Contact Details:

Hugo Villi: [hugo.villi9@etu.univ-lorraine.fr](mailto:hugo.villi9@etu.univ-lorraine.fr); phone number: +33 06 05 25 45 65

Florent Moissenet: [florent.moissenet@protonmail.com](mailto:florent.moissenet@protonmail.com)

Team name : Laboratoire d’Analyse de la Marche et de la Posture

The algorithm presented in the following lines is based on the publication of Luo et al. [1]:

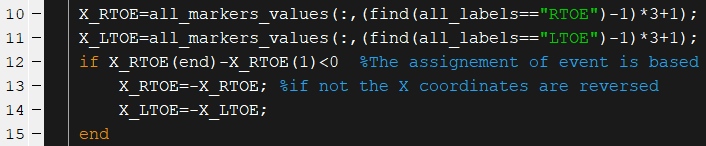
[1] Luo, G., Lei, G., Cao, Y., Liu, Q., and Seo, H., 2017, “Joint Entropy-Based Motion Segmentation for 3D Animations,” The Visual Computer, **33**(10), pp. 1279–1289.

The principle of this algorithm is to compute the joint entropy between two frames. The more changes there are between two frames, the higher is the joint entropy. The hypothesis is that gait events create changes in markers trajectories, and that these changes can be detected by looking at the joint entropy curves.

The first lines extract the needed information, the labels of the markers and the coordinates



The next loop is used the get the information of the toes markers, that will be used to determine which events is detected. It’s the get\_event function that will use this data, and the X coordinates must go towards the positive for the function to work. If it’s not the case, the opposite values of the coordinates are taken.



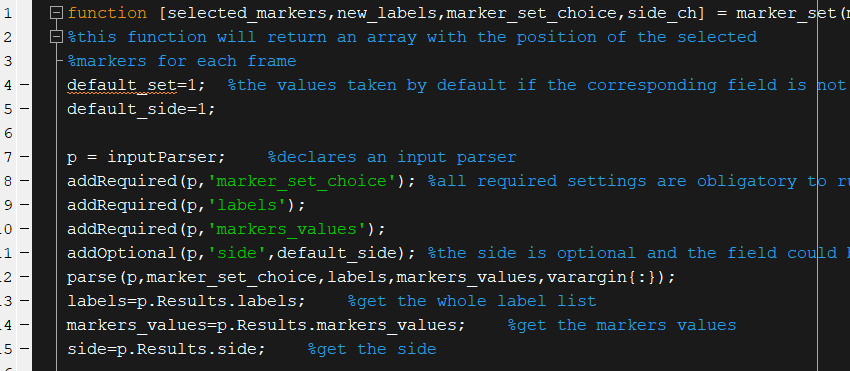
Then it is possible to choose to compute one joint entropy curve using the markers of the left and right side, or to compute two joint entropy curves, one for each side. This file presents only the first case, the second executing the same functions, but duplicate them for the left and right sides.

The set of marker used is chosen by the operator, and this information is communicated the to the marker\_set function. This function will return a new array with only the coordinates of the selected markers. 

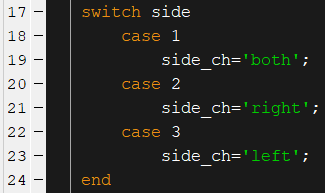
The input needed are:

* The selection of the set, that are:
  + 1=all the markers
  + 2=the minimal marker set
  + 3=the lower limbs, including the pelvis
  + 4=the lower body without the pelvis
  + 5=the shanks and foot
  + 6=the feet
  + 7=the feet and the ankle
* The label list
* The array with all the coordinates
* The choice between : both sides, right side or left side

