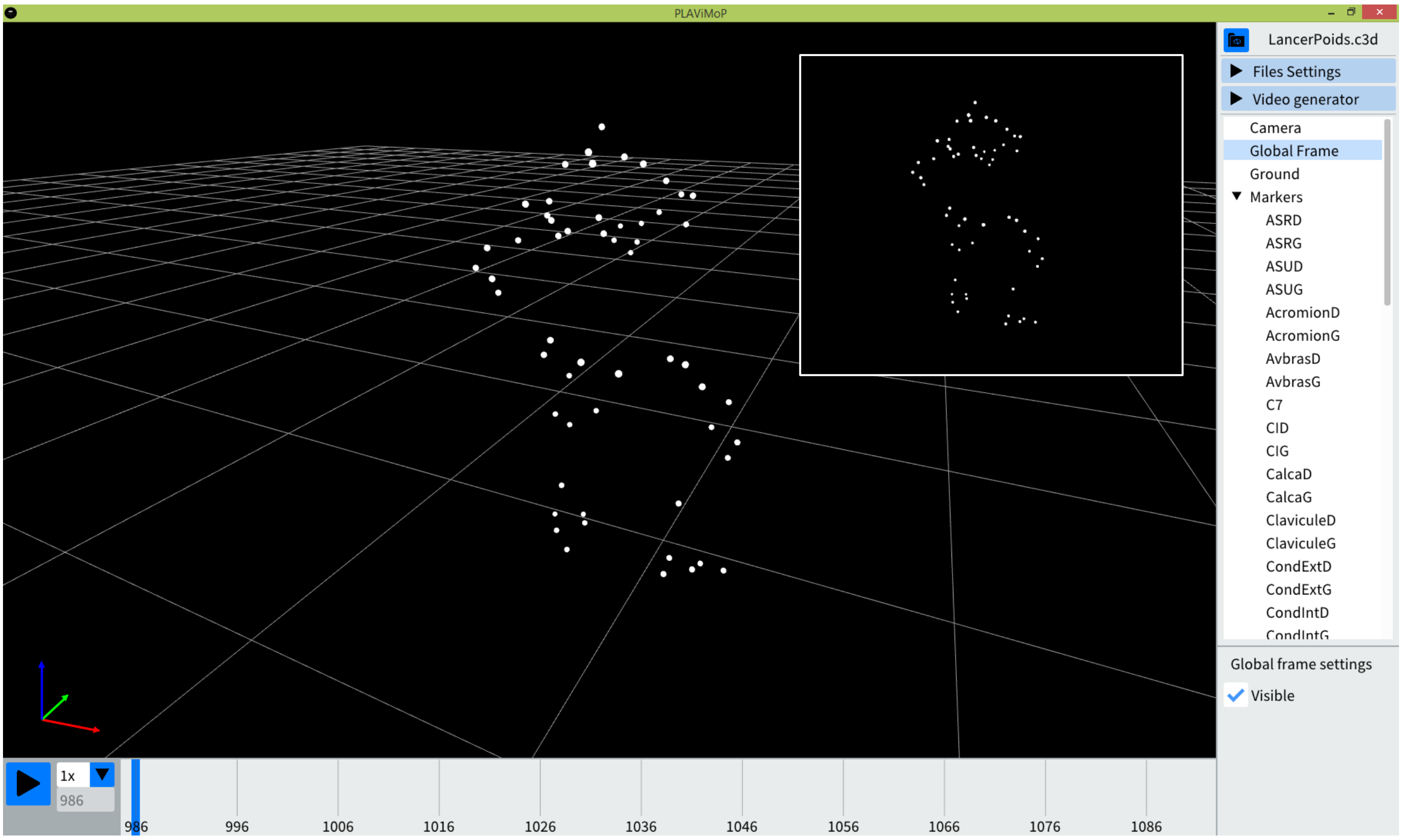


Camera settings

Perspective			Projection
Custom			View
-1.9	1.8	-0.4	Position
0.8	0.8	0.1	Target
-100.000			Azimuth
20.000			Elevation
3.000			Zoom

4.2.2.2. Global frame and ground

You can choose to show or hide the global frame and the ground in the 3D view window. For this, simply check or uncheck the case “visible” after clicking on “Global Frame” and/or “Ground”.



Global frame settings

☑ Visible

4.2.3. Create a personal object and camera settings

To summarize this part:

1) Modify directly the proprieties of the objects (for example change the colour or the size of some dots of the C3D file) and of the camera (size of the character, position in the window).

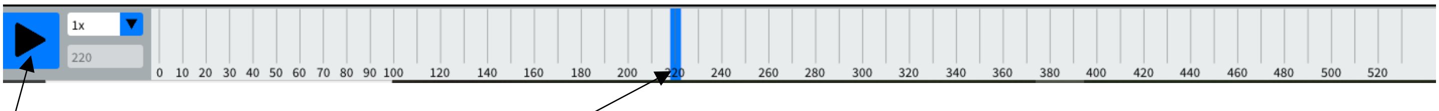
2) Save your configurations by using the tools “save Json configuration file” for both object and camera configurations.

Once you have created an object and/or a camera configuration, you can visualize others C3D files with these configurations by clicking to “load Json configuration file” and select the required file.

4.3. Play the C3D file

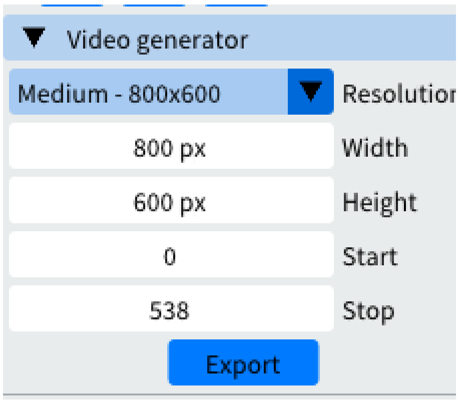
To play the content of a C3D file:

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



4.4. Generate a video

You can generate a video (*.mov) from your c3d. This video will keep all the characteristics of the C3D file (size, colour, view, etc.). For this simply, choose the resolution (Low, Medium, High or Custom), the start and stop frames and click on “Export”.



▼ Video generator

Medium - 800x600 ▼ Resolutior

800 px Width

600 px Height

0 Start

538 Stop

5. PLAViMoP 2 functionalities

To start the application, double click on the PLAViMoP2.exe file. It will launch the Matlab graphical user interface and Moveck application.

The user interface is divided in four zones:

- Load movement
- Spatial transformation
- Velocity transformation
- Exportations/Plug-in

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 Moveck with the Moveck 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.

