Author(s): Burger, Birgitta; Toiviainen, Petri

Title: MoCap Toolbox - A Matlab toolbox for computational analysis of movement data

Year: 2013

Version:

MOCAP TOOLBOX – A MATLAB TOOLBOX FOR COMPUTATIONAL ANALYSIS OF MOVEMENT DATA

Birgitta Burger **Petri Toiviainen**
Finnish Centre of Excellence in Interdisciplinary Music Research, Department of Music,
University of Jyväskylä, Jyväskylä, Finland
birgitta.burger@jyu.fi petri.toiviainen@jyu.fi

ABSTRACT

The *MoCap Toolbox* is a set of functions written in Matlab for analyzing and visualizing motion capture data. It is aimed at investigating music-related movement, but can be beneficial for other research areas as well. Since the toolbox code is available as open source, users can freely adapt the functions according to their needs. Users can also make use of the additional functionality that Matlab offers, such as other toolboxes, to further analyze the features extracted with the *MoCap Toolbox* within the same environment. This paper describes the structure of the toolbox and its data representations, and gives an introduction to the use of the toolbox for research and analysis purposes. The examples cover basic visualization and analysis approaches, such as general data handling, creating stick-figure images and animations, kinematic and kinetic analysis, and performing Principal Component Analysis (PCA) on movement data, from which a complexity-related movement feature is derived.

1. MOTIVATION AND OVERVIEW

The *MoCap Toolbox* is a Matlab[1] toolbox dedicated to the analysis and visualization of motion capture (MoCap) data. It has been developed for the analysis of music-related movement, but is potentially useful in other areas of studies as well. It is open source, distributed under GPL license, and freely available for download at:
www.jyu.fi/music/coe/materials/mocaptoolbox.

The *MoCap Toolbox* is mainly intended for working with recordings made with an infrared marker-based optical motion capture system. Such motion capture systems are based on an active source emitting pulses of infrared light at a very high frequency, which is reflected by small, usually spherical markers attached to the tracked object (e.g., a participant dancing or playing an instrument). With each camera capturing the position of the reflective markers in two-dimensional, a network of several cameras can be used to obtain position data in three dimensions. Besides optical motion capture, the *MoCap Toolbox* can also be used for analyzing data captured with other tracker technologies, such as inertial or magnetic trackers. However, some features of the toolbox will be limited, since such trackers do not produce position data, but derivative data, (e.g., acceleration). Furthermore, the toolbox is optimized for the use of 3-dimensional position data, so using data with six degrees of freedom (position and rotation) might require customized adjustments of functions.

There are proprietary (closed source) software solutions available for motion capture analysis and visualization, such as Visual3D[2] or MotionBuilder[3], and applications that are primarily used for recording data (such as Qualisys Track Manager[4] or Vicon Nexus[5]). However, such applications are usually either too limited in their functionality, too focused on visualization and/or too restrictive to adapt to the needs of the researcher, such as developing new movement features useful for their individual research questions. To overcome these issues, we implemented this toolbox in Matlab, a generic scientific computing environment, and made it available to other researchers to be used in favor of their needs. The *Mocap Toolbox* is not the only Matlab toolbox available for motion capture analysis; one other toolbox worth mentioning is the toolbox created by Charles Verron [1]. This toolbox is more limited than the *MoCap Toolbox*, but offers a graphical user interface (GUI).

Matlab offers pre-built visualization opportunities and gives access to a large range of other functionality. Some functions included in the *MoCap Toolbox* use, for example, the *Signal Processing Toolbox* provided by MathWorks, or the *FastICA* package[6], a freely available third-party toolbox for Independent Component Analysis. Furthermore, the users themselves can make immediate use of the additional functionality and toolboxes provided by Matlab, for example the *Statistics Toolbox*, to further analyze features extracted with the *MoCap Toolbox* without the need to switch between different applications. *MoCap Toolbox* code is written using the generic Matlab syntax and is openly assessable, so users can add and adapt functions to their own needs.

