Source Code Tutorial

For the paper: Efficient Unsupervised Temporal Segmentation of Motion Data

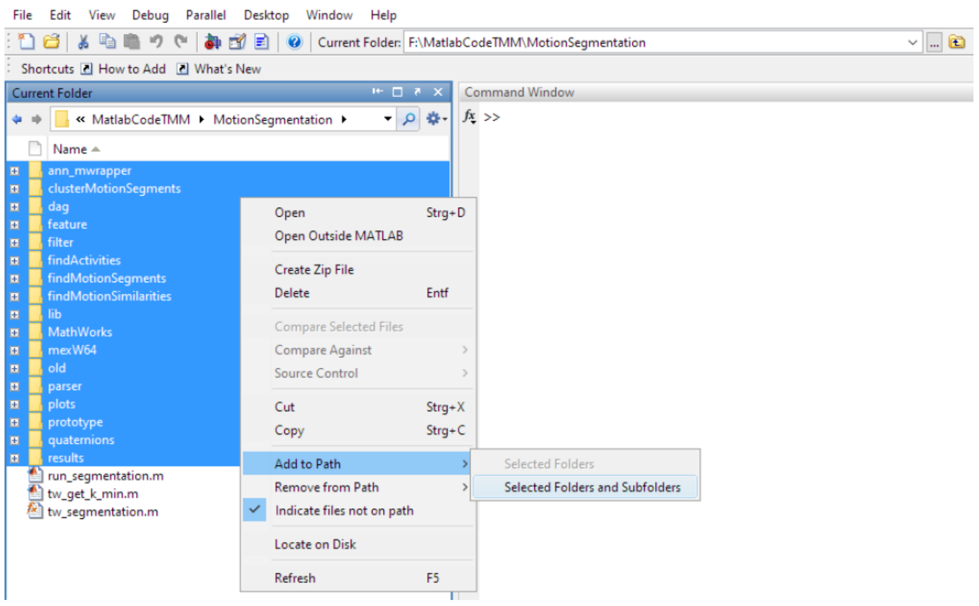
Author: Björn Krüger, Gokhale Method Institute, kruegerb.cs@gmail.com

Introduction: This document lists the steps needed to run the Matlab code to the paper "Efficient Unsupervised Temporal Segmentation of Motion Data" published in the IEEE Transaction on Multimedia. The provided source code is available for Windows 64bit only and was developed under Matlab 2013a and tested under Matlab 2010a. Since some binaries are used that were compiled for win64, this code won't work under other systems and I'm not able to give any support for those.

If you use any of this code for your project, or compare against the results reported in our paper, please cite the TMM paper accordingly. You can find all needed details on the papers webpage: http://cg.cs.uni-bonn.de/en/publications/paper-details/krueger2017Segmentation/

Tutorial for a first segmentation of MoCap data:

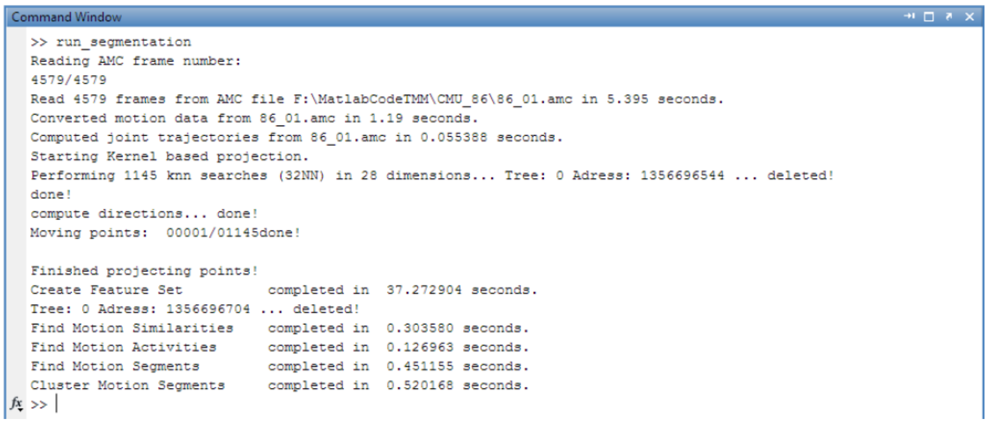
1. Extract the zip file, containing the code and some example files from the CMU Motion capture data base (F:\MatlabCodeTMM in this document).
2. Start Matlab.
3. Switch to Folder MotionSegmentation (was included to the zip file).
4. Add all folders and subfolders to Matlab path:



5. Now you are ready to start a segmentation of a motion sequence by typing this command:
```
>> run_segmentation
```

6. A dialog to select an asf and an amc file will open. Select files from the provided example data.
7. Computation will start after the file selection. A few figures will open to give some insight about the segmentation (self-similarity matrices, 3d projection of feature space before and

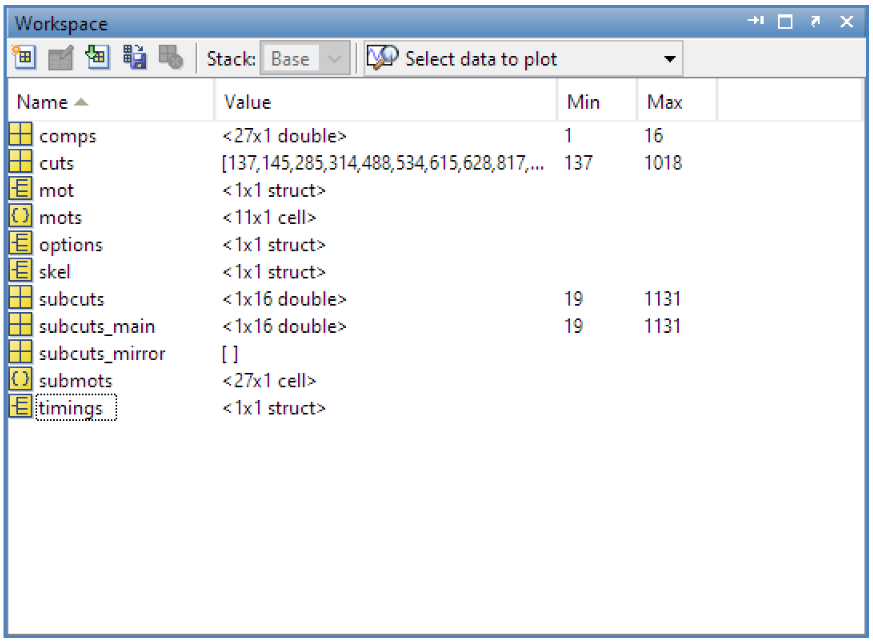
after the feature bundling. Timings and progress are shown in the Matlab console:



```
>> run_segmentation
Reading AMC frame number:
4579/4579
Read 4579 frames from AMC file F:\MatlabCodeTMM\CMU_86\86_01.amc in 5.395 seconds.
Converted motion data from 86_01.amc in 1.19 seconds.
Computed joint trajectories from 86_01.amc in 0.055388 seconds.
Starting Kernel based projection.
Performing 1145 knn searches (32NN) in 28 dimensions... Tree: 0 Adress: 1356696544 ... deleted!
done!
compute directions... done!
Moving points:  00001/01145done!

Finished projecting points!
Create Feature Set          completed in  37.272904 seconds.
Tree: 0 Adress: 1356696704 ... deleted!
Find Motion Similarities    completed in  0.303580 seconds.
Find Motion Activities      completed in  0.126963 seconds.
Find Motion Segments        completed in  0.451155 seconds.
Cluster Motion Segments     completed in  0.520168 seconds.
>> |
```

8. The output is stored in various variables:



Name	Value	Min	Max
comps	<27x1 double>	1	16
cuts	[137,145,285,314,488,534,615,628,817,...	137	1018
mot	<1x1 struct>		
mots	<11x1 cell>		
options	<1x1 struct>		
skel	<1x1 struct>		
subcuts	<1x16 double>	19	1131
subcuts_main	<1x16 double>	19	1131
subcuts_mirror	[ ]		
submots	<27x1 cell>		
timings	<1x1 struct>		

a. comps: an n x 1 array storing a cluster id for each motion segment.
b. cuts: an array storing the cuts found by the first segmentation step (region growing).
c. subcuts: an array storing the cuts found by the second segmentation step (path search).
d. skel: the skeleton structure of the mocap sequence from the .asf-file.
e. mot: the mot structure of the mocap sequence from the .amc-file.
f. mots: a cell array of structs containing the motion sequences when cutting the motion after the first segmentation step.
g. submots: a cell array of structs containing the motion sequences when cutting the input motion after the second segmentation step.