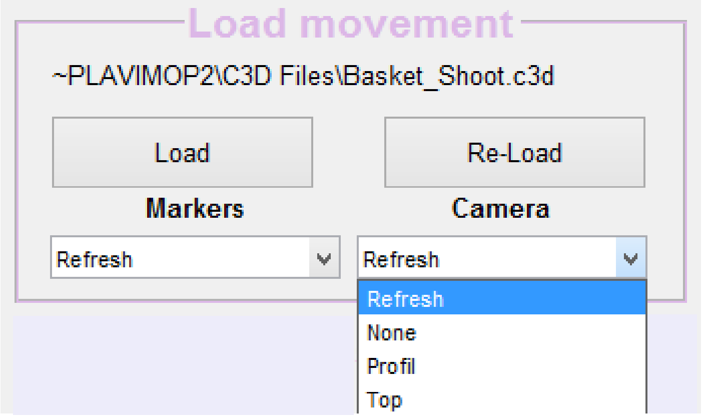
If you are “lost” in your transformation, you can easily reload the original C3D file with the **Re-Load** button.

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 2 Software.

5.2. Load a camera and an object configuration file

A C3D file can be loaded with particular Moveck Visualisation camera (*.pcs) and objects(*.pos) configuration files. For that, use the **popup menu** of the load movement zone to choose the files. The files listed here are the ones present in the “Camera settings” and “Markers settings” folders of the application.



If you have created a new *.pos or *.pcs file with Moveck since PLAViMoP 2 have been started, you can refresh the list by selecting ”Refresh” in the corresponding popup menu.

5.3. Spatial transformation

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

5.3.1. *Mirror*

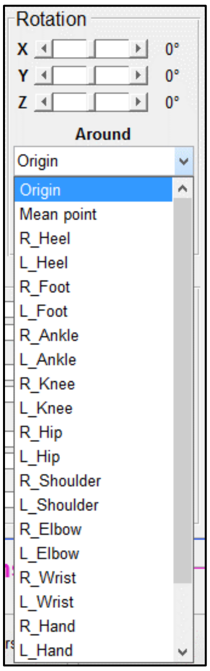
This transformation enables to create horizontal, lateral or vertical symmetry of the original motions.

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



Rotation

X 0°

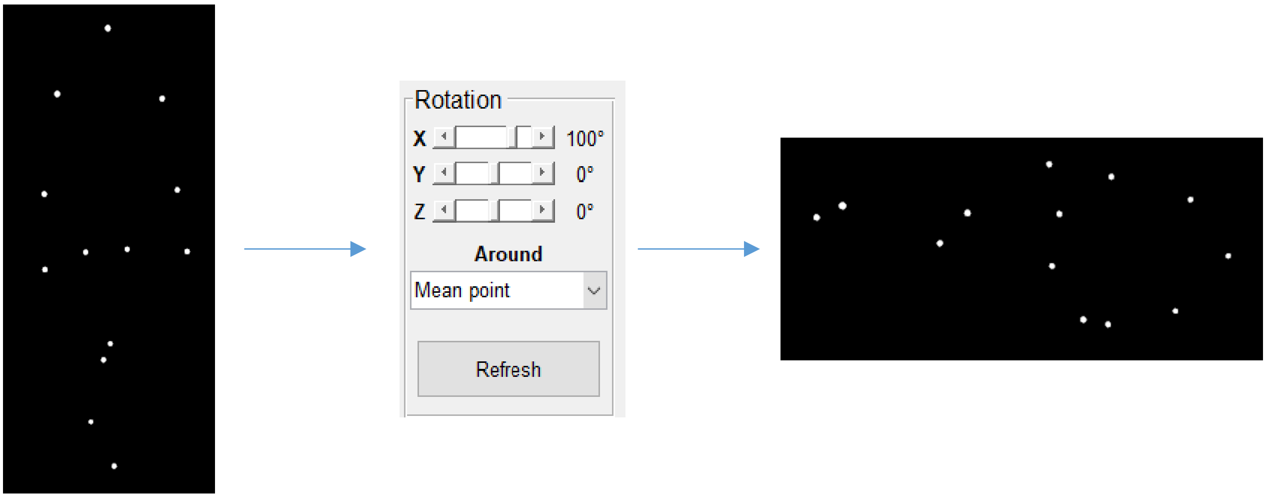
Y 0°

Z 0°

Around: Origin

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 100° around X-axis made about the mean point:



Rotation

X 100°

Y 0°

Z 0°

Around: Mean point

5.3.3. *Scrambled*

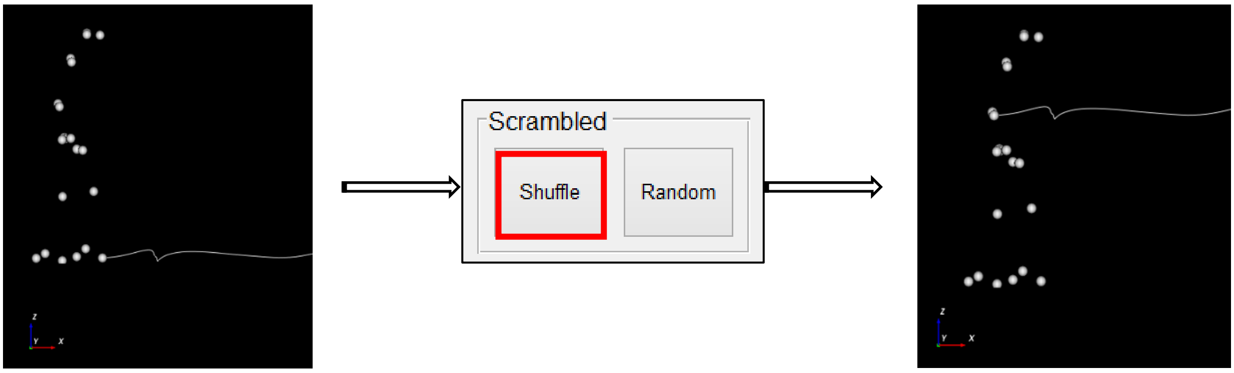
This transformation enables to scramble each point constituting the sequence.

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.

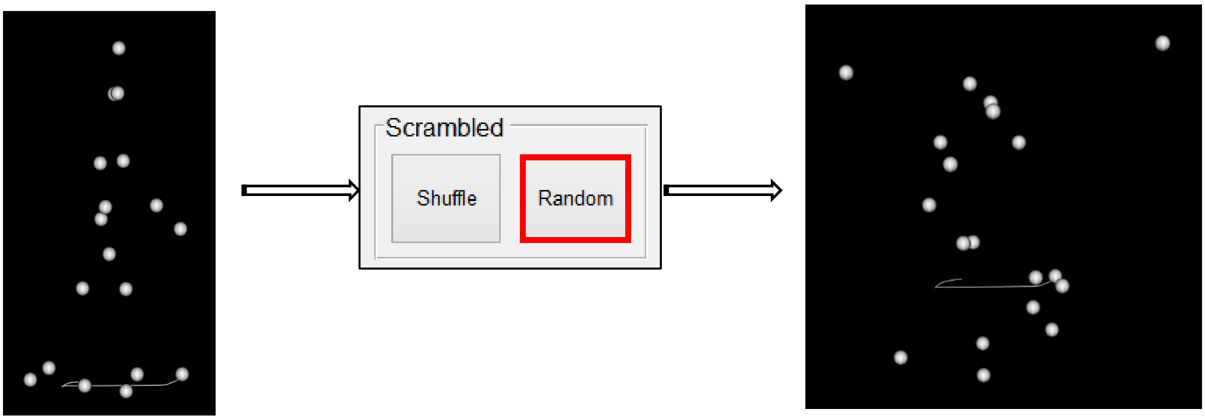
In the example presented below, the right elbow randomly inherited the trajectory and dynamic of the left foot marker.



