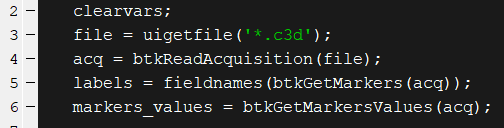
This file will describe how the work of So [1] have been implemented in matlab.

The starting point is the .c3d file where the XYZ coordinates of an acquisition are stored. It is assumed that the file does not contain missing samples.

1. Code
   1. Data extraction

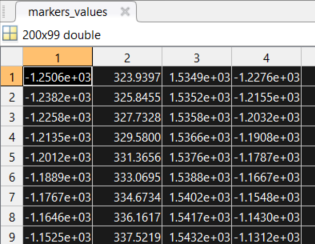


These lines are used to extract the X,Y and Z coordinates of each marker, as well as their label. As the principle of the algorithm is to detect events based only on cinematic data, it will be the only information needed.

The labels are stocked in a cell array on a single column.

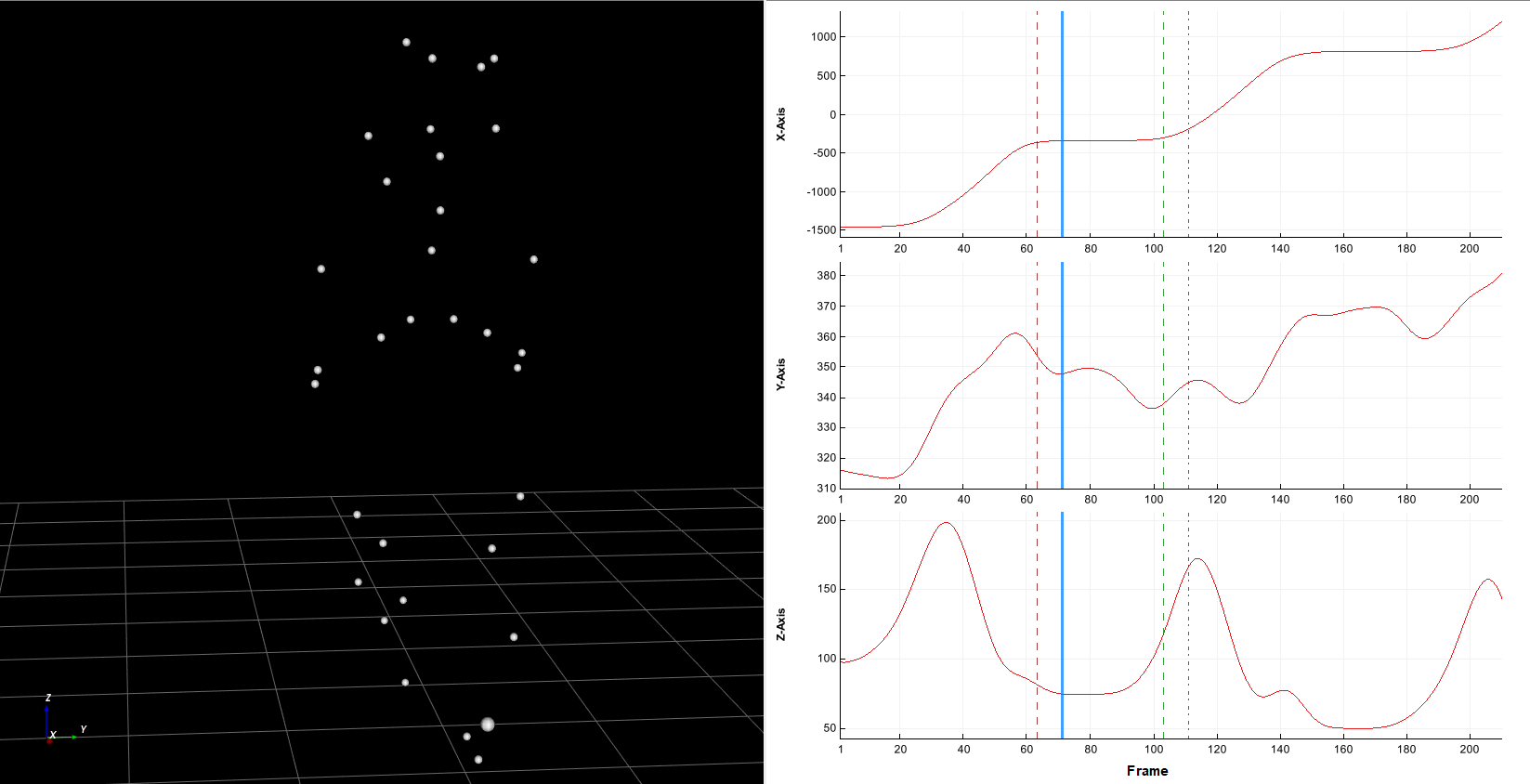


The marker values are stocked in a double array.



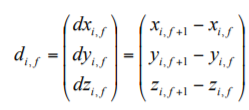
All the coordinates for all the markers for all the frames are stored in this array and are organized this way:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | MARKERS COORDINATES | | | | | | | | |
|  | 'LFHD' | | | 'RFHD' | | | 'LBHD' | | |
| FRAMES | X | Y | Z | X | Y | Z | X | Y | Z |
| 1 | -1250,6 | 323,9 | 1534,9 | -1227,6 | 240,1 | 1539,1 | -1376,8 | 344,4 | 1528,4 |
| 2 | -1238,2 | 325,8 | 1535,2 | -1215,5 | 241,8 | 1539,6 | -1364,5 | 346,7 | 1528,3 |
| 3 | -1225,8 | 327,7 | 1535,8 | -1203,2 | 243,5 | 1540,4 | -1352,2 | 349,1 | 1528,7 |
| 4 | -1213,5 | 329,6 | 1536,6 | -1190,8 | 245,1 | 1541,5 | -1339,8 | 351,3 | 1529,4 |
| 5 | -1201,2 | 331,4 | 1537,6 | -1178,7 | 246,7 | 1542,8 | -1327,5 | 353,5 | 1530,3 |
| 6 | -1188,9 | 333,1 | 1538,8 | -1166,7 | 248,3 | 1544,3 | -1315,1 | 355,6 | 1531,5 |
| 7 | -1176,7 | 334,7 | 1540,2 | -1154,8 | 249,7 | 1546,0 | -1302,8 | 357,6 | 1532,8 |
| 8 | -1164,6 | 336,2 | 1541,7 | -1143,0 | 251,1 | 1547,8 | -1290,6 | 359,4 | 1534,2 |
| 9 | -1152,5 | 337,5 | 1543,2 | -1131,2 | 252,3 | 1549,5 | -1278,4 | 361,0 | 1535,6 |

The data visualized in MOKKA, where the position of each marker can be seen as well as the coordinates of a single ankle marker along time. 

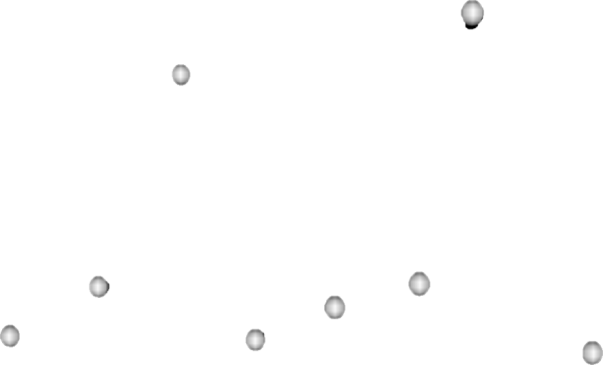
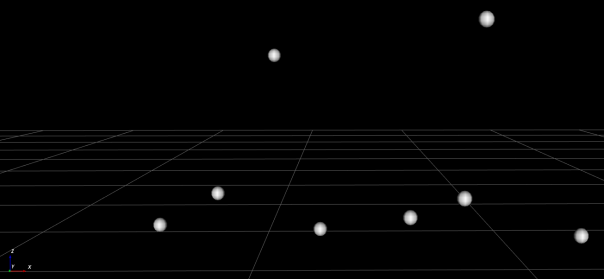
* 1. Displacement vector

The first step of the algorithm of So is to compute the displacements vectors.



With the displacement vector of the marker i in frame to the next frame. It forms an instantaneous velocity of marker i in frame f to the next frame.

The principle is illustrated in the following picture. The markers coordinate at frame are in green, the marker coordinates at frame are in white. The difference between the coordinates of the two frame results in the vectors drawn in red. (the focus is made on the markers of the foot).

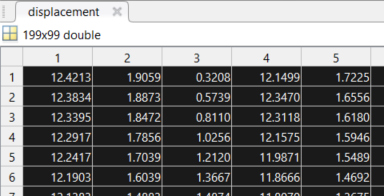


In MATLAB it is translated this way:

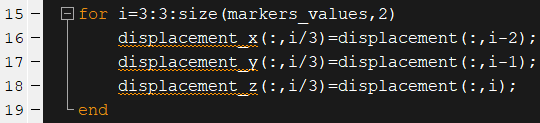


The first line preallocate the needed space for the array. Then, with each row of the array corresponding to the X,Y,Z coordinates of all the markers for a frame, as shown in the date extraction part, the only action needed is to do row(i+1)-row(i).

The result:



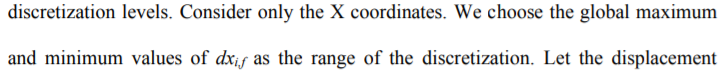
The displacement are then sorted by X,Y and Z axis for the next steps:

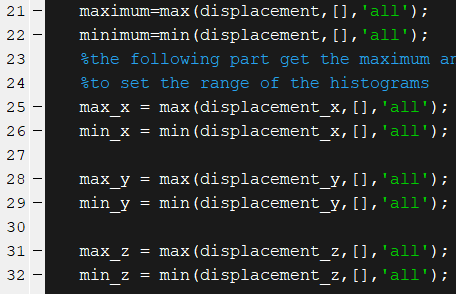


* 1. Displacement histogram

The following step is to calculate the displacement histogram according to a number of discretization levels. The X,Y and Z coordinates will now be considered separately, but the calculation will also be carried out for the whole array of displacements.