With these steps, you’re able to make a first run of the segmentation algorithm. If you need more information on the skeleton and motion data structures used in this code, consider the basictools.pdf included to the zip-file.

Some comments on options:

The code at hand has a couple of parameters that can be used to control the behavior. To play around with these parameters a struct called “options” is defined in the main script run_segmentation. You can set all segmentation parameters here. This will override the default parameters. I recommend to take some time and play around with one example sequence to get some feeling on how the algorithm works if you change these parameters. To get deeper into this, set the existing DEBUG parameters (there are several) to true. This will plot some more graphs, that will help to understand what is going on.

If you have a machine with multiple cores, and a recent Matlab version it is beneficial to open a pool of workers in Matlab to speed up the feature bundling. You can do so by typing:

```
>> matlabpool open X
```

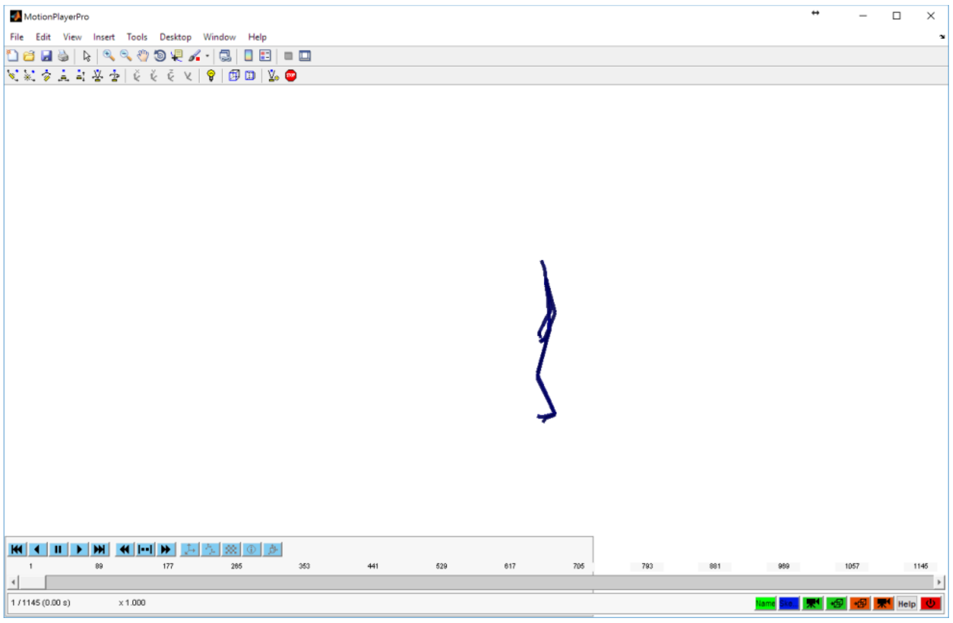
In your console. Where X is replaced by an integer number specifying the number of workers you want to run. Typically, this is the number of cores built into your machine (Sometimes I only use X-1 to make sure I have one core free for other tasks during heavy computations).

Introduction to the Motionplayer:

To watch the results of the segmentation process, it is more fun to look at the motions compared to self-similarity matrices ☺

This code package includes a Matlab based player for motion data. You can start it for a single motion by typing:

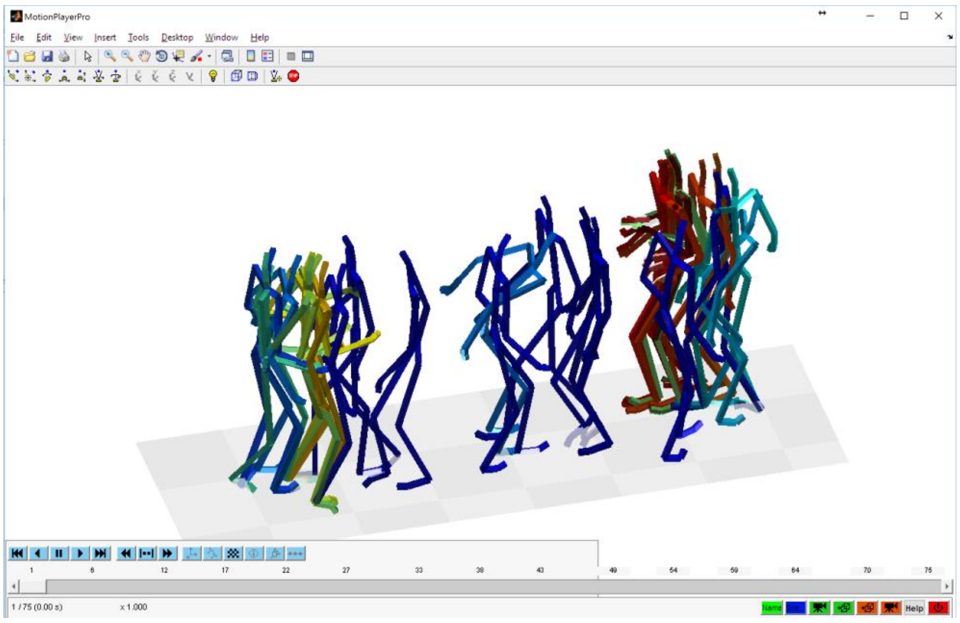
```
>> motionplayerPro('skel', skel,'mot', mot)
```



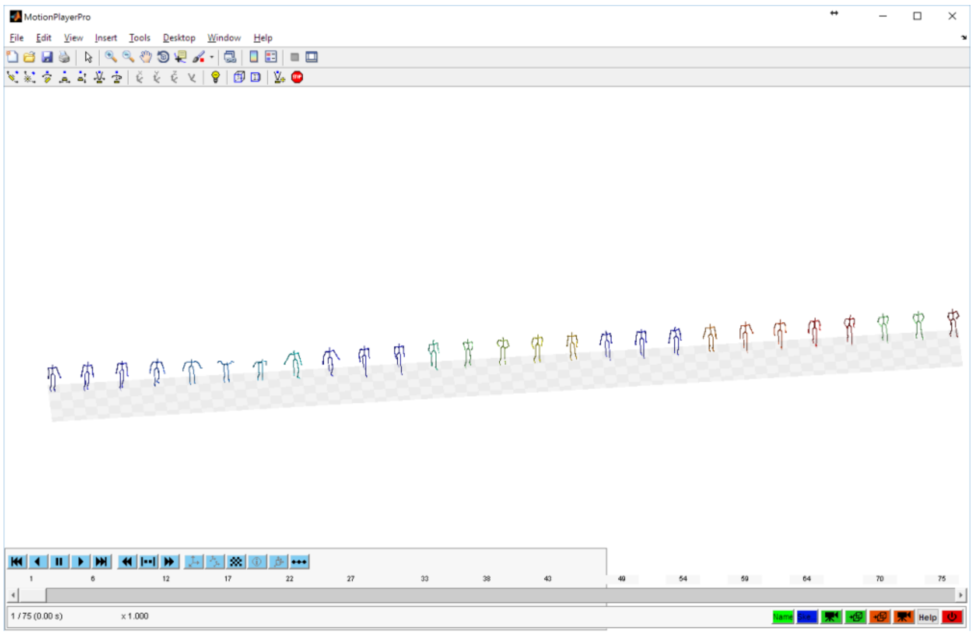
This player has some default player controls (light blue buttons), can show various coordinate frames of the mocap data, show joint names. Below the frame bar, you can display the filename of the motion (green), activate a moving camera (green), export frames to obj files (green, for better rendering if you want to), export single frames to obj (orange), render the whole current scene into an .avi video (uncompressed!). “Shift”+mouse can be used to rotate the scene, the mousewheel can be used to zoom into the scene.

If a cell array of motion structures is passed instead of a single motion struct multiple motions can be displayed:

```
>> motionplayerPro('skel', skel,'mot',submots)
```



The last button (only displayed when multiple motions are loaded) in the row of light blue buttons can be used to line up all motions:



I leave it up to you to find out all details of this Matlab figure ☺ Keep in mind, in the end it is only a Matlab figure, don’t blame me, or any of my co-authors, if this is to slow or not working perfect…