[1] www.mathworks.com

[2] www.c-motion.com/products/visual3d/
[3] www.autodesk.com/motionbuilder
[4] www.qualisys.com/products/software/qtm/
[5] www.vicon.com/products/nexus.html
[6] www.cis.hut.fi/projects/ica/fastica/

The *MoCap Toolbox* supports various motion capture data formats, in particular the .c3d[7] file format (which, e.g., Vicon[8] or OptiTrack[9] optical motion capture systems can produce), the .tsv format and the .mat format, both produced by the Qualisys motion capture system[10], and the .wii data format produced by the *WiiDataCapture* software[11].

The *MoCap Toolbox* provides 64 functions for analyzing and visualizing motion capture data. The main categories can be summarized as data input and edit functions, coordinate transformation and coordinate system conversion functions, kinematic and kinetic analysis functions, time-series analysis functions, visualization functions, and projection functions. Furthermore, it uses three different data structures, the *MoCap data structure*, the *norm data structure*, and the *segm data structure*. To convert between the different data representations and enable certain visualizations, three different parameter structures are used, the *m2jpar*, the *j2spar*, and the *animpar structures*. Both the data and the parameter structures will be discussed and explained in the next section.

2. DATA REPRESENTATIONS

The *MoCap Toolbox* uses three different data structures, the *MoCap data structure*, the *norm data structure*, and the *segm data structure*. A *MoCap data structure* instance is created when mocap data is read from a file to the Matlab workspace using the function `mcread`. A *MoCap data structure* contains the 3-dimensional locations of the markers (in the `.data` field) as well as basic information, including the type of structure, the file name, number of frames of the recording, the number of cameras used for the recording, the number of markers in the data, the frame rate, the names of the markers, and the order of time differentiation of the data. Additionally, the *MoCap data structure* contains fields for data captured with analog data, such as EMG. Finally, the time stamp of the recording and the data type (e.g., 3D) can be added.

A *MoCap data structure* instance is also created when the function `mcm2j` is used. This function transforms a marker representation to a joint representation. These two representations use the same data structure, although they are conceptually different: the marker representation reflects the actual marker locations, whereas the joint representation is related to locations derived from marker locations. A joint can consist of one marker, but it can also be derived from more than one markers. It can, for example, be used for calculating the location of a body part where it is impossible to attach a marker. The midpoint of a joint, for instance, can be then derived as the centroid of four markers around the joint.

The *norm data structure*, created by the function `mcnorm`, is similar to the *MoCap data structure*, except that its `.data` field has only one column per marker. This column contains the Euclidean norm of the vector data from which it was derived. If, for instance, `mcnorm` is applied to velocity data, the resulting *norm data structure* holds the magnitudes of velocities, or speeds, of each marker.

The third data structure, the *segm data structure*, is not, like the other two, related to points in space (markers or joints), but to segments of the body (see, e.g., [2]). The function `mcj2s` performs a transformation from a joint representation to a segment representation and produces as output a *segm data structure* instance. Most fields of a *segm data structure* are similar to the ones of a *MoCap data structure*, however, the `.data` field is replaced by four other fields. The `.parent` field contains information about the kinematic chains of the body, i.e., how the joints are connected to form segments, and how segments are connected to each other. The fields `.roottrans` and `.rootrot` store the location and orientation of the center of the body, the root. The `.segm` field consists of several subfields that store the orientation of the body segments in several ways. The `.eucl` subfield contains for each segment the Euclidean vector pointing from the proximal to the distal joint of the segment. The length of each segment is stored in the `.r` subfield. The `.quat` subfield includes the rotation of each segment as a quaternion representation (see, e.g., [3] and [4]). Finally, the `.angle` subfield contains the angles between each segment and its proximal segment.

To convert between the different representations and to enable certain visualizations, the *MoCap Toolbox* offers three different parameter structures: *m2jpar, j2spar*, and the *animpar structures*.

The *m2jpar structure* is used by the function `mcm2j` and contains the information needed to perform the transformation from marker to joint representation. Besides fields holding the number of joints and the names of the joints, it includes a field with the numbers of the markers defining the location of each joint.

The *j2spar structure* is used by the function `mcj2s` and contains the information needed to perform the transformation from joint to segment representation. Besides the fields containing the segment names and the number of the root (center of the body) joint, it includes fields with the numbers of the three joints that define the frontal plane of the body and a vector indicating the number of the parent segment (the segment that is proximal in the kinematic chain) for each segment.