To create the displacements histogram, the maximum and minimum of the displacement for each axis are computed. This max/min is found among all the frame and all the markers.



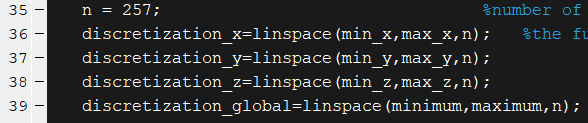


Concerning the formulation of this phrase : “We choose the global maximum and minimum values of as the range of the discretization”, it could be understood that the maximum and minimum have to be found specifically for each frame, hence meaning that each frame would have different maximum and minimum value. As these maximum and minimums will be used to create a histogram and to calculate probabilities, it sounded more logical to choose shared max and mins for all frames. This reflection can be extended to the dimensions too: should the displacements along the X, Y and Z axis have the same max and min? For this reason the algo will also be carried out with this setting.

This will result as something looking like this example:

|  |  |
| --- | --- |
| X | |
| min | max |
| -2 | 7 |
| Y | |
| min | max |
| 2 | 11 |
| Z | |
| min | max |
| -4 | 8 |
| ALL | |
| min | max |
| -4 | 11 |

Then the space between these maximum and minimum is evenly distributed into intervals. The number of intervals is stored in the variable . This number of interval may have an impact in the result of this algorithm.



This will result into an array similar to this one:

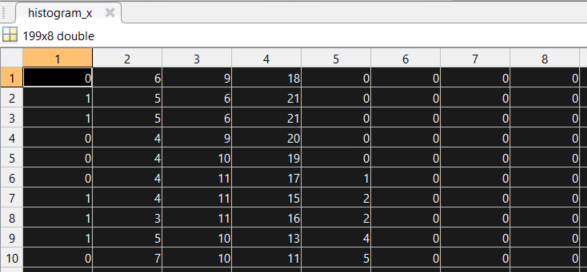
|  |  |  |  |
| --- | --- | --- | --- |
| X | | | |
| min |  |  | max |
| -2 | 1 | 4 | 7 |
| Y | | | |
| min |  |  | max |
| 2 | 5 | 8 | 11 |
| Z | | | |
| min |  |  | max |
| -4 | 0 | 4 | 8 |
| ALL | | | |
| min |  |  | max |
| -4 | 1 | 6 | 11 |

Here we can clearly observe that the size of the intervals are not the same among all dimensions.

Once the intervals are set, the histograms for each frames are computed. To do so, the built- in MATLAB function histcounts is used. It will count how many elements of the displacements array are in each intervall of the linearized space computed the step before.



The resulting array is organized like the following:



Here the number of level is set to 8 to demonstrate the functionning of the function. The following figure gives a visual representation of the classification.

Frame

Minimum

Maximum

←Intervals→

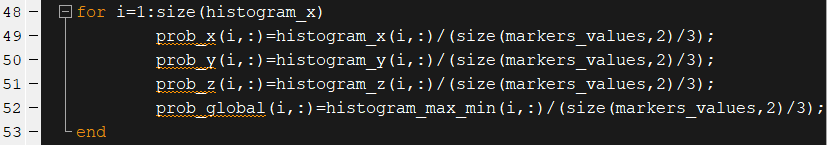
In red: Displacement vectors

With this step this part of the publication is reached: “Let the displacement histogram be the n discretized outcome of in frame ".

* 1. Probabilities

With the histogram previously computed, it is possible to calculate the probabilities for a marker to have a displacement vector in a given interval. In the publication, the probabilities are simply computed this way: “, where .”

The corresponding MATLAB code is simple.



The histogram is simply divided by the number of markers.

* 1. Joint probabilities

Now comes into play the “entropy” part of the publication. The algorithm finds key poses by looking at the quantification of the mutual information between the frame and , named . This mutual information gives a measure of the overlap information carried in X and Y. Hence a low will indicate a high probability of dissimilar velocities between frames, whereas a high will indicate a high probability of similar velocities.

To compute, the first step described in the publication is to express the joint probability of the discretized displacements vector in the form of a matrix, with the number of levels of discretization. This matrix is called , with the element in position representing the probability of a marker to be in the interval at frame and at the interval at frame .

To compute this matrix, a special function has been created in MATLAB.



To work, this function needs the histograms, the displacements, and the linearized space, called discretization. The labels of the markers are also needed, as they will be treated individually.

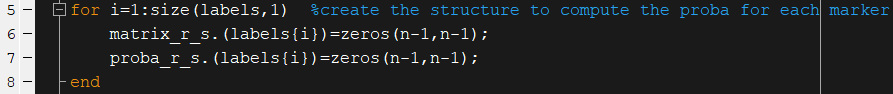
Then the total number of event, i.e. each marker displacement, is calculated as it will be needed to compute the probabilities. It is only needed to multiply the number of frames-1 by the number of markers to find this number.



Then the C matrix is declared.

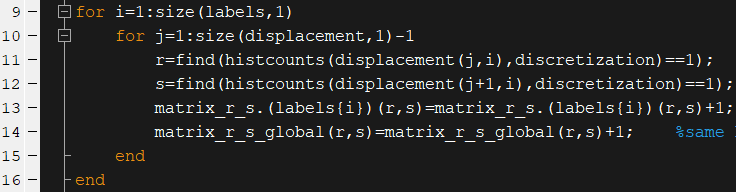


Structures are also created to store individual C matrix for each marker. It is not obligatory to complete the algorithm, but can give valuable information.



The get the C matrix filled with the wanted probabilities, the first steps is to sum how many times markers goes from interval to .

To do so, the following lines are used:



The loop will go through each marker and each frames. The lines 11 and 12 will find the position of the displacement vector in the histogram. The histcounts function will put a 1 in the good interval and the function find is used the get the position of the 1, i.e. in which interval it is. Then a 1 is added in the correct position of the matrix, for the each marker individually at line 13 and for all the markers at line 14.

histcounts

find

Interval=1

Displacement

Or in tables:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | X | | | | | | | | |
| Frame | interval 1 | | | interval 2 | | | interval 3 | | |
| f | -2 | X | 1 | |  | 4 | |  | 7 |
| f+1 |  | X |  |

Position in the matrix:

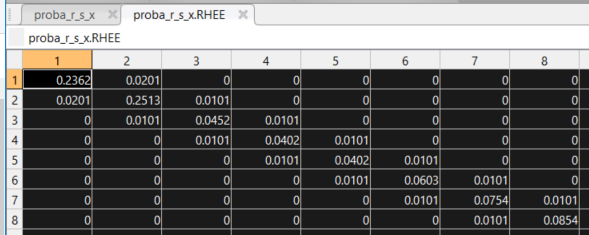
Another example:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | X | | | | | | | | |
| Frame | interval 1 | | | interval 2 | | | interval 3 | | |
| f | -2 |  | 1 | |  | 4 | | X | 7 |
| f+1 |  | X |  |  |

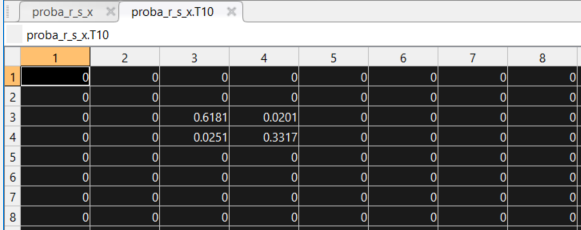
Once the matrices are full, the next step is to divide them by the correct number: the number of frame for individual markers and by the total number of events for the whole set.



For individual markers, the C matrix is looking like the following table:



Here is the C matrix for the right heel marker, and it is possible to see that the data is distributed along the diagonal. It is logical as even if there are some peaks of acceleration during the walk, it stays gradual. Hence there are no values in the right top or left bottom corners, as it would mean that the displacement vector went from the maximum to the minimum in the time of one frame.

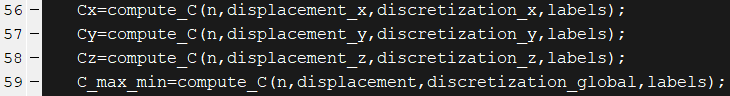


Here is the same matrix for the marker on the 10th thoracic vertebrae. The probabilities are concentrated on one square because unlike the heel marker, the movement of the top of the body is much more continuous.



This is the matrix for the whole set of markers and all the frames.

The following lines execute the described function for each dimension.

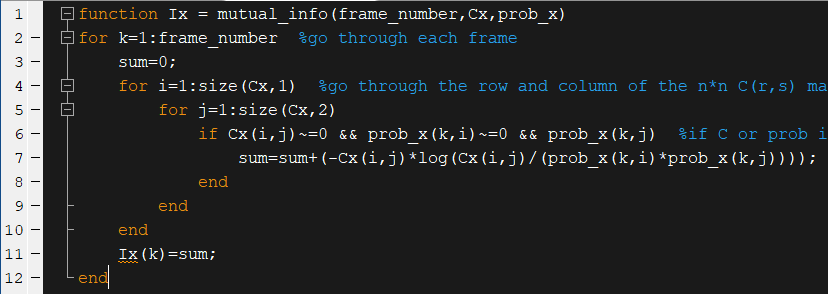


* 1. Mutual information

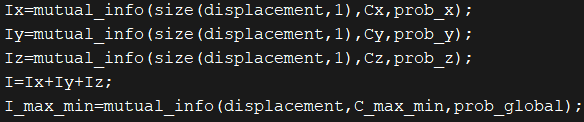
Then the mutual information must be computed, by applying the following formula:

A specific function as been coded to apply it, which needs the number of frames, the C matrix and the probabilities computed from the histogram to work.