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.

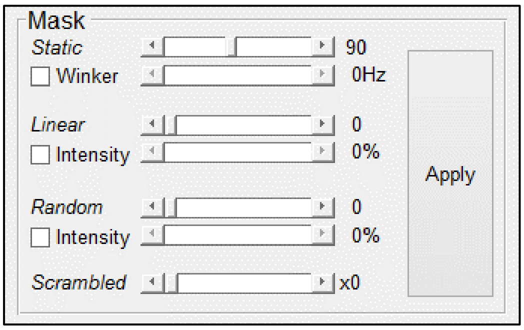


5.4. Masking PLDs

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.



Mask

Static 90

□ Winker 0Hz

Linear 0

□ Intensity 0%

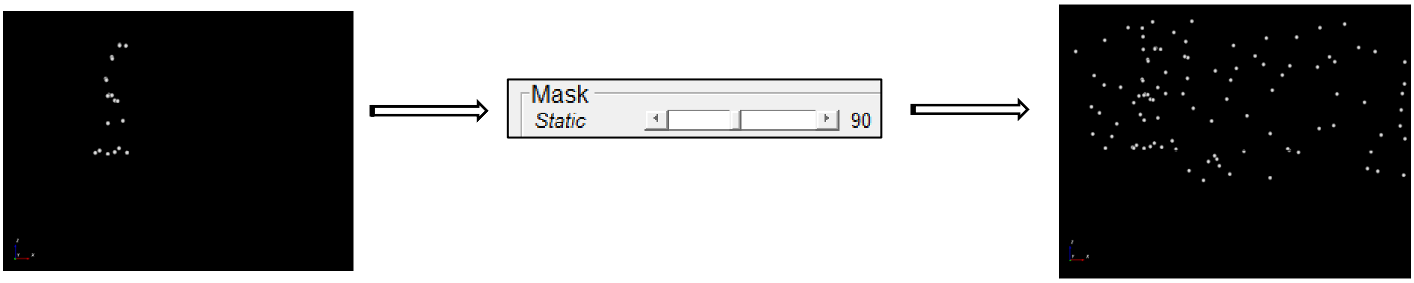
Random 0

□ Intensity 0%

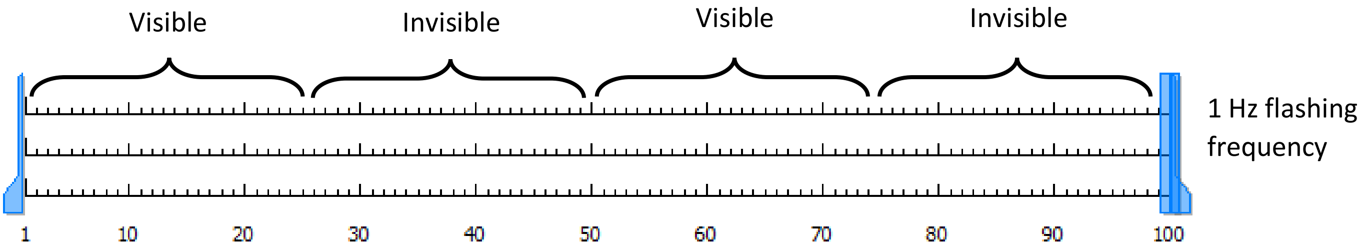
Scrambled x0

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.

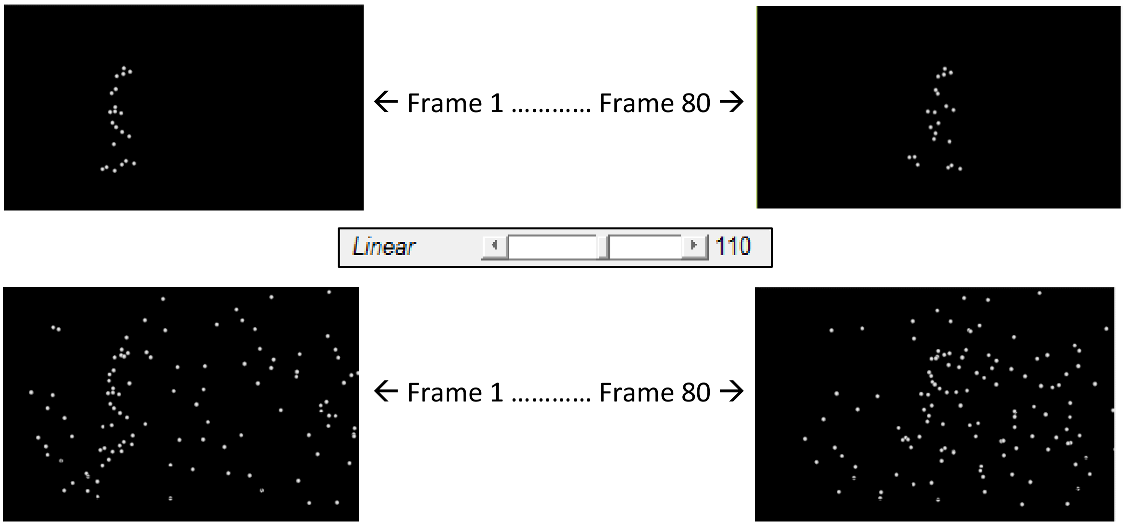


The second behaviour occurs if the **Winker** button is checked. The associated **slider** becomes enable. Use it to specify the flashing frequency (from 1 to 25 Hz). As the C3D file frame rate is 100 Hz, a flashing frequency of 1 Hz causes a mask being visible/invisible alternatively during 25 consecutive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