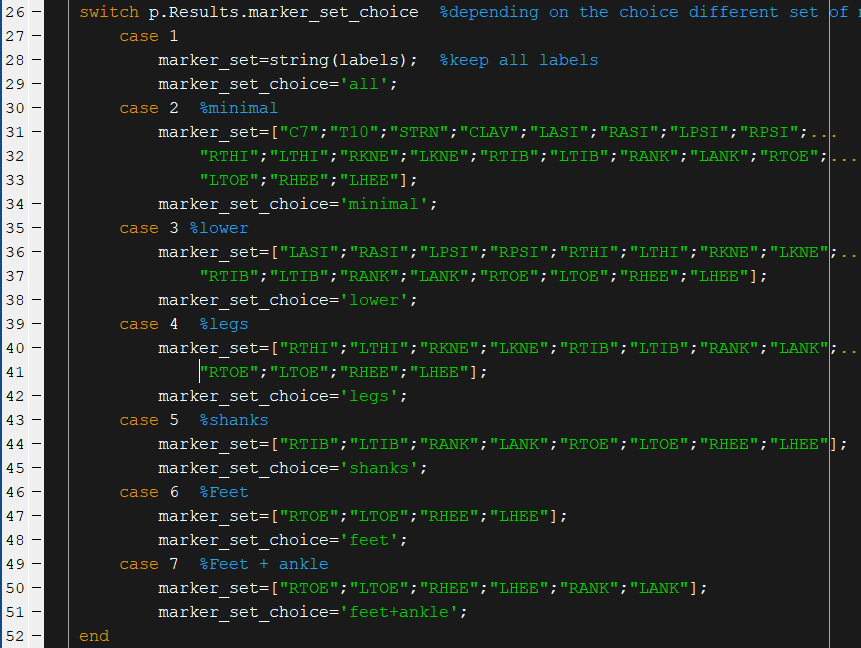
An input parser is used to get these parameters

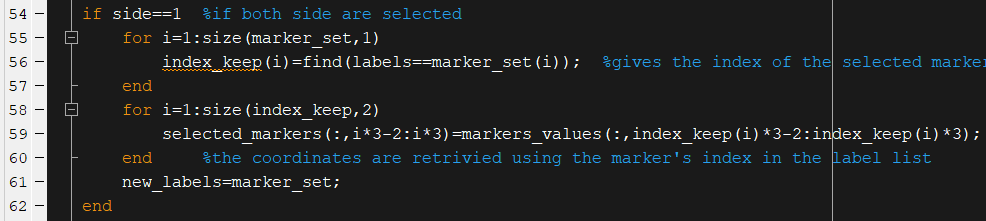


The function will return strings with the choices made for the side and set:

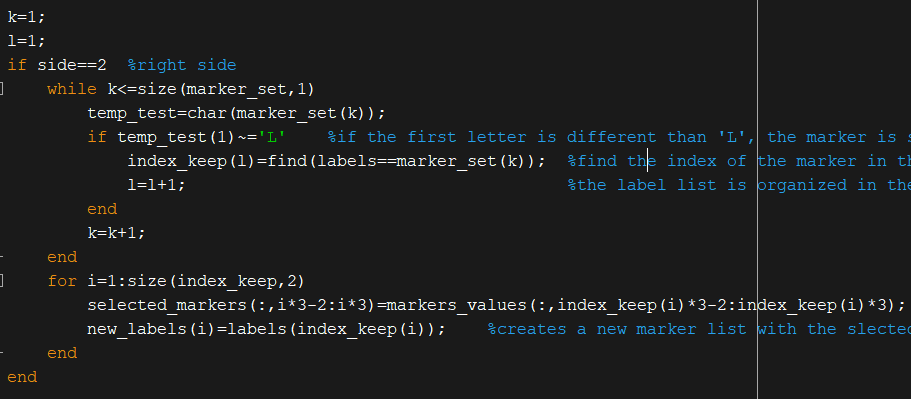


To select the markers associated with the set choice, a list with all the markers that have to be kept were created.



Next only the coordinates of the selected markers will be kept. To do so, the indexes of the labels kept are found using the find function. The index of the labels corresponds to the index of the coordinates. 

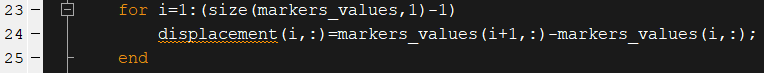
For the right and left side, a simple test based on the first letter of the labels is used. For the right side, if the letter L is not detected, the label is kept.



