MPI08 file format description

Gerard Pons-Moll and Andreas Baak

September 30, 2010

Abstract

This document contains the description of the file formats and conventions used in the MPI08 database. It contains the file format description of 3 different types of file: `MeshModel.dat`, `SensorOris.dat`, `Proj??.dat` and `MPI_<actor>_<seq>_<take>_<frameRate>_<view>.mat`, which correspond to 3D mesh model of the subject, calibrated/uncalibrated and synchronized sensor orientation data, camera projection matrices and segmented silhouette images. This description corresponds to the data used in [1, 2] which we make public for research purposes.

1 Filetypes and formats

In this section a description of the different files needed to use the MPI08 database is given. The descriptions explained here correspond to the data used in the projects *Multisensor Fusion for 3D Full-Body Human Motion Capture* [1] and *Analyzing and Evaluating Markerless Motion Tracking Using Inertial Sensors* [2]. Please for a practical guide on how to read and use this files, please use the `MPI08_demo.m` file included in the database.

1.1 Files of type MeshModel.dat

This files are located in the `InputFiles` directory. They contain a mesh model together with an inserted skeletton. The structure of the file is the following:

Header The header should include 4 numbers:

- `Number of Vertices`: The total number of vertex points of the 3D mesh model
- `Number of Faces`: The total number of faces of the 3D mesh model. A face is each of the triangular patches conecting three vertices of the mesh.
- `Number of Joints`: Total number of joints of the 3D mesh model skeletton. The number of joints determine the allowed articulated motions of the body. A joint is a point in the mesh conecting to rigid segments, i.e. the elbow. The nature of the joint is determined by its location and direction. Note that a 3 Dof joint like the shoulder is generally modelled by the concatenation of 3 revolute joints. Therefore, this joints count as 3 in the joint count. IMPORTANT: The root joint is not counted.

- `Number of Skinning Weights`: Since the human motion is not completely rigid, skinning weights are incorporated. This weights should add up to one. Thereby, each vertex is influenced by as many joints as skinning weights and the influence of each joint is determined by the corresponding weight. This rather simple approach is commonly known as *Linear Blend Skinning*, an excellent description can be found in [3].

Data The 3D mesh model is represented by three diferent kinds of data. First, the vertex points, second, the faces of the mesh and finally the joints forming the skeletton of the mesh. The data is arranged in the file with the following convention:

- `VERTICES`:

```
X   Y   Z   J.ID1   w1   J.ID2   w2
```

  - `X,Y,Z`: 3D coordinates of the mesh vertex.
  - `J.ID1`: Joint ID influencing the vertex
  - `w1`: Weight of joint J.ID1
  - `J.ID2`: Joint ID influencing the vertex
  - `w2`: Weight of joint J.ID2

- `FACES`:

```
V.ID1   V.ID2   V.ID3
```

  - `V.ID1`: Vertex ID of the first vertex of the face.
  - `V.ID2`: Vertex ID of the second vertex of the face.
  - `V.ID3` :Vertex ID of the third vertex of the face.

- `JOINTS`:

```
J-ID   X   Y   Z   ω1   ω2   ω3   PJ.ID   Ball
```

  - `J-ID`: Joint ID
  - `X,Y,Z` : 3D coordinates of the joint location, q
  - ω_1 , ω_2 , ω_3: Unit rotation axes vector, the vertices influenced by this joint can rotate around this axes.
  - `PJ.ID`: The parent joint ID. For example, the knee joint is the parent of the ankle joint.
  - `Ball`: Indicates whether this is a ball joint (3 DoF joint, usually shoulders and hips). $0 \rightarrow$ revolute joint, $1 \rightarrow$ balljoint like in [4], by default it is set to 0.