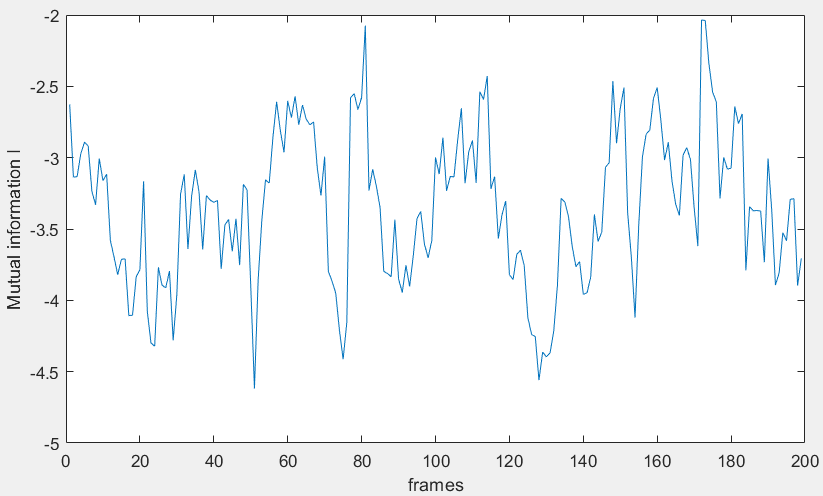
Then there are three nested loops: one to go through all the frames and two other to go through the rows and columns of the C matrix. Then the values are tested to make sure neither of the probabilities or the value in the C matrix is equal to zero, otherwise it would generate non-rational values. The result is a value of mutual information for each frame of the measurement.



Then the global mutual information is simply the sum of the mutual information for the three dimensions.



The resulting can be plotted



* 1. Keyposes detection

Key poses are considered as a change of velocity in the motion. A low implies a high probability of dissimilar velocity, hence a high probability of key poses. The algo will look for minima in .

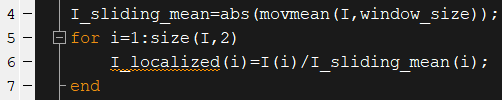
To do so a specific function has been developed. The input data is composed of the mutual information , the window size to proceed to a sliding mean, and a threshold.



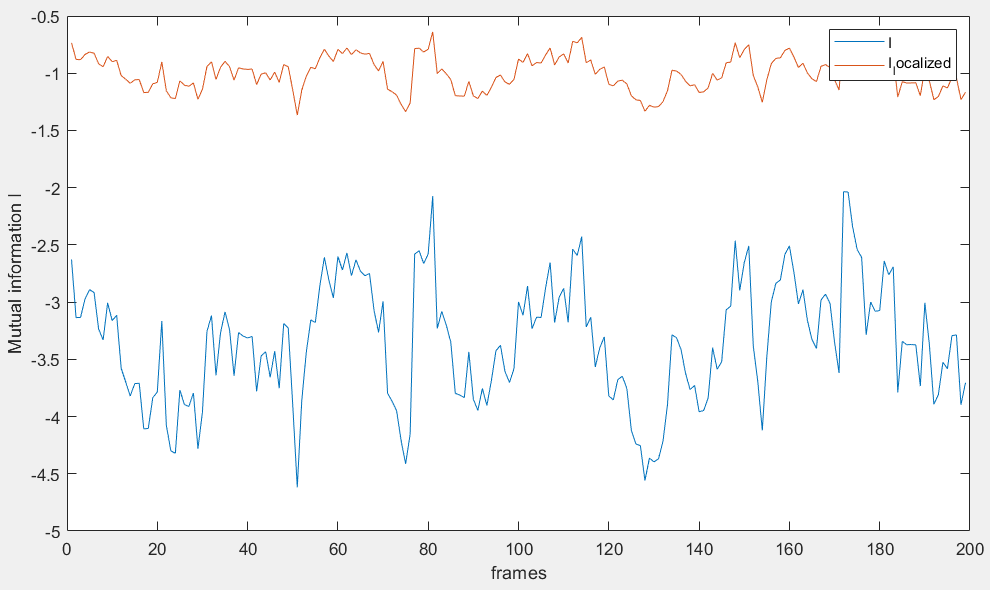
The first step of the key pose detection described in the publication is to localize the mutual information to emphasize local changes. This localized is computed as

With . form a window always centered on .

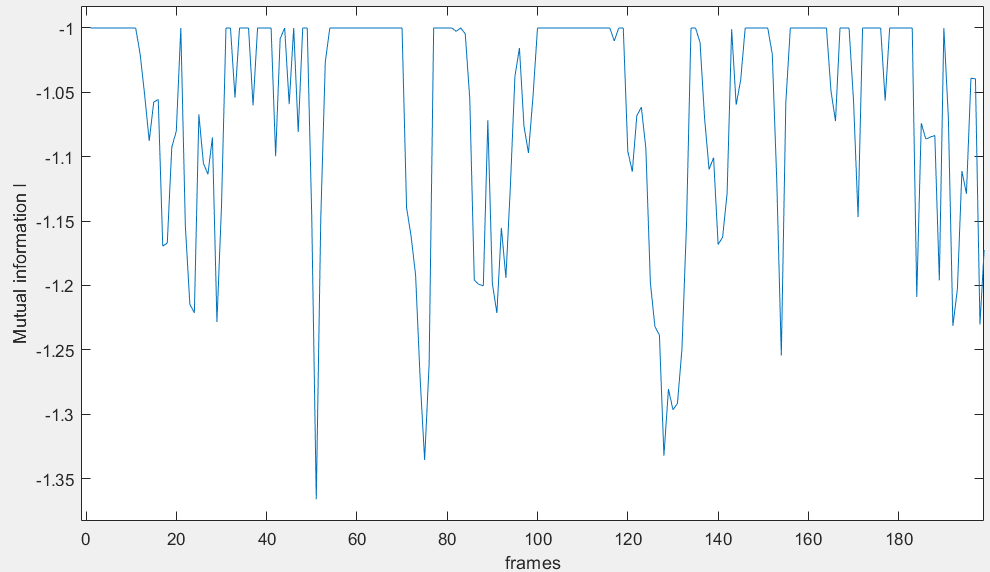
To do so in MATLAB, the sliding mean is computed around each frame and then this mean is used to divide the original. The absolute values of the moving mean are taken, because it has been observed that this sliding mean can take negative values and would inverse.



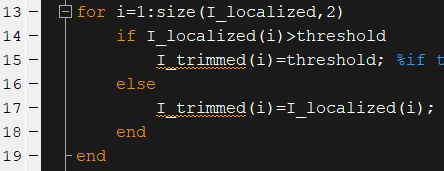
The sliding mean is computed with the built-in MATLAB function movmean. The window size is one of the settings that will have an impact on the result. For the following figure the windows size have been set to 60. The following picture put the ‘raw’ and localized mutual information side to side.



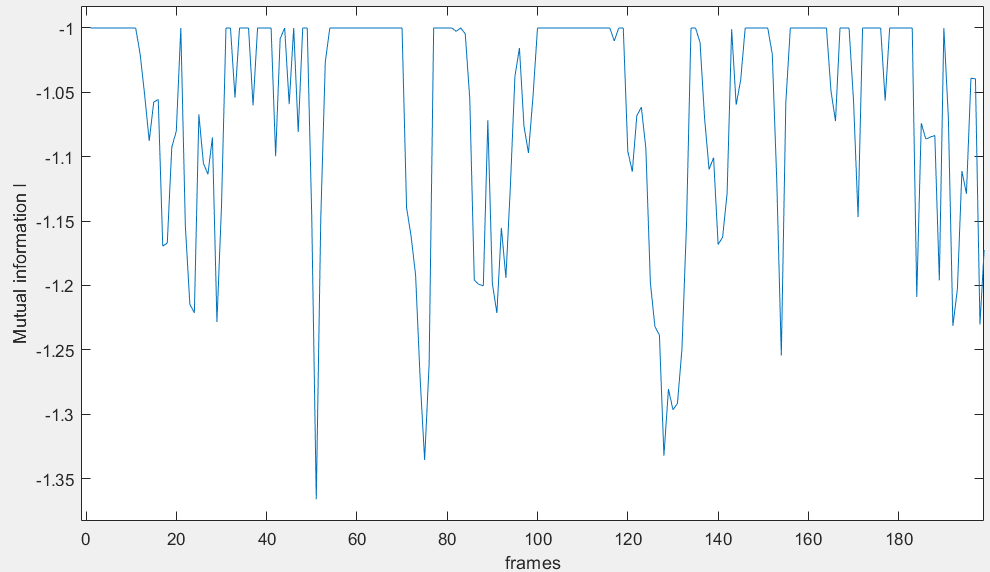
Then a threshold is applied to this localized data. To do so in MATLAB, all values above the given threshold are reduced to the threshold itself. In the following figure, the threshold has a value of -1.



To trim the date the following lines of code are executed.