The *animpar structure* is used by the functions `mcplotframe` and `mcanimate` and contains the information needed to create frame (stick figure) plots and animations. The structure includes fields for the screen size, limits of the plotted area, viewing angles, marker sizes, plotting colors, connection line configurations and widths, and plotting of marker and frame numbers. Additionally, the structure contains fields related to creating animations, such as the frames per second, a substructure for perspective projection parameters, and settings for plotting marker traces.

[7] www.c3d.org
[8] www.vicon.com
[9] www.naturalpoint.com/optitrack/
[10] www.qualisys.com
[11] www.jyu.fi/music/coe/materials/mocaptoolbox

3. USING THE TOOLBOX

In what follows, we will give an introduction to the use of the toolbox for research and analysis purposes.

The *MoCap Toolbox* manual, provided with the download of the toolbox, offers an example chapter with eleven demos explaining the basic usage of the toolbox. Additionally, a demo data set called `mcdemodata`, including motion capture data and associated parameter structures, is provided with the download. The *MoCap data structures* `dance1` and `dance2` used below are available in the `mcdemodata` data set.

3.1 Reading Data and Filling Gaps

Recorded motion capture files can be imported into Matlab using the function `mcread` storing the content of the file as a *MoCap data structure*, i.e.,

```
d = mcread('file.tsv');
```

An essentially useful first step is usually to check for missing frames in the recording. Taking the *mocap data structure* `d`, we can use

```
mcmissing(d)
```

to detect missing frames in the recording. In case of missing data, we can fill them using linear interpolation with the function `mcfillgaps`:

```
d = mcfillgaps(d);
```

From this point onwards, we will use the two *MoCap data structures* `dance1` and `dance2` from the `mcdemodata`. Since they are already available as *MoCap data structures* and do not contain missing data, both importing and gap filling are not required anymore.

3.2 Visualizing and Animating Data

A good approach to get an overview of the data is to visualize and animate data. Using the *MoCap Toolbox*, mocap data can be plotted in different two ways: as a time series or as single frames. As a function of time, marker location data can be plotted with the function `mcplottimeseries`, e.g.,

```
mcplottimeseries(dance1,[1 20 28],
 'dim',3)
```

which plots the third/vertical dimension of markers 1, 20, and 28 (left front head, right hand, and right foot) (see Fig. 1).

Marker locations as single frames can be plotted using the function `mcplotframe` (using the (x,y) projection of the markers):

```
mcplotframe(dance1,450);
```

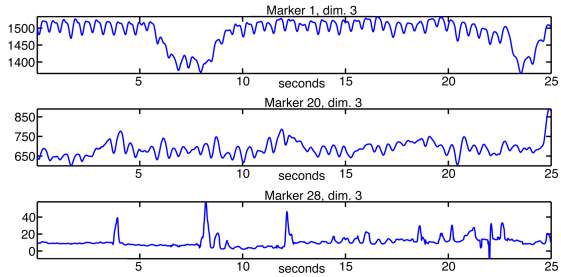

Figure 1. Marker location data plotted as function of time using `mcplottimeseries`.

This call, plotting the 450th frame of the recording (see Fig 2a), uses the default *animation parameter structure*. However, if a customized *animpar* structure is used, we can, for instance, set the connection lines between the markers to obtain a visualization that is easier to understand and that looks more human-like (see Fig. 2b):

```
ap = mcinitanimpar;
ap.conn = [1 2; 2 4; 3 4; 3 1; 5 6; 9
 10; 10 12; 11 12; 11 9; 8 9; 8 10; 8
 5; 8 6; 5 9; 5 11; 6 10; 6 12; 7 11;
 7 12; 7 5; 7 6; 5 13; 13 15; 13 16;
 16 19; 15 19; 6 14; 14 17; 14 18; 17
 20; 18 20; 9 21; 11 21; 10 22; 12 22;
 21 23; 23 25; 23 26; 25 26; 22 24; 24
 27; 24 28; 27 28];
mcplotframe(dance1,450,ap);
```

In case users collected the data with a Qualisys motion capture system and created a bone structure during the labeling process in the Qualisys software, they can export the so-called label list (which contains the marker connections) and use this file to create the connection matrix by employing the function `mccreateconnmatrix`.