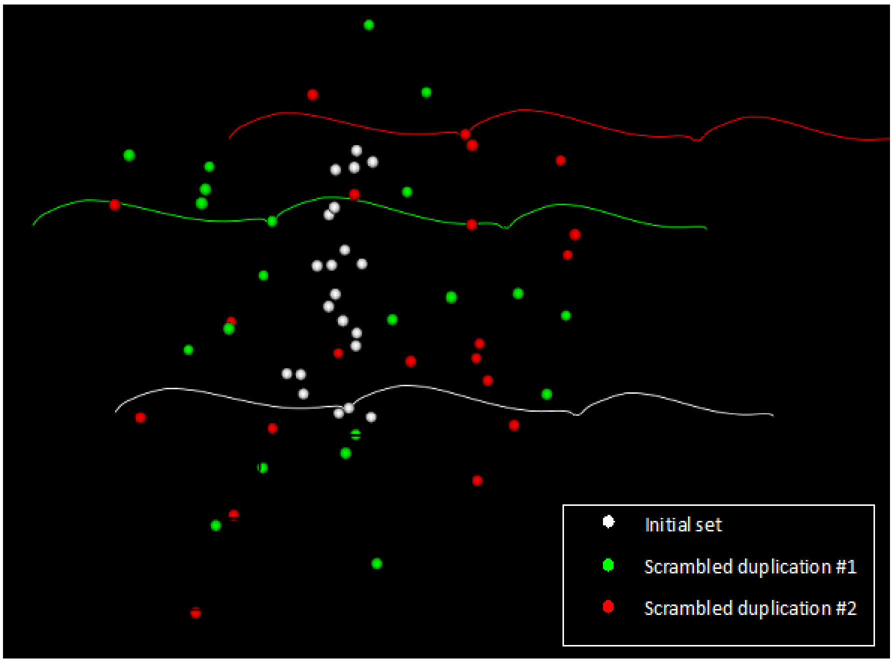
The random mask is a point light with a randomly defined trajectory. Both initial position and instantaneous acceleration are randomly chosen. Masks move along the three axes. A control loop ensures that the mask stays in the bounding box limits (rebound).

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:

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

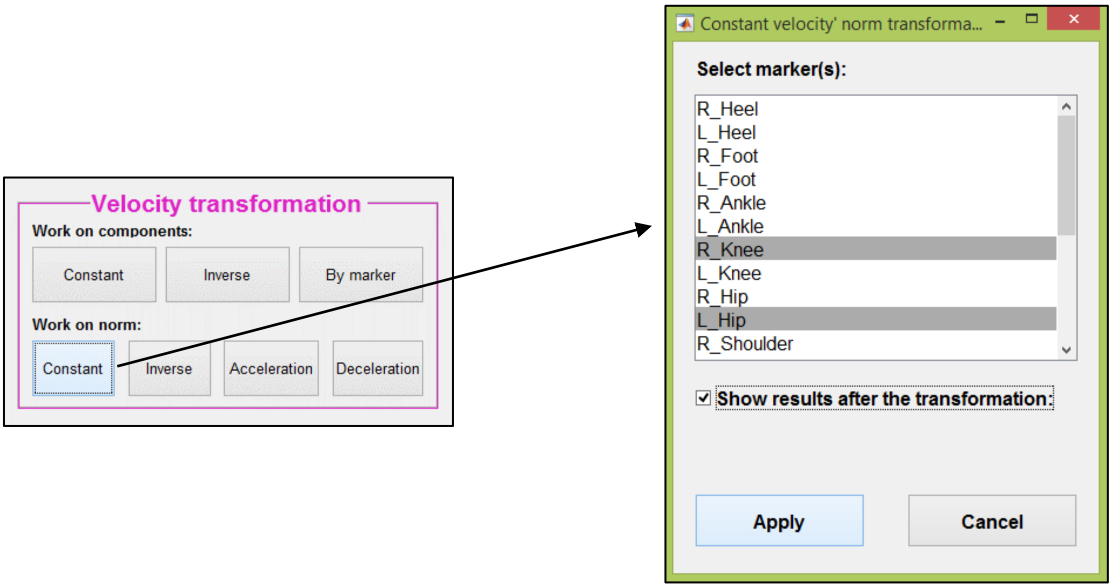
All transformations detailed below enable to modify the dynamic of the original sequence but in keeping the original trajectory and movement duration.

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 2, 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 and can be accessible in the csv table (see here).

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.

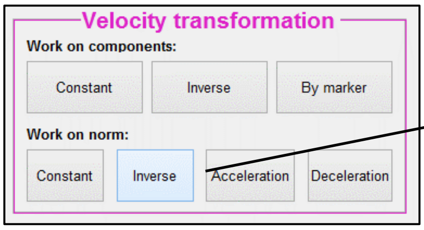
In PLAViMoP 2, to process this kind of transformation, just click on the **Inverse** button of the “Work on norm” section to open the point light selection window.

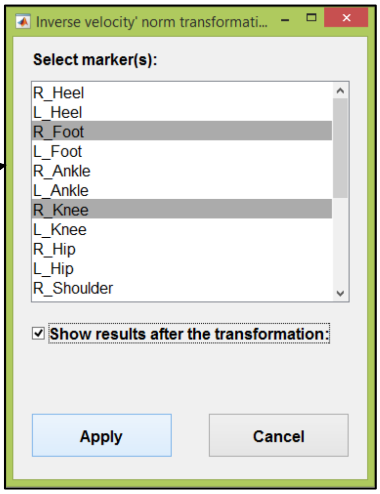
You can select several markers at the same time by holding Ctrl key.

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 and can be accessible in the csv table (see here).





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

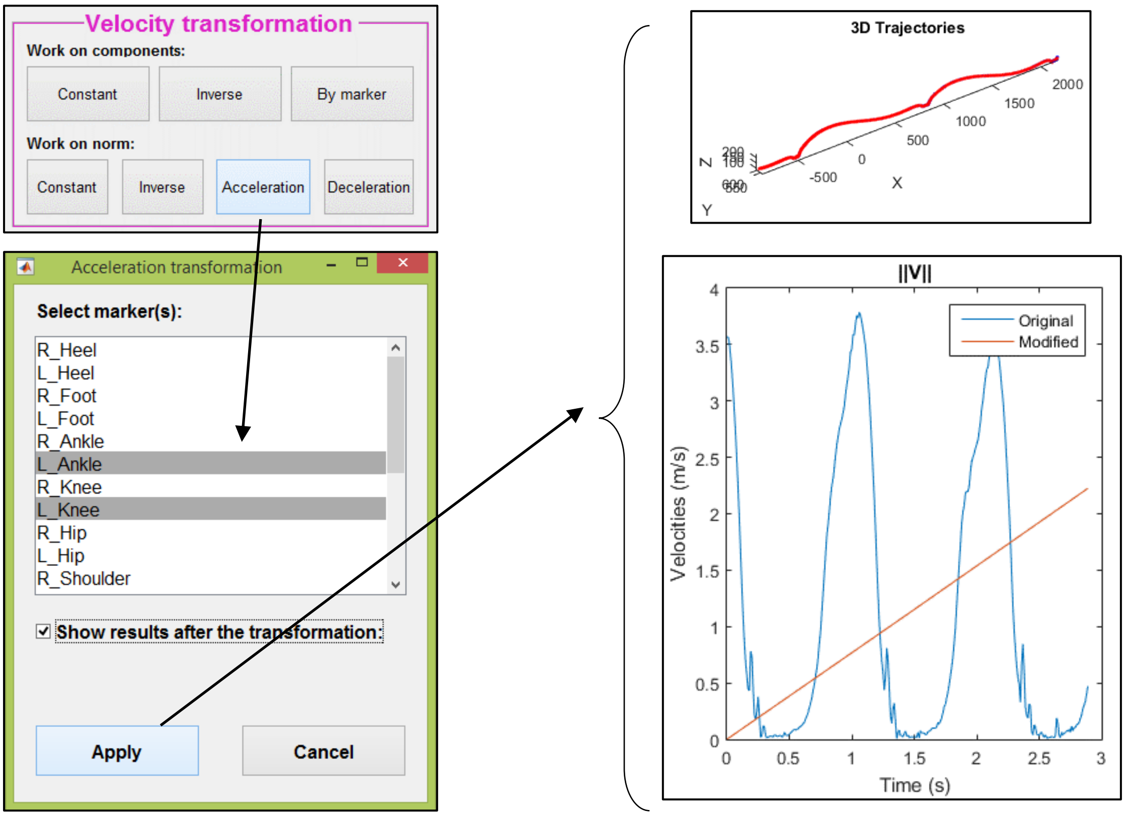
In PLAViMoP 2, to process this kind of transformation, just click on the **Acceleration** button of the "Work on norm" section to open the point light selection window.