Then the local minima of each through are found. A through correspond to the following figure:

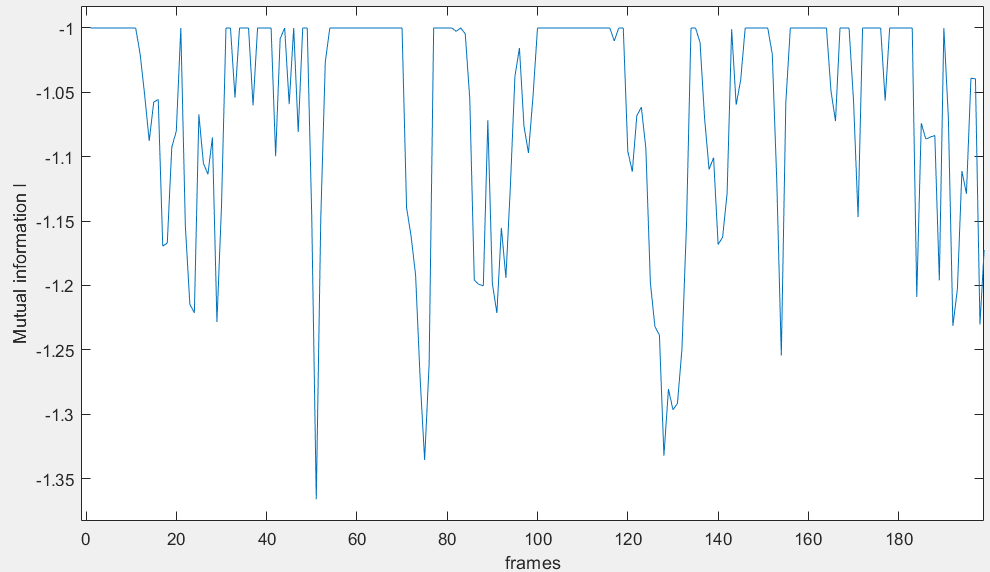
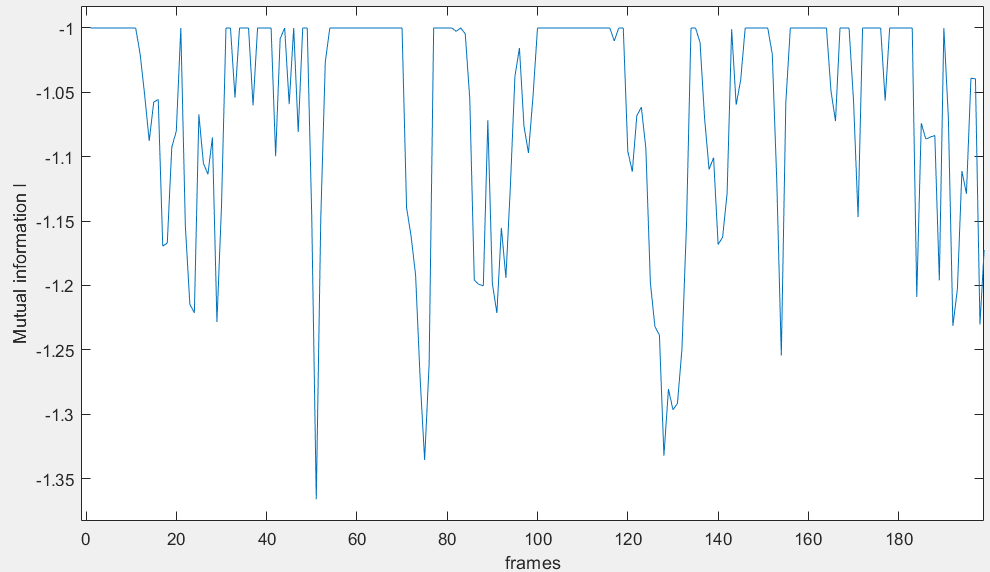


Through 1

2

3

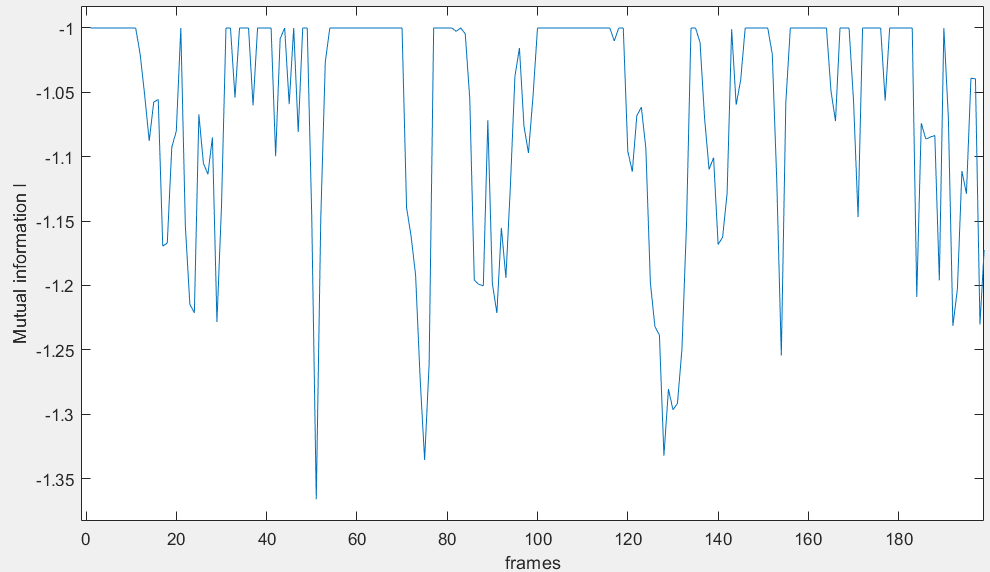
The findpeak function could have been used but this function returns every peak, whereas the algo must only look for the lowest value in the through:



findpeak

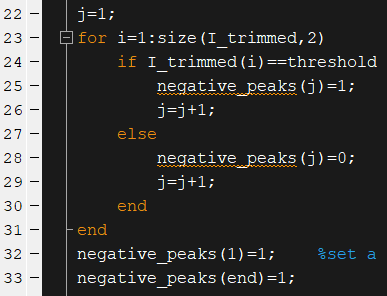
What is wanted

So to find these minimal values, the trimmed file is converted in a ‘binary’ file. When the value of I is equal to the threshold it is converted into a 1, otherwise it is converted into a 0.

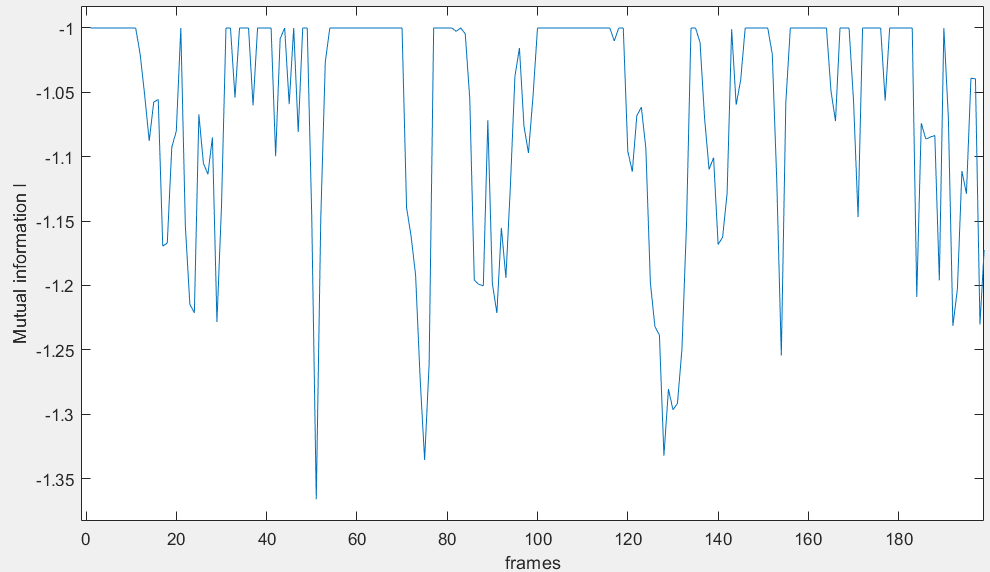


1100000011110100000000000000001111111

These lines of code execute the conversion:



The next part of this function will look for the ‘start’ and ‘stop’ of the functions, to find at which frame the through begins and at which frame it ends.

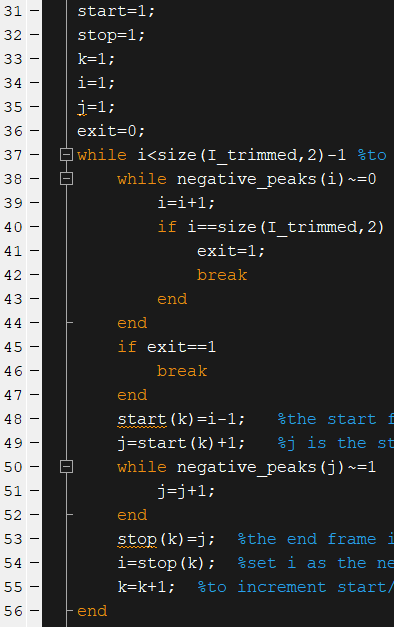


1100000011110100000000000000001111111

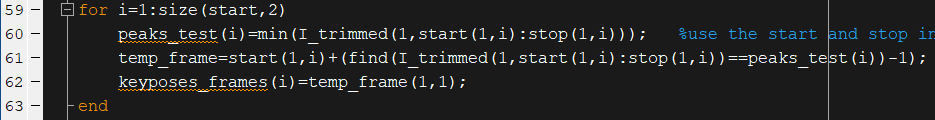
Start=green

Stop=red

The code:



Then having found the limit of each through, it is possible to look for the minimum value in this through.



The only part that may generate an error is that once the minimal value is found, the frame of the peak is traced by using the ‘find’ function. If in the through the minimal value occurs twice, it may generate inconsistencies.

The identified peaks are the key poses frames.

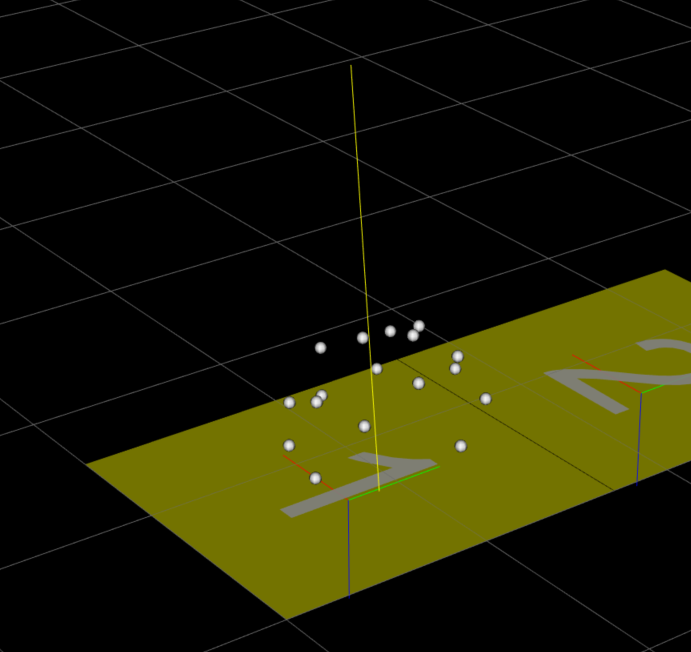
1. Testing

Specific files have been created in order to test the algorithm. At first these files were used to synchronize the motion capture system qualisys and the spatio-temporal system Optogait, but can also be used for the purpose of testing the algorithm.