We can change the general color scheme and the colors of individual markers, connector lines, traces, and numbers by adjusting the values of the respective fields of the *animpar* structure, for example (see Fig. 2c):

```
ap.colors = 'wrbgy';
ap.markercolors = 'bmgyrrrrrrrrkk';
mcplotframe(dance1,450,ap);
```

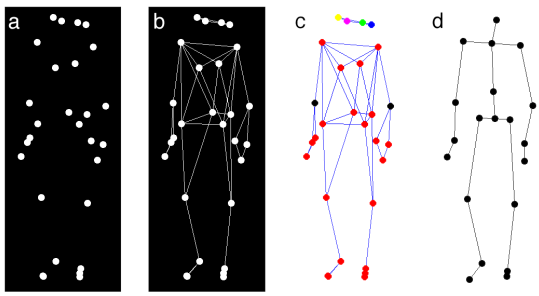

Figure 2. Marker location data plotted as frame using `mcplotframe`: a) using the default parameters; b) using a connection matrix; c) changing colors; d) joint transformation.

The function `mcanimate` is used to create animations:

```
mcanimate(dance1,an);
```

The *MoCap Toolbox* produces the single animation frames as .png files. They have to be compiled into a movie using other software, such as QuickTime Pro on Mac, or MovieMaker on Windows.

Animations can be created as 2D projections in two ways, either orthographic (default) or perspective, the latter one by including the perspective projection parameter:

```
mcanimate(dance1,an,1);
```

3.3 Kinematic Analysis

Kinematic variables, such as velocity and acceleration, are estimated using the time-derivative function `mctimeder`:

```
d1v = mctimeder(dance1,1); %vel.
d2v = mctimeder(dance2,1);
d1a = mctimeder(dance1,2); %acc.
d2a = mctimeder(dance2,2);
```

To analyze such time series, we can calculate their means and standard deviations using `mcmean` and `mcstd` (ignoring eventual missing frames). For this sample analysis, we will take the norm data, that is, the magnitudes of the 3-dimensional data of velocity and acceleration. To simplify the approach, we first combine the data from marker 1 (left front head) of the four *MoCap data structures* using mcconcatenate:

```
dva = mcconcatenate(d1v,1,d1a,1,d2v,1,
 d2a,1);
dva_mean = mcmean(mcnorm(dva));
dva_std = mcstd(mcnorm(dva));
```

The results (see Table 1) show that both mean and standard deviation of velocity and acceleration of the left front head marker are higher for `dance2` than for `dance1`, so the dancer in `dance2` moved faster and at a wider range of speeds and also used more and larger directional changes.

		mean	SD
velocity	`dance1`	235.85	110.79
	`dance2`	520.24	192.27
acceleration	`dance1`	2233.66	1326.55
	`dance2`	3347.21	1423.19

Table 1. Means and standard deviations of velocity and acceleration (magnitudes) of the left front head marker data of `dance1` and `dance2`.

The cumulative distance travelled by a marker can be calculated with the function `mccumdist` (returning a *norm data structure*):

```
d1dist = mccumdist(dance1);
d2dist = mccumdist(dance2);
```

We use the Matlab function `barh` for plotting markers 1 (left front head), 20 (right finger), and 28 (right foot) (see Fig. 3):

```
figure, barh([d1dist.data(1500,[1 20
 28]); d2dist.data(1500,[1 20
 28])],'b');
```

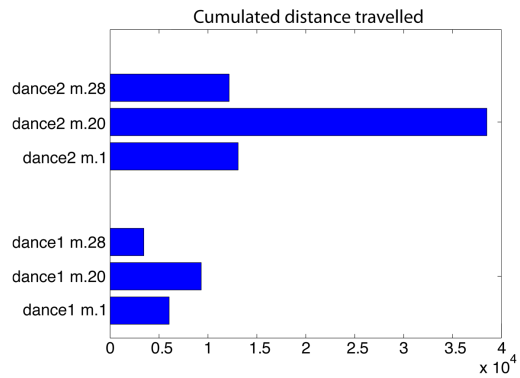

Figure 3. Cumulated distance travelled by markers 1, 20, and 28 of mocap data `dance1` and `dance2` (labels and title were added separately).