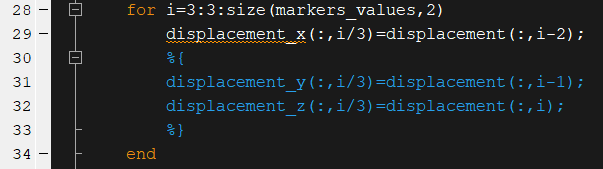
The same is made for the left side, with the R letter.

The function returns an array with only the coordinates of the selected markers.

The displacement vectors are simply computed by subtracting the frame f to the frame f+1.



It is possible to run this algorithm with all the dimensions, but it was observed that the X axis (or the axis of forward progression) gave the best results.



The two following lines are used to get the maximum and minimum of the displacement vectors.



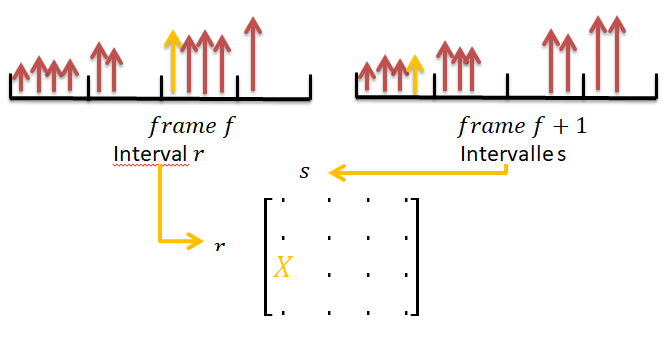
Then then distance between these maximums and minimums is divided in n intervals, called levels of discretization. The number of interval is one of the important settings of this algorithm.



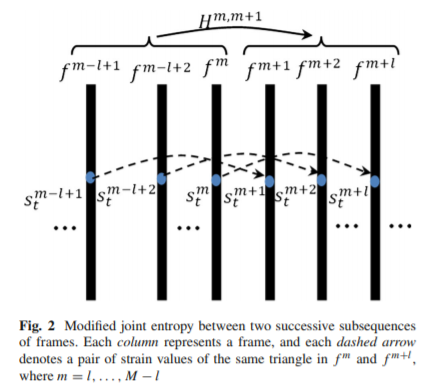
These intervals are one of the input for thee function computing the joint probability between the two frames. The others inputs are the number of levels of discretization, the displacement vectors, and the subsequence size.



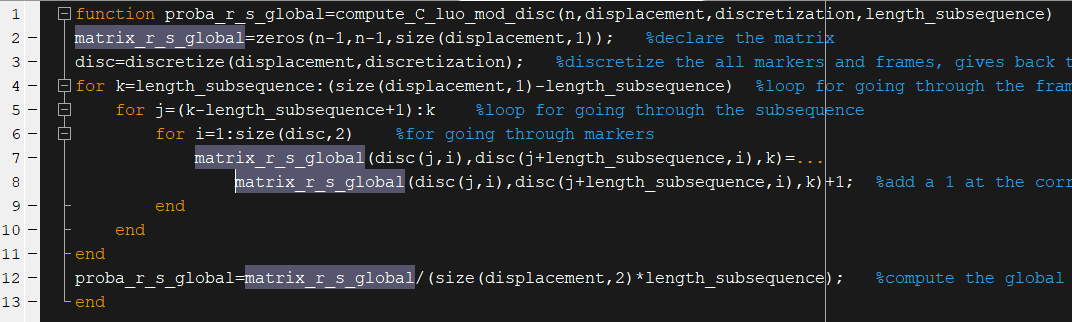
The joint probability is computed by looking at the position of the displacement vector of a marker in the intervals created earlier. The index of the interval r at the frame f and the index s at the frame f+1 are used to fill a matrix. Having a displacement vector going from the interval r to s between two frames will add a 1 in the corresponding position. This is done for all the markers, and this matrix is divided by the total number of markers to get the joint probability between the two frames.