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 and can be accessible in the csv table (see here).



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

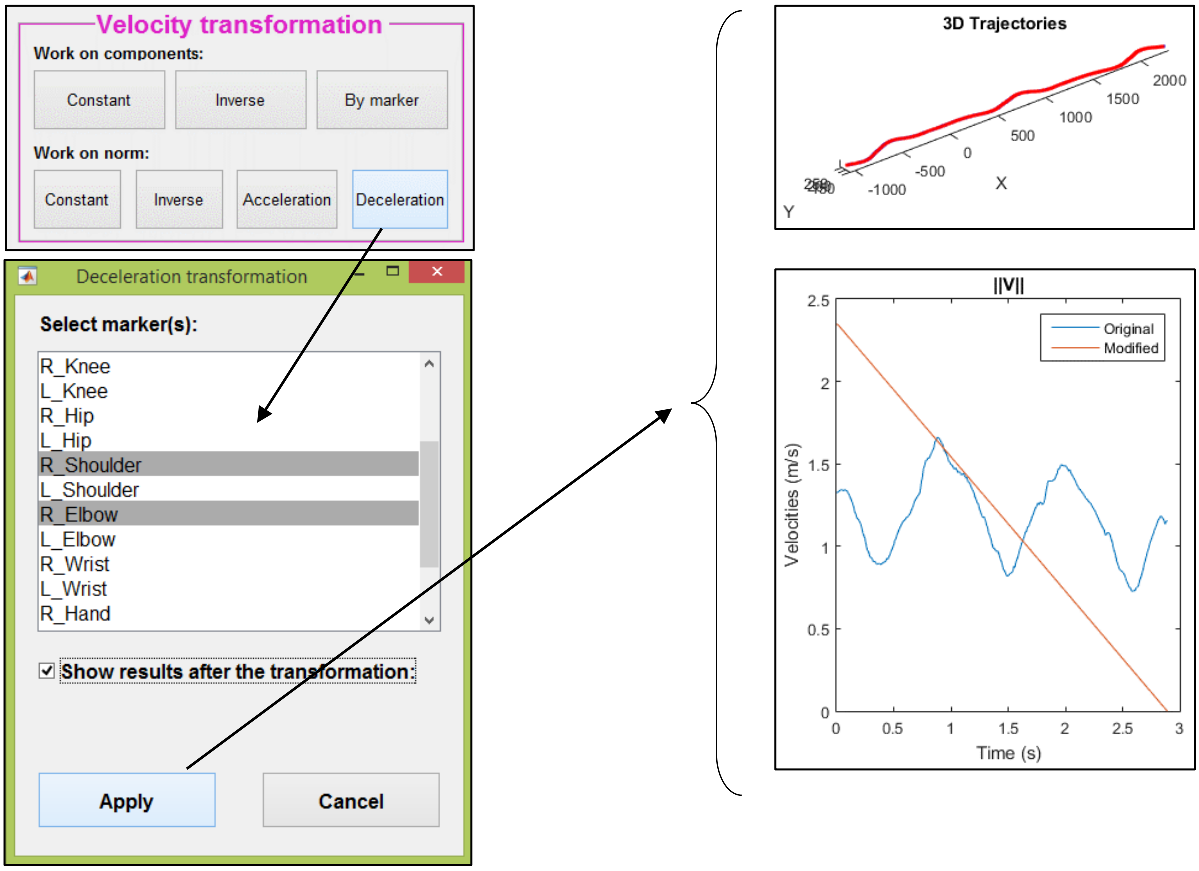
In PLAViMoP 2, to process this kind of transformation, just click on the **Deceleration** button of the "Work on norm" section to open the point light selection window.

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 and can be accessible in the csv table (see here).



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

5.5.2. Transformations applied to each component of the velocity

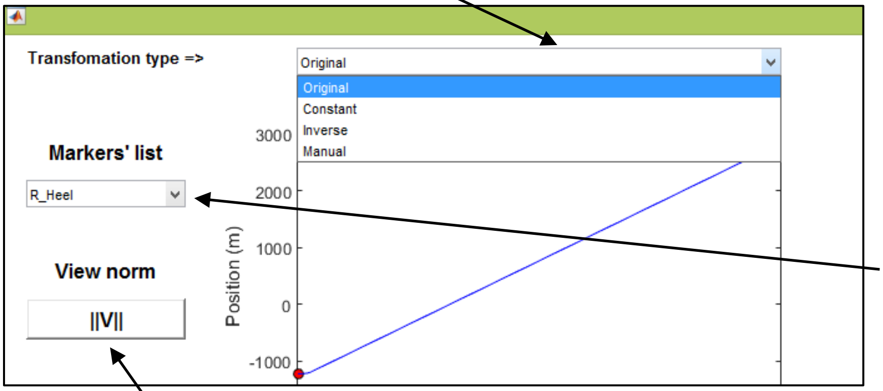
The main difference between this series of transformations and the previous ones is that the initial trajectory of point light is not maintained after the transformation. The modifications are applied to one, two or three components of the velocity for a given point light and it results in a new velocity 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 2 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**.