We can see in Figure 3 that the three markers, especially the right hand marker, travelled more for `dance2` than for `dance1`, so we can assume that the amount of movement was higher in `dance2`.

A measure related to the amount of movement is the area covered by the movement, which can be calculated using `mcboundrect`. If we want to calculate the bounding rectangle of the four hip markers, we do:

```
br1 = mean(mean(mcboundrect(dance1,[9
 10 11 12])));
br2 = mean(mean(mcboundrect(dance2,[9
 10 11 12])));[12]
```

The bounding rectangle value for `dance1` equals .1806 and for `dance2`, it equals .9724. Since the value for `dance2` is higher, `dance2` not only had a higher amount of movement, but also used more space than `dance1`. The bounding rectangle measure was found to be a relevant movement feature in [5] and [6].

We can also calculate distances between markers using `mcmarkerdist`. The standard deviation of the distance between left and right finger,

```
md1 = std(mcmarkerdist(dance1,19,20));
md2 = std(mcmarkerdist(dance2,19,20));
```

gives us information about the variability of the marker distance. The standard deviation of the finger marker distance for `dance1` equals 49.0 and for `dance2` 175.65, so the fingers in `dance2` exhibited more variable distances.

Periodicity of movement can be estimated using the function `mcperiod`. It is based on autocorrelation, and

[12] `mcboundrect` uses window decomposition. The function output here is averaged across the windows and the four markers.

either the first or highest peak of the autocorrelation function is taken as periodicity estimation dependent on the parameter input. With

```
d1m1 = mcgetmarker(d1a,20);
d2m1 = mcgetmarker(d2a,20);
[per1 ac1 eac1] = mcperiod(d1m1,2,
 'highest');
[per2 ac2 eac2] = mcperiod(d2m1,2,
 'highest');
```

we calculate the periodicity of the acceleration of the right finger marker. `mcperiod` resulted in a periodicity estimate for each dimension being [1.04, 0.53, 0.52] for `dance1` and [1.04, 1.01, 1.06] for `dance2`. While the first dimension is similar, the second and third dimensions are roughly half for `dance1`, suggesting that in this case the finger moved in double tempo in y and z directions.

A more accurate periodicity analysis can be performed using windowed autocorrelation:

```
[per1 ac1 eac1] = mcwindow(@mcperiod,
 d1m1,2,0.25);
[per2 ac2 eac2] = mcwindow(@mcperiod,
 d2m1,2,0.25);
```

To allow visual inspection of the time development of the periodicity, the enhanced autocorrelation (eac) matrix can be plotted as an image (see Fig. 4). The colors indicate the regularity of periodic movement, with warm colors corresponding to regions of regular periodic movement in the period-time plane:

```
figure, imagesc(eac1(:,:,3)), axis xy
set(gca, 'XTick',0:4:46, 'XTickLabel',
 0.5*(0:4:46), 'YTick',[0 30 60 90
 120], 'YTickLabel',[0 0.5 1 1.5 2.0])
figure, imagesc(eac2(:,:,3)), axis xy
set(gca, 'XTick',0:4:46, 'XTickLabel',
 0.5*(0:4:46), 'YTick',[0 30 60 90
 120], 'YTickLabel',[0 0.5 1 1.5 2.0])
```

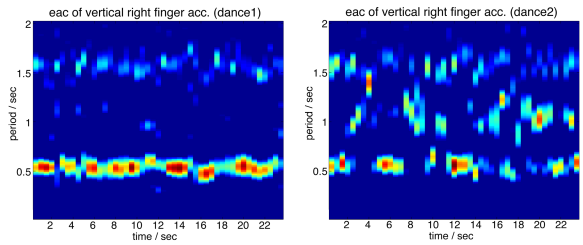

Figure 4. Enhanced autocorrelation function of the vertical components of the right finger acceleration in `dance1` and `dance2`.

We can see in Figure 4 that the vertical component of the right finger acceleration of `dance1` shows quite clear periodic movement with a period of about 500 milliseconds, whereas the periodicity for `dance2` is weaker and more irregular.