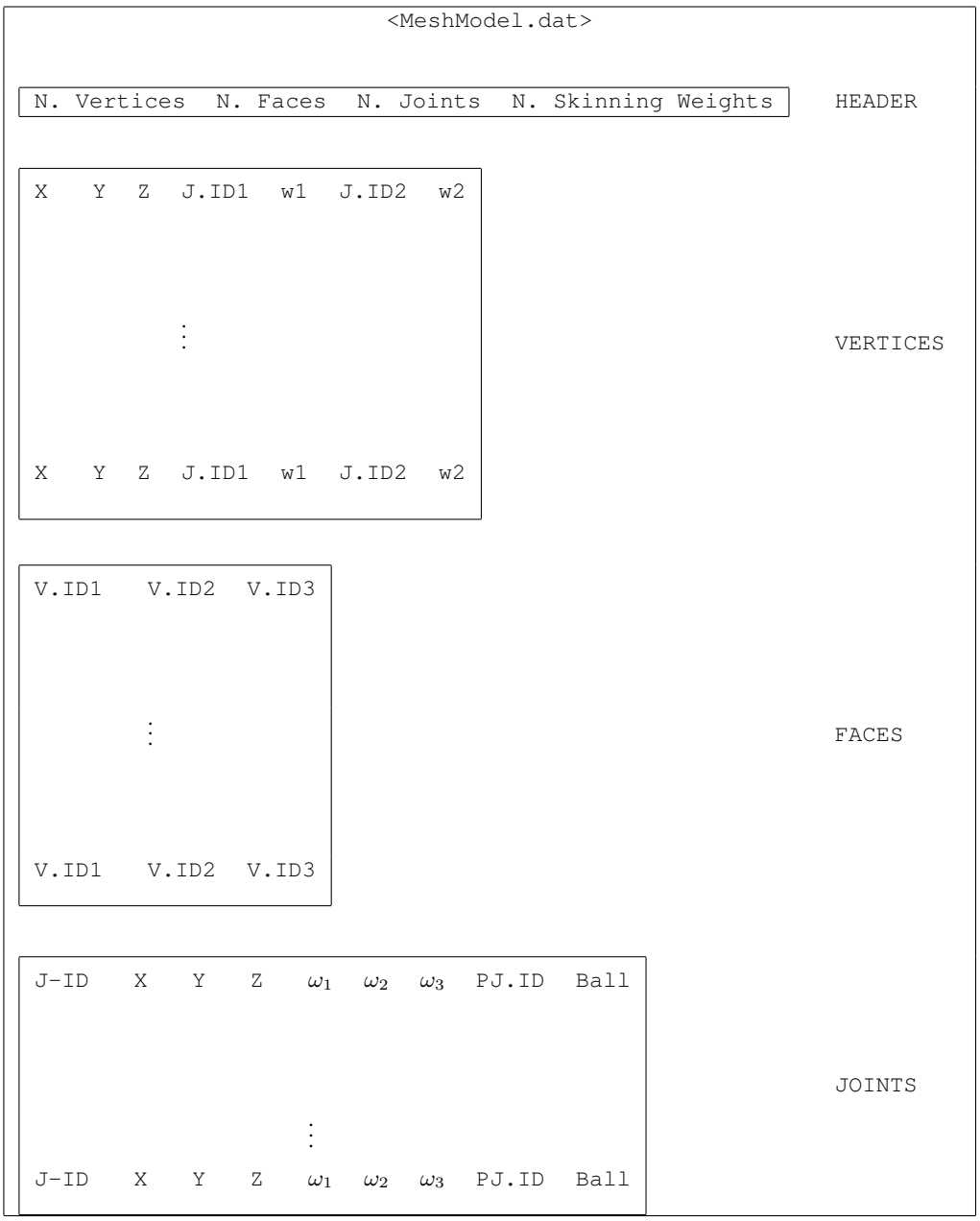

Figure 1: MeshModel.dat file description

1.2 Files of type SensorOrientation.dat

This files are located in the `MPI08_PriorFiles`. This files contain the orientation data from the Inertial Measurement Units (IMU). In the recordings five IMU where used attached at the limb extremities and the back neck, see [1]. The orientation of each of the sensor is given as a quaternion (4 numbers) for each frame.

Header

```
Numframes:  406   NumSensors 5
lankle 556   lankle 1032   neck 134   lwrist 432   rlwrist 916
```

The first line of the file contains the number of frames of the sequence and the number of sensors used. In the example above the number of frames is 406 and the number of sensors is 5. The second line of the file consists of N columns, as many columns as sensors, 5 in our case. Each column consists of the body part where the sensor was attached and the joint ID of the mesh by which the sensor is influenced. Thereby the second line of the file alternates `BodyPart J.ID`, see the file illustration in Figure 2. The first corresponds to the left shin (`lknee_lankle`), the second to the right shin (rknee_rankle), the third to the upper back (chest_neck), the fourth to the left hand (lwrist_lhand), and the last to the right hand (rwrist_rhand).

Data Each new line of data corresponds to the quaternions of each of the sensors (5 in our case) starting from the first frame of the sequence. Each new line indicates a new frame in the sequence. A line contains 20 numbers, defining 5 unit quaternions, one for each sensor. The vector entries of each quaternion are stored in the order $q = (w, x, y, z)$, so first the real part of the quaternion, and then the three imaginary parts are depicted. The orientation data for each sequence is given in three different

SensorOrientaiton.dat

```
Numframes:  406   NumSensors 5
lknee_lankle 15   rknee_lankle 20   chest_neck 22   lwrist_lhand 6   rlwrist_lhand 11
```

$q_w^1 \quad q_x^1 \quad q_y^1 \quad q_z^1 \quad \cdots \quad q_w^3 \quad q_x^3 \quad q_y^3 \quad q_z^3 \quad \cdots \quad q_w^5 \quad q_x^5 \quad q_y^5 \quad q_z^5$ frame 1

$\vdots \qquad\qquad\qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots$

$q_w^1 \quad q_x^1 \quad q_y^1 \quad q_z^1 \quad \cdots \quad q_w^3 \quad q_x^3 \quad q_y^3 \quad q_z^3 \quad \cdots \quad q_w^5 \quad q_x^5 \quad q_y^5 \quad q_z^5$ frame N

Figure 2: SensorOrientation.dat file description

representations for each sequence. The files with the prefix `SensorOrisTa_` contain the orientation data that is temporally aligned to the 40 fps image sequences. Additionally, the orientation data has been rotated so that in the reference coordinate system the Y-axis points upwards (as opposed to the Xsens convention that the Z-axis points upwards).

The files with the prefix `SensorOrisTaSaOnlyGlobal_` contain the orientation data that is now expressed in the global tracking coordinate system. That is, the coordinate system offset (obtained from calibration) q_G has been applied to express the orientations in the global inertial coordinate system . These files have been used in the project Multisensor-Fusion for 3D Full-Body Human Motion Capture.

The files with the prefix `SensorOrisTaSa_` contain orientations that are calibrated so that a) the offset from the inertial coordinate system to the tracking coordinate system is aligned and b) the offset from the sensor coordinate system to the bone coordinate system of our skeleton is aligned. These files have been us in the project Analyzing and Evaluating Markerless Motion Tracking Using Inertial Sensors [2]. The orientation data is - for ease of parsing and debugging - stored in text files.

1.3 Files of type proj??.dat

This files are located in the `InputFiles` directory. The camera projection matrices are obtained with the camera calibration toolbox in Matlab. The `proj??.dat` files contain the camera projection matrices for each camera. In the `proj??.dat` the *intrinsic and* extrinsic camera parameters are included in the form of 3 matrices $P, \mathcal{K}, \mathcal{M}$ using the same