A basketball has been covered with markers, before being dropped onto the force plate.

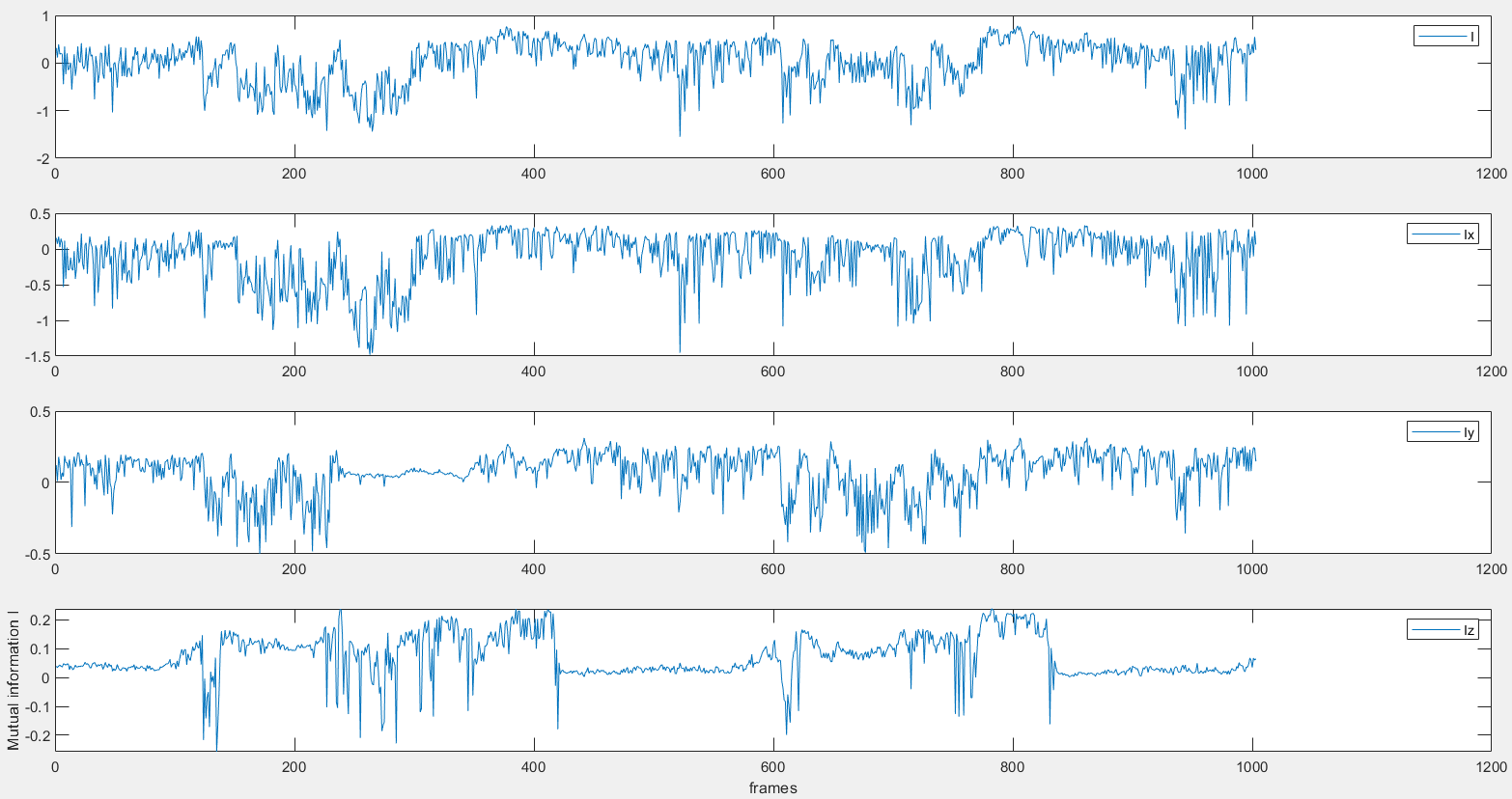


Resulting in the following files:



It was expected that the impact would be detected by the algorithm, based on the statements made in the publication: “A keypose involves a change of velocity (or a change of displacement vectors between frames) in the motion. A low mutual information If indicates a high probability of dissimilar velocity between frames, which indicates a high probability of occurrence of key pose.”.

After processing the file with 256 levels of discretization, the following results are produced:

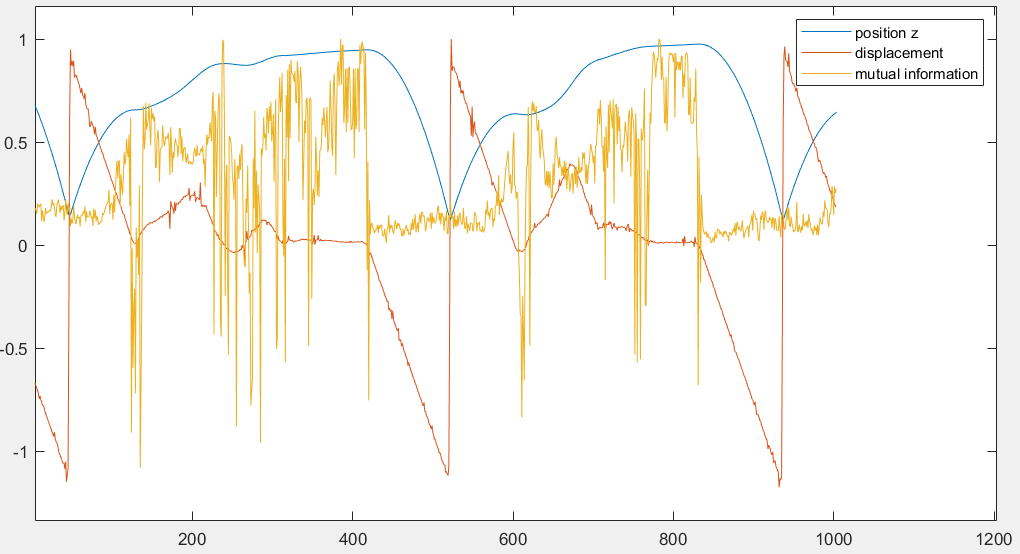


In order: I, Ix, Iy,Iz.

Each plot represents the level of mutual information between each frame for the 3 dimensions and for the sum of the three dimensions, as stated in the publication.

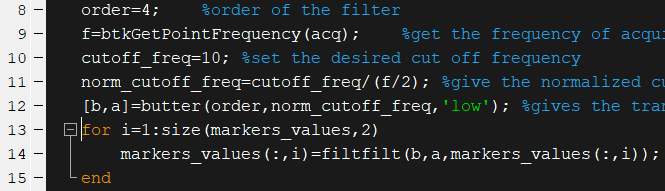
By taking a marker and overlapping its Z coordinate value, its Z displacement and the mutual information in Z, it allows visualizing the link between these variables.

* 1. Behavior

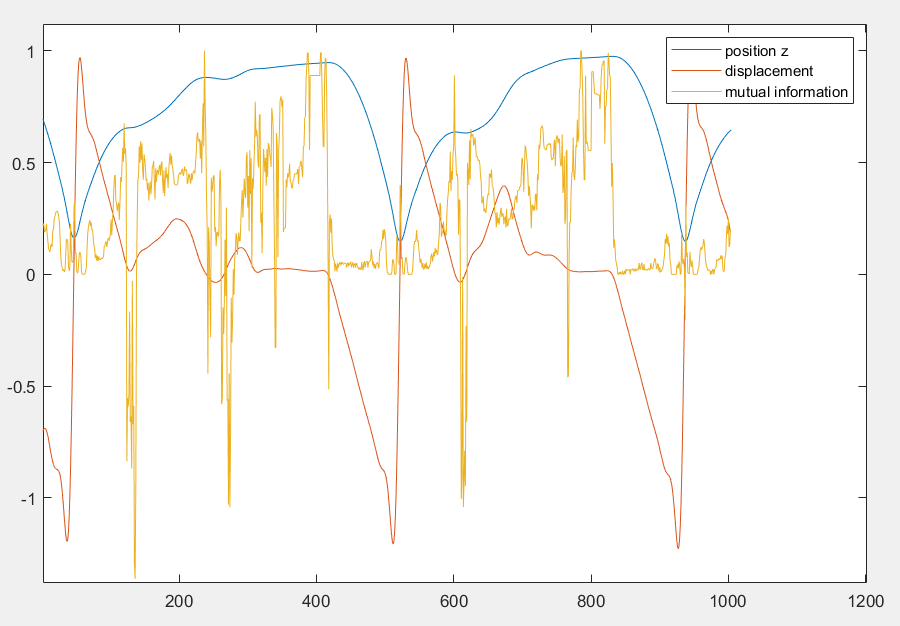


To be able to compare these curves they have been normalized by dividing them by their maximum value.