3.4 Kinetic Analysis

The *MoCap toolbox* offers the possibility to calculate kinetic variables using Dempster's body-segment model [7]. To make our present data compatible with Dempster's model, we first have to reduce the amount of markers from 28 to 20. We will accomplish this with a marker-to-joint transformation, implemented in the function `mcm2j`. The *m2jpar parameter structure* required for this transformation is created like this:

```
m2j = mcinitm2jpar;
m2j.nMarkers = 20;
m2j.markerNum = {[9 10 11 12],[9 11],
 21,23,26,[10 12],22,24,28,[7 8 7 8 9
 10 11 12],[5 6],[1 2 3 4],5,13,[15
 16],19,6,14,[17 18],20};
m2j.markerName = {'root', 'lhip',
 'lknee','lankle','ltoe','rhip',
 'rknee','rankle','rtoe','midtorso',
 'neck','head','lshoulder','lelbow',
 'lwrist','lfinger','rshoulder',
 'relbow','rwrist','rfinger'};
```

The joint `'root'`, for example, is obtained by calculating the centroid of markers 9, 10, 11, and 12. The marker-to-joint transformation is carried out as follows:

```
d1j = mcm2j(dance1,m2j);
d2j = mcm2j(dance2,m2j);
```

Figure 2d visualizes frame 450 of the joint representation of `dance1`. The next step is to do the joint-to-segment transformation. The *j2spar parameter structure* required for the transformation is created like this:

```
j2s = mcinitj2spar;
j2s.rootMarker = 1;
j2s.frontalPlane = [6 2 10];
j2s.parent = [0 1 2 3 4 1 6 7 8 1 10
 11 11 13 14 15 11 17 18 19];
j2s.segmentName = {'lhip','lthigh',
 'lleg','lfoot','rhip','rthigh',
 'rleg','rfoot','ltorso','utorso',
 'neck','lshoulder','luarm','llarm',
 'lhand','rshoulder','ruarm','rlarm',
 'rhand'};
```

The joint-to-segment transformation is accomplished using the function `mcj2s`:

```
d1s = mcj2s(d1j,j2s);
d2s = mcj2s(d2j,j2s);
```

In order to calculate kinetic variables, such as energy, each body part has to be associated to its parameter (i.e., masses and lengths) specified by the Dempster model. Therefore, a variable is created specifying the types of the segments[13]:

[13] For a list of the segment types, see the *MoCap Toolbox* manual.

```
s_ind = [0 0 8 7 6 0 8 7 6 13 12 10 11
 3 2 1 11 3 2 1];
```

This variable associates each joint with a segment type. Each component indicates the type of body segment for which the respective joint is a distal joint. Joints that are not distal to any segment have zero values.

The parameters for each body segment can be then obtained using the function `mcgetsegmpar`:

```
spar = mcgetsegmpar('Dempster',s_ind);
```

With this body-segment representation we can estimate kinetic variables for each segment individually. The time-average of the kinetic energy of the whole body, for example, can be calculated like this:

```
[trans1 rot1] = mckinenergy(d1j,d1s,
 spar);
[trans2 rot2] = mckinenergy(d2j,d2s,
 spar);
kinEn1 = sum(mcmean(trans1)) +
 sum(mcmean(rot1));
kinEn2 = sum(mcmean(trans2)) +
 sum(mcmean(rot2));
```

The value for the overall kinetic energy of `dance1` equals 2.21, and the value for `dance2` is 11.37, thus more energy was used in `dance2`, which supports our argumentation drawn earlier, that there is more movement in `dance2` than in `dance1`.

3.5 Principal Component Analysis (PCA)

Principal component analysis can be used to decompose motion capture data into components that are orthogonal to each other. By using

```
[pc1 p1] = mcpcaproj(d1j,1:5);
[pc2 p2] = mcpcaproj(d2j,1:5);
```

we calculate the first five principle component projections of the position data (as joint representations) of `d1j` and `d2j`. `p1.l` and `p2.l` contain the amount of variance explained by each component. From these variances, we can derive, for instance, a measure of movement complexity, defined as the cumulative sum of the proportion of explained variance contained in the first five PCs (see, e.g., [5] and [8]):

```
pcapropvar1 = cumsum(p1.l(1:5));
pcapropvar2 = cumsum(p2.l(1:5));
```

The results, presented in Table 2, indicate that, in case of `pcapropvar1`, most movement is already explained with the first component, and the first five components explain almost all movement. In case of `pcapropvar2`, however, only about 50% of the movement is explained with the first component, and the first five components explain less than the first five components of `pcapropvar1`, so more components are needed to fully explain the movements of `dance2`. Such a movement would be characterized as complex, since a high number of PCs is needed to explain the movement sufficiently, whereas a low proportion of unexplained variance (`dance1` case) implies a simpler movement.