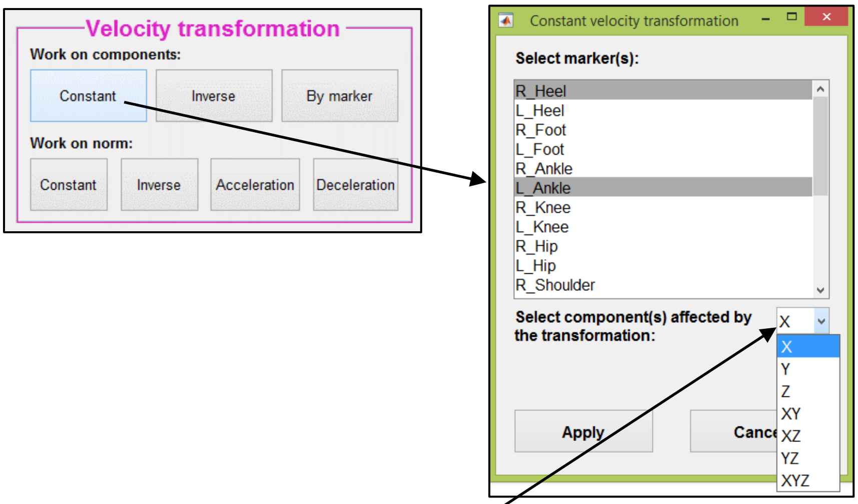
All the transformations are memorized and are definitively applied to the C3D file when closing the window. So it is not necessary to close the window after each point light transformation.

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.



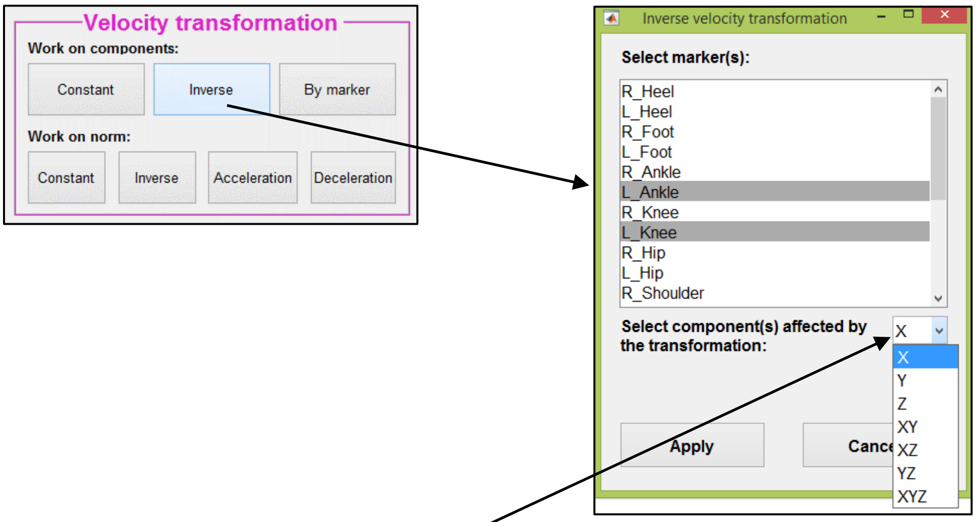
In the opening window, use the popup menu to select the component(s) that will be affected by the transformation.

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.

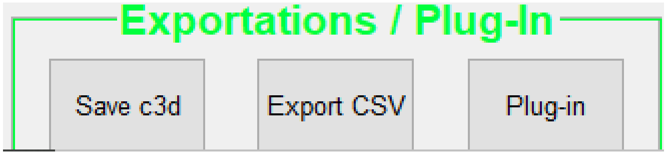
Click on **Apply** button to finalize the transformations. The C3D file will be updated, the new velocity and acceleration components and norms will be written and available as described here.

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 a new c3d or a csv with all the coordinates of the movement. You can also use pre-installed plug-in or your own transformations routines.

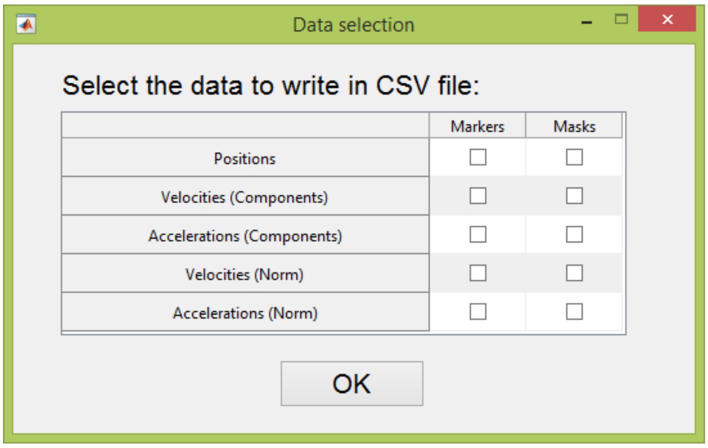


5.7.1. Save as C3D

To save your C3D, click on save C3D and choose a directory and a name.

5.7.2. Export CSV

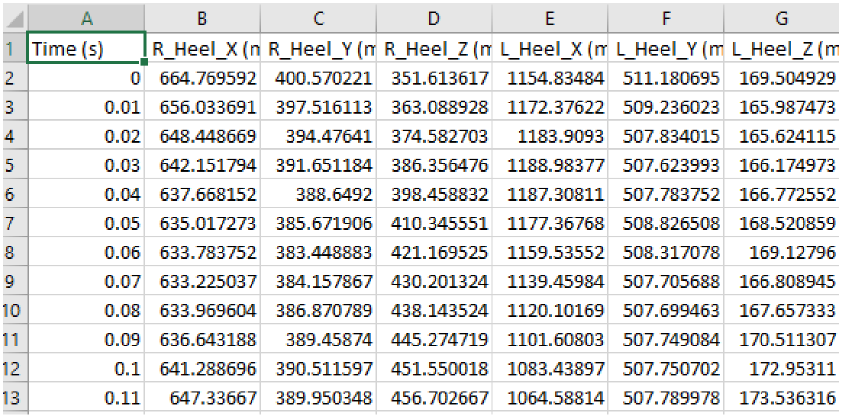
This button allows to save 3D coordinates (X, Y, Z) of markers &/or masks associated with your C3D, as well as their 3D velocities' and accelerations' components and norms Once you give a valid filename, just check the corresponding boxes in the following window and click the "OK" button.



Select the data to write in CSV file:

	Markers	Masks
Positions	□	□
Velocities (Components)	□	□
Accelerations (Components)	□	□
Velocities (Norm)	□	□
Accelerations (Norm)	□	□

A CSV file is created with columns of data separated by semicolon character.