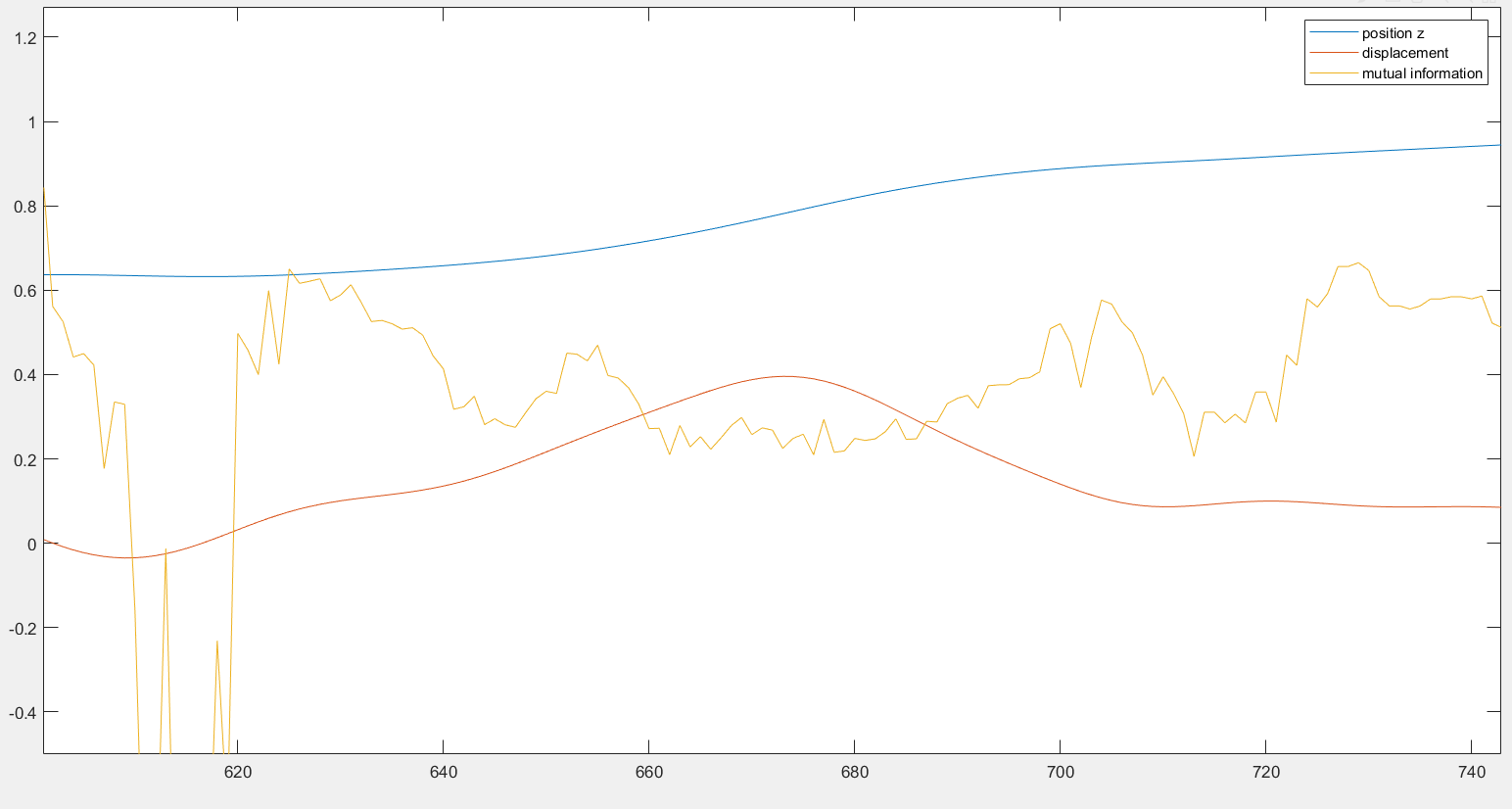
The noise visible in the mutual information curve shows that change in the velocity of the marker does have an important effect. To diminish this effect a filter can be applied. The filter chosen is a 4th order butterworth low-pass filter (10Hz).

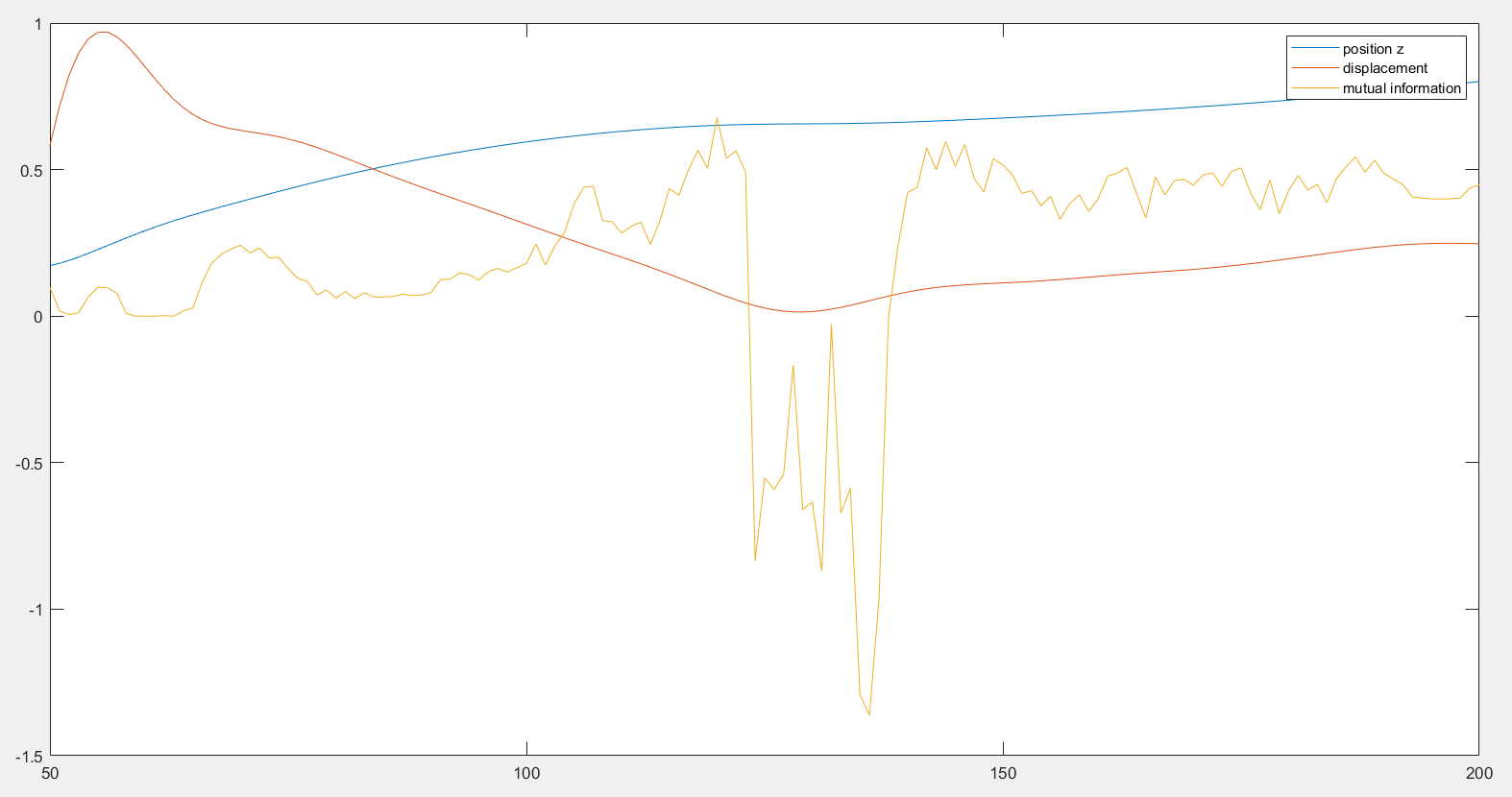


The results:

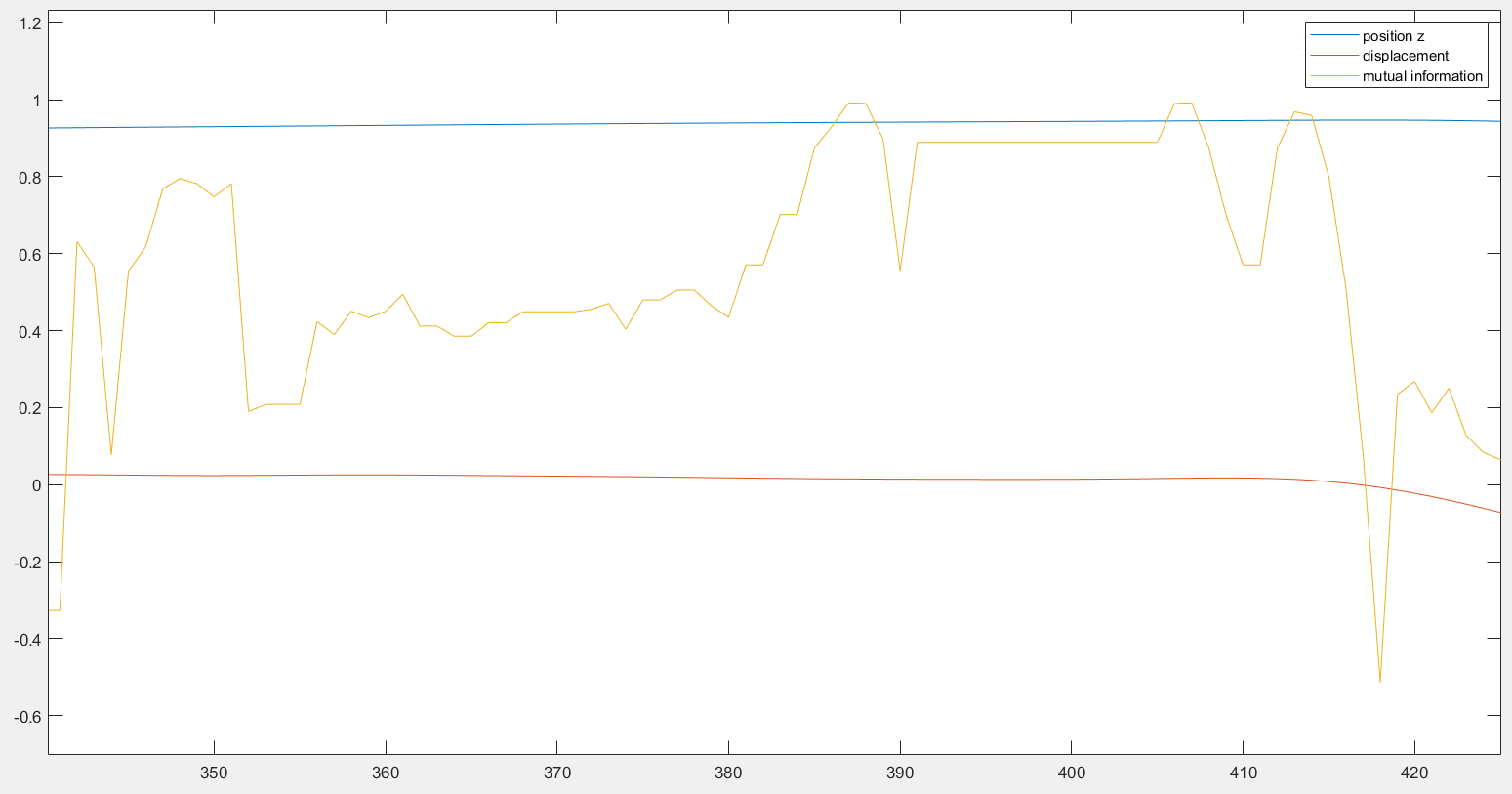


The filter does smooth the displacement curve, but the mutual information stays really sensible to change in the displacement curve. The mutual information gives results that are sometimes logical for the expectation expressed before, such as in these moments:



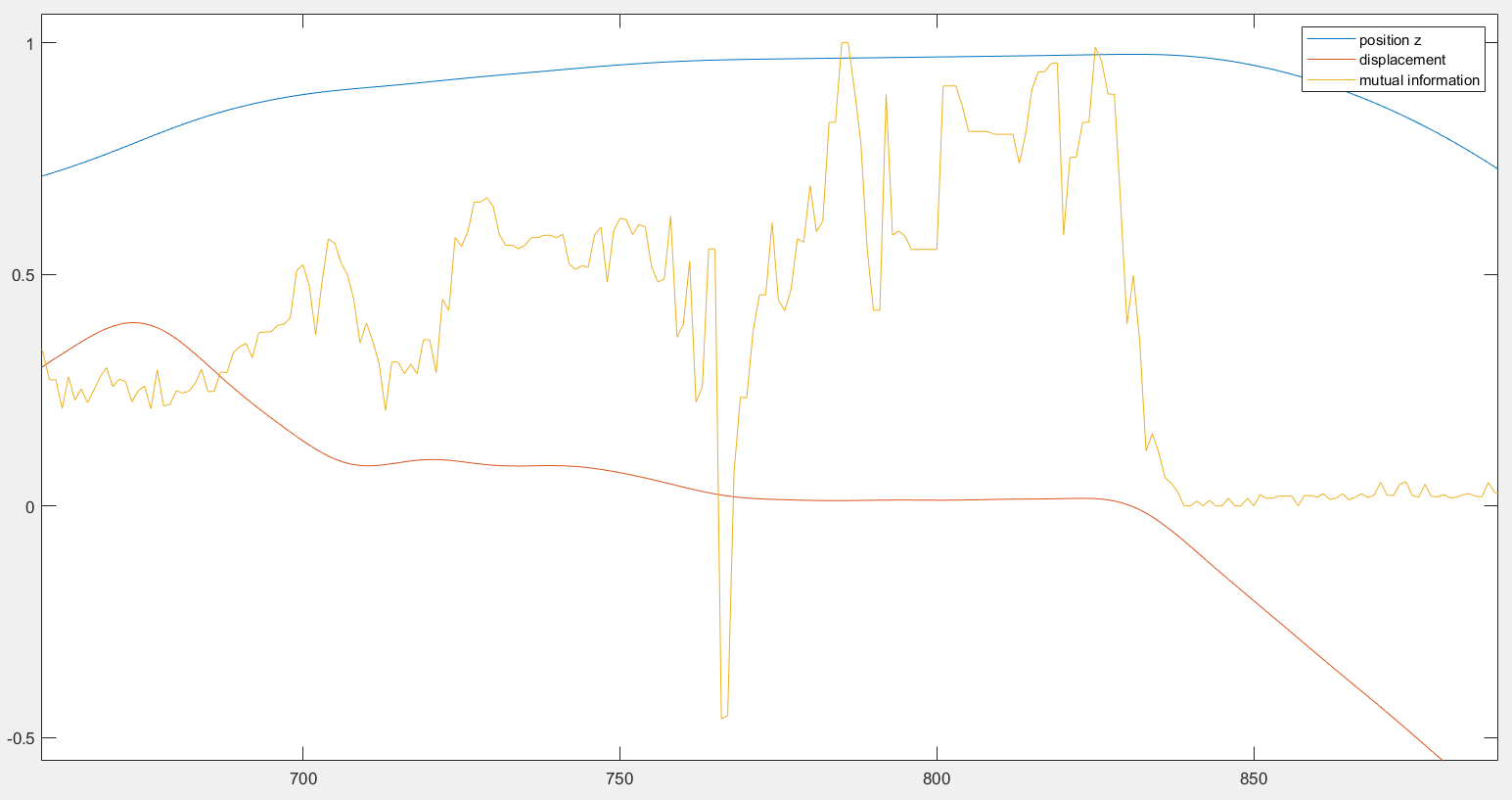


For the two previous figures, there is a visible change in speed shown by an inflexion in the displacement curve, and the mutual information value diminishes, accordingly to what is expected from the algo.

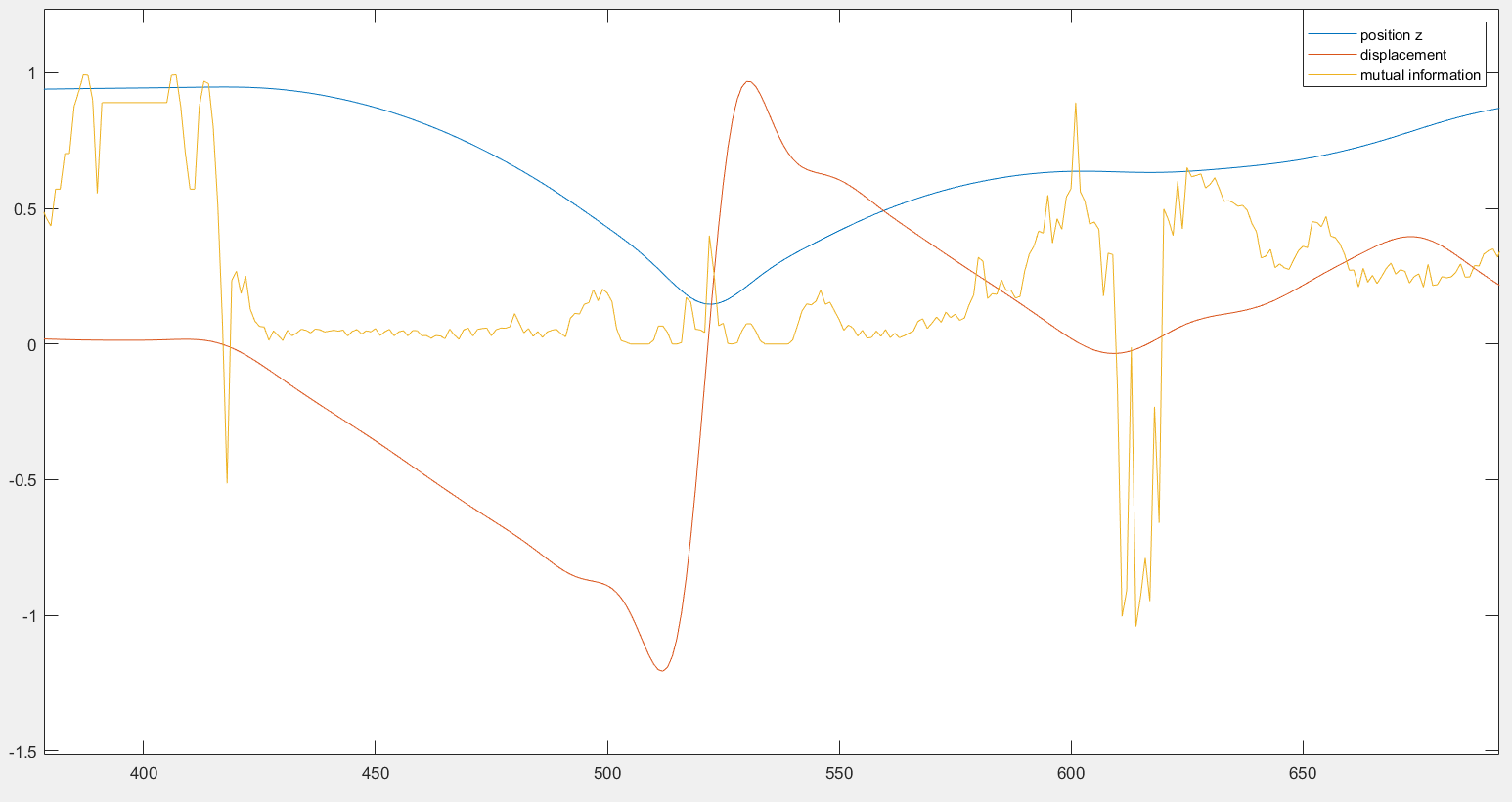


Here the displacements stay constants, resulting in a higher value of mutual information.