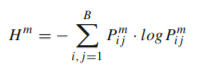
Luo et al. added another layer to this computation by taking the joint probability between two subsequences, and not only two pairs of neighboring frames.



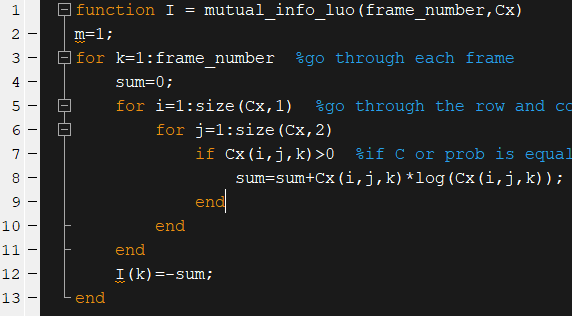
The joint probability matrix is computed the same way, but the matrix is divided by the number of markers\*size of the subsequence. This size is another important setting.



With this joint probability it is now possible to compute the joint entropy such as:



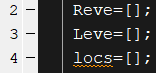
The zero values are excluded from this equation as it would results in infinite values. The function will return the joint entropy value for each frame.



Then the get\_event function can be run. It’s this function that will return the events. The inputs of this function are:

* The joint entropy curve
* The size of the subsequence
* The setting for the peak height, that will be a coefficient applied to the mean value of the curve
* The minimal peak distance
* The setting for the peak prominence, based on the maximal, minimal and mean value of the curve
* The choice for the contact or off event
* The X coordinates of the Toes for the right and left feet

The first 3 lines are used to declare the variables so the code does not return errors if any of these variables are not created during the execution.



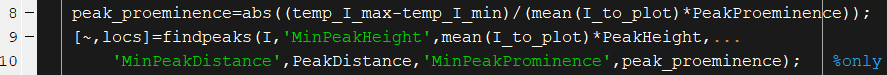
Then the joint entropy is trimmed so the 0 values at the extremities are not taken into account.



The max and min values of the curve are retrieved.



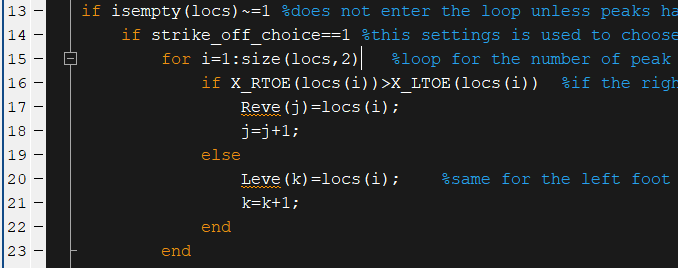
Then the findpeaks function is used to get the indexes of the peaks. The minimal height is computed by multiplying the mean value by a coefficient. The minimal prominence is computed by executing the following equation:

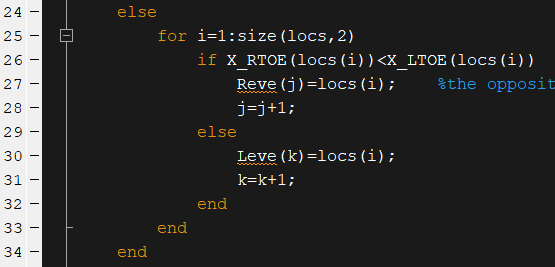


Now only the indexes of the events are retrieved, the next lines are used to determine which events correspond to the indexes.

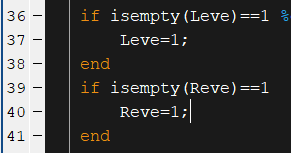
If the strike option is selected, the difference between the right and left foot strike is made by looking at the x coordinates of the toe markers. If it is the right toe marker that is in front, it is identified as a right foot strike, and the same is made for left events.

For the toe off events, the index will be assigned to the most backward foot.





To avoid error in the rest of the code, if no events were detected, the function returns events for the right and left foot with an index of one.



The execution time is of 39.39 s.



The score for foot off is: 796987.205

The score for foot strike is: 80146.8855