	`pcapropvar1`	`pcapropvar2`
`cumsum(1)`	0.79	0.48
`cumsum(1:2)`	0.90	0.78
`cumsum(1:3)`	0.95	0.85
`cumsum(1:4)`	0.97	0.90
`cumsum(1:5)`	0.98	0.93

Table 2. Cumulative variances of the first five principle components for `dance1` (`pcapropvar1`) and `dance2` (`pcapropvar2`).

4. CONCLUSION

The *MoCap Toolbox* is a Matlab toolbox dedicated to the analysis and visualization of motion capture data. It has been developed for the analysis of music-related movement, but is potentially useful in other areas of studies as well. It has attracted researchers' attention working in various fields and has been downloaded for being used in a wide range of different research purposes; music-related, but also, for instance, face recognition, sports, gait, or biomechanics research. It has also gained attraction in artificial intelligence research, such as robotic motion, human-robot interaction, and machine learning.

The *MoCap Toolbox* has continuously been developed further since its first launch in 2008 by both the authors and the users, whose bug reports and suggestions for new functionality has greatly helped to improve and extend it.

In the future error handling will be improved, for instance, when wrong data structures are used. Toolbox functions usually recognize the mistake, but in the present version, some functions do not return sufficiently clear error messages.

Furthermore, some functions will be adapted to standard Matlab conventions, as it is already done in, for instance, `mcplottimeseries` (specifying the plotting parameters as a strings-value combination).

Individual functions will be improved, such as `mcfillgaps`, that would benefit from the implementation of more advanced gap-filling methods than linear filling, for example spline interpolation. Additionally, more body segment models besides Dempster's model will be included, such as models proposed in [9] or [10].

As commercial tools (e.g., Visual3D) commonly provide GUIs instead of operating on a command-line basis, a graphical user interface could also be implemented for the *MoCapToolbox*. It would make the toolbox more user-friendly – for example, connection matrices of stick figures could be drawn in the GUI, or gap filling could be graphically supported.

Acknowledgments

This study was supported by the Academy of Finland (project 118616).

5. REFERENCES

[1] C. Verron, Traitement et Visualisation de Vonnées Gestuelles Captées par Optotrak. IDMIL Report, 2005.

[2] D.G.E. Robertson, G.E. Caldwell, J. Hamill, G. Kamen, and S.N. Whittlesey, Research Methods in Biomechanics. Human Kinetics, 2004.

[3] P. Kelland, Introduction to Quaternions, with Numerous Examples. Rarebooksclub.com, 2012.

[4] A.J. Hanson, Visualizing Quaternions. Morgan Kaufmann Publishers, 2005.

[5] B. Burger, S. Saarikallio, G. Luck, M.R. Thompson, and P. Toiviainen, "Relationships between perceived emotions in music and music-induced movement," in Music Perception 30, 2013, pp. 519-535.

[6] G. Luck, S. Saarikallio, B. Burger, M.R. Thompson, and P. Toiviainen, "Effects of the Big Five and musical genre on music-induced movement," in Research in Personality 44, 2010, pp. 714-720.

[7] W.T. Dempster, Space Requirements of the Seated Operator: Geometrical, Kinematic, and Mechanical Aspects of the Body with Special Reference to the Limbs. WADC Technical Report 55-159, Wright-Patterson Air Force Base, 1955.

[8] S. Saarikallio, G. Luck, B. Burger, M.R. Thompson, and P. Toiviainen, "Dance moves reflect current affective state illustrative of approach-avoidance motivation," in Psychology of Aesthetics, Creativity, and the Arts, in press.

[9] C.E. Clauser, J.T. McConville, and J.W. Young, Weight, volume and center of mass of segments of the human body. AMRL Technical Report 69-70, Wright-Patterson Air Force Base, 1969.

[10] C.L. Vaughan, B.L. Davis, and J.C. O'Connor, Dynamics of Human Gait. Human Kinetics, 1992.