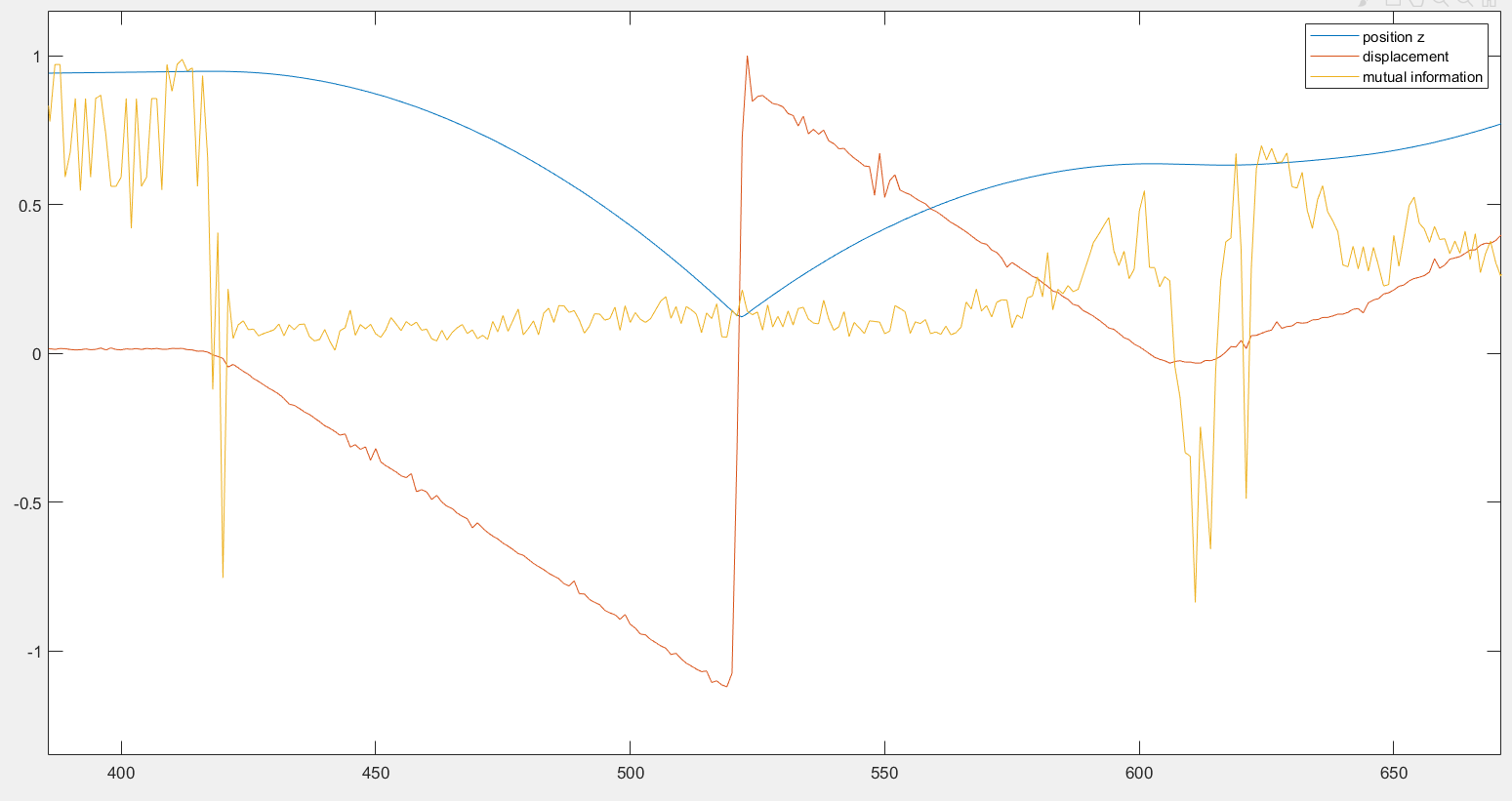
But other parts of the curves show unexpected behavior, for example a drop in the mutual information despite showing a smaller inflexion.



Another part where the algo does not behave as expected is during the impacts.



The mutual information is low but constant after the drop, which is logical as the ball accelerates towards the ground, but when the ball hits the ground and goes back away from the ground, creating an important change in the displacement, the mutual information does not decrease, and even increase slightly. The filtering processes smooth the curve and could affect the values of mutual information, but the unfiltered curve shows the same behavior:



These inconsistencies in the behavior have no explanation for the time being.

* 1. Limits of the histogram

One of the concerns expressed at the begging was the fact that the histograms for the dimensions X, Y and Z didn’t had the same limits, hence not the same intervals, maybe impacting the results.

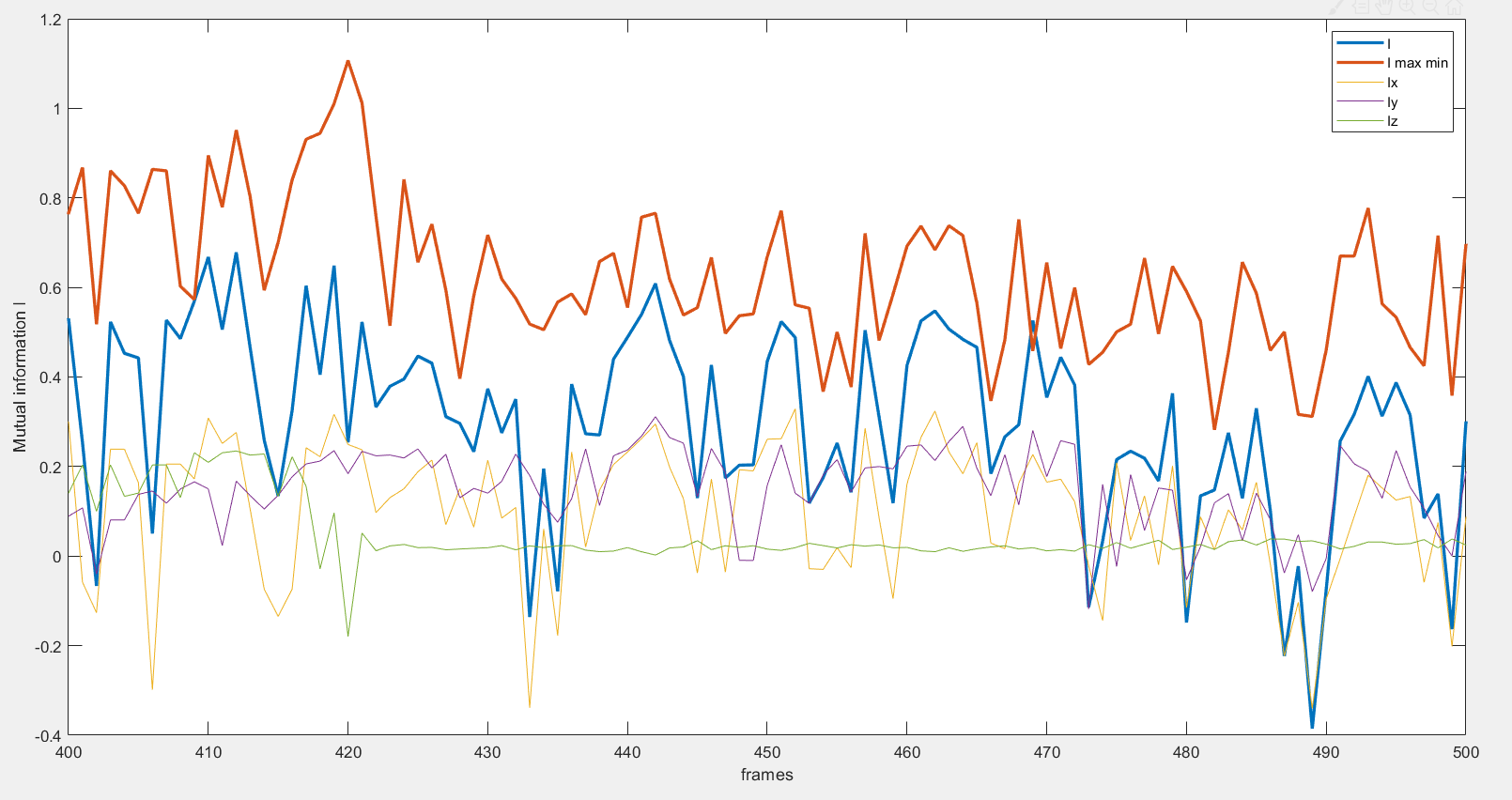
For the files studied, the limits of each dimension are the following:

|  |  |  |
| --- | --- | --- |
|  | Min | max |
| X | -20,76 | 16,55 |
| Y | -11,07 | 8,73 |
| Z | -26,30 | 38,12 |
| Global | -26,30 | 38,12 |

Resulting in the following intervals with 8 levels of discretization:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | | | | | | | | |
| min |  |  |  |  |  |  |  | max |
| -20,76 | -16,10 | -11,43 | -6,77 | -2,10 | 2,56 | 7,23 | 11,89 | 16,55 |
| Y | | | | | | | | |
| min |  |  |  |  |  |  |  | max |
| -11,07 | -8,60 | -6,12 | -3,64 | -1,17 | 1,31 | 3,78 | 6,26 | 8,73 |
| Z | | | | | | | | |
| min |  |  |  |  |  |  |  | max |
| -26,30 | -18,25 | -10,20 | -2,14 | 5,91 | 13,96 | 22,01 | 30,07 | 38,12 |
| Global | | | | | | | | |
| min |  |  |  |  |  |  |  | max |
| -26,30 | -18,25 | -10,20 | -2,14 | 5,91 | 13,96 | 22,01 | 30,07 | 38,12 |

By giving each dimension respective limits, the result is the following:



Being the sum of it is generally following the dimension having the greatest absolute value at the frame. Despite having the largest displacement, the Z dimension have the smallest impact in the total value of . This effect is decreased when the maximal and minimal value of all the dimensions are taken.

* 1. Effect of the number of intervals