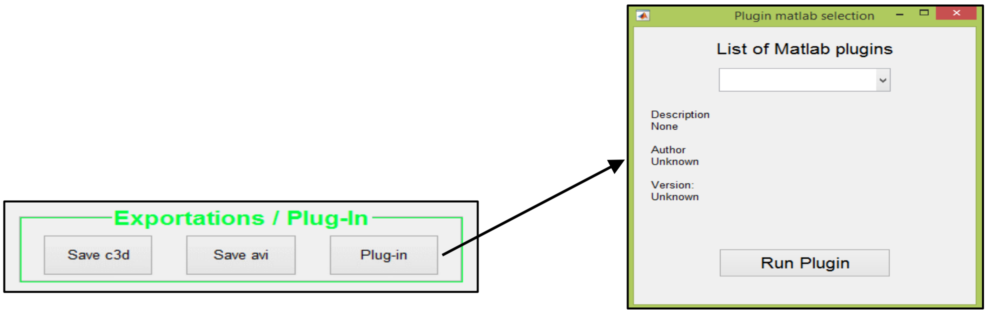


	A	B	C	D	E	F	G
1	Time (s)	R_Heel_X (n	R_Heel_Y (m	R_Heel_Z (m	L_Heel_X (m	L_Heel_Y (m	L_Heel_Z (m
2	0	664.769592	400.570221	351.613617	1154.83484	511.180695	169.504929
3	0.01	656.033691	397.516113	363.088928	1172.37622	509.236023	165.987473
4	0.02	648.448669	394.47641	374.582703	1183.9093	507.834015	165.624115
5	0.03	642.151794	391.651184	386.356476	1188.98377	507.623993	166.174973
6	0.04	637.668152	388.6492	398.458832	1187.30811	507.783752	166.772552
7	0.05	635.017273	385.671906	410.345551	1177.36768	508.826508	168.520859
8	0.06	633.783752	383.448883	421.169525	1159.53552	508.317078	169.12796
9	0.07	633.225037	384.157867	430.201324	1139.45984	507.705688	166.808945
10	0.08	633.969604	386.870789	438.143524	1120.10169	507.699463	167.657333
11	0.09	636.643188	389.45874	445.274719	1101.60803	507.749084	170.511307
12	0.1	641.288696	390.511597	451.550018	1083.43897	507.750702	172.95311
13	0.11	647.33667	389.950348	456.702667	1064.58814	507.789978	173.536316

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 2. 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.

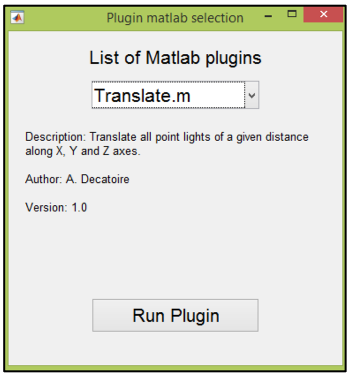


Description
None

Author
Unknown

Version:
Unknown

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



Translate.m

Description: Translate all point lights of a given distance along X, Y and Z axes.

Author: A. Decatoire

Version: 1.0

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 velocitv 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

s7.2. 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:

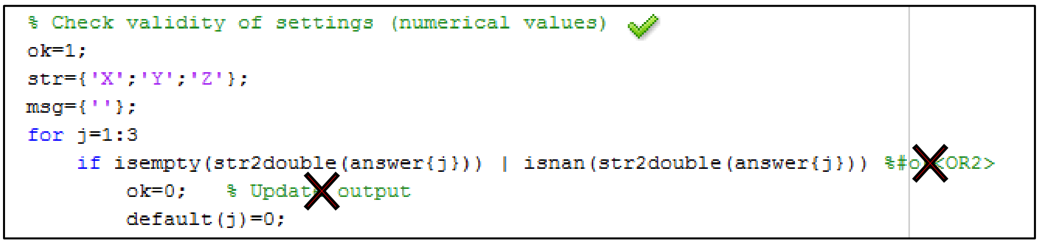
  function Outputs=Name_Of_Plugin(Inputs)

  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;    % Update 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 2: 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 for markers: 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). For example:

Inputs.Markers.R_Heel
Inputs.Markers.L_Heel
Inputs.Markers.R_Foot …

- The name of the markers with the pre-string “V_” for velocity and “A_” for acceleration for scalars: 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

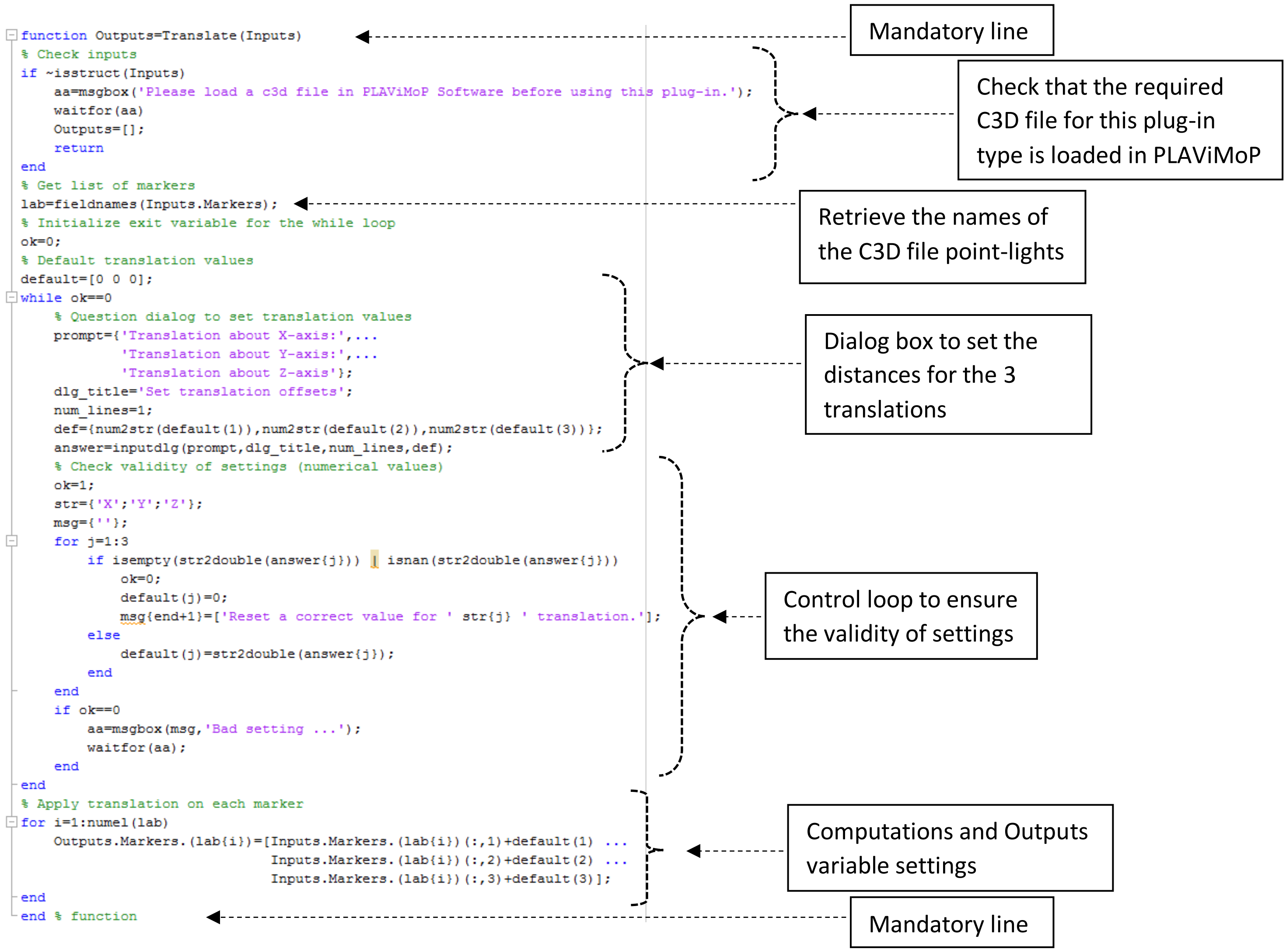
Inputs.Scalars.V_R_Foot …The name of the markers with the pre-string “NV_” for velocity’s norms and “NA_” for acceleration’s norms for analogs: these data are stored in the Inputs variable and send to the plug-in function as n x 1 matrix (where n is the number of frames). For example:

Inputs.Analogs.NV_R_Heel
Inputs.Analogs.NV_L_Heel
Inputs.Analogs.NV_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 2 Software:



```
function Outputs=Translate(Inputs)
% Check inputs
if ~isstruct(Inputs)
    aa=msgbox('Please load a c3d file in PLAViMoP Software before using this plug-in.');
    waitfor(aa)
    Outputs=[];
    return
end
% Get list of markers
lab=fieldnames(Inputs.Markers);
% Initialize exit variable for the while loop
ok=0;
% Default translation values
default=[0 0 0];
while ok==0
    % Question dialog to set translation values
    prompt={'Translation about X-axis:',...
            'Translation about Y-axis:',...
            'Translation about Z-axis'};
    dlg_title='Set translation offsets';
    num_lines=1;
    def={num2str(default(1)),num2str(default(2)),num2str(default(3))};
    answer=inputdlg(prompt,dlg_title,num_lines,def);
    % 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=0;
            default(j)=0;
            msg{end+1}=['Reset a correct value for ' str{j} ' translation.'];
        else
            default(j)=str2double(answer{j});
        end
    end
    if ok==0
        aa=msgbox(msg,'Bad setting ...');
        waitfor(aa);
    end
end
% Apply translation on each marker
for i=1:numel(lab)
    Outputs.Markers.(lab{i})=[Inputs.Markers.(lab{i})(:,1)+default(1) ...
                              Inputs.Markers.(lab{i})(:,2)+default(2) ...
                              Inputs.Markers.(lab{i})(:,3)+default(3)];
end
end % function
```

Mandatory line

Check that the required C3D file for this plug-in type is loaded in PLAViMoP

Retrieve the names of the C3D file point-lights

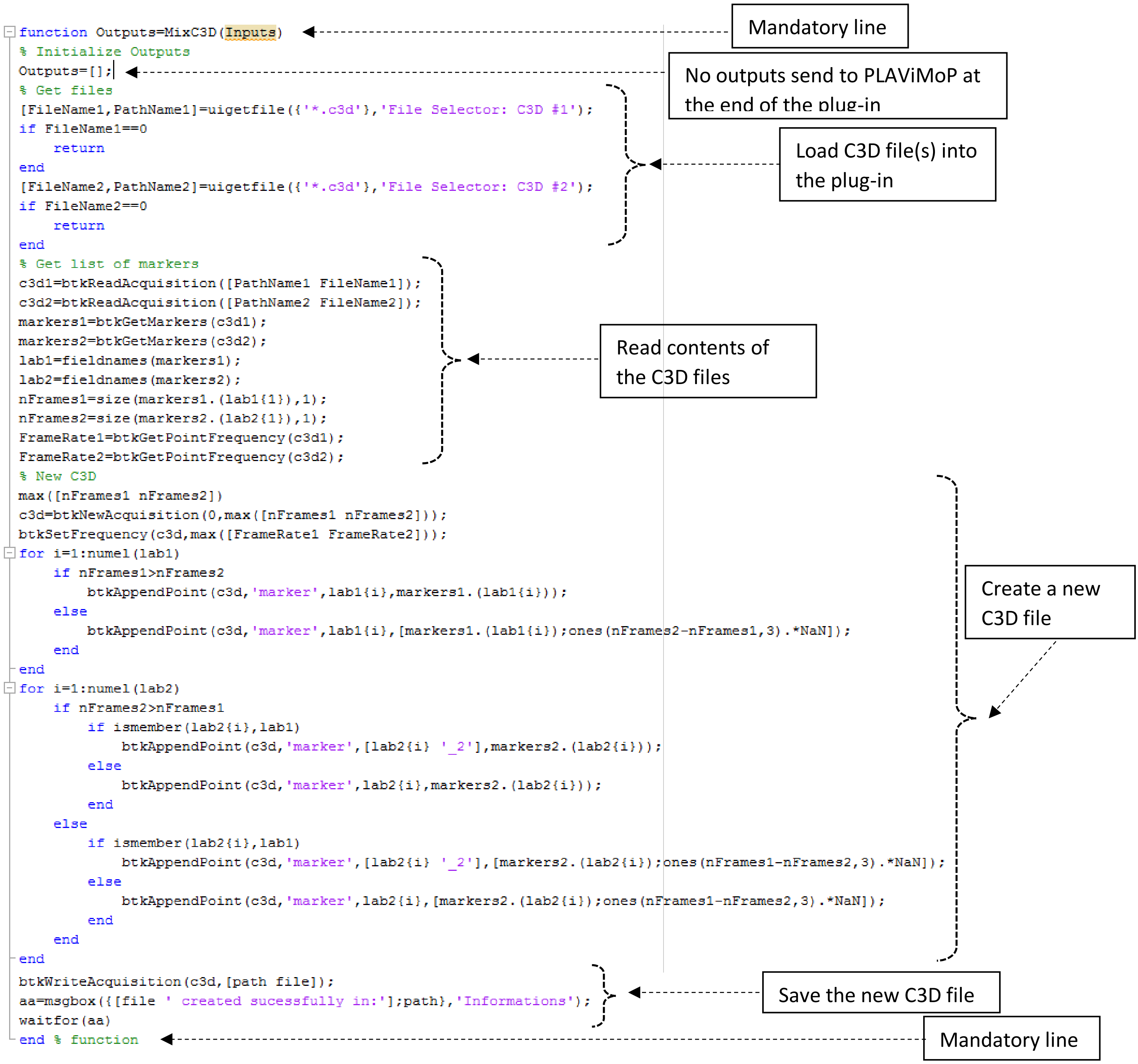
Dialog box to set the distances for the 3 translations

Control loop to ensure the validity of settings

Computations and Outputs variable settings

Mandatory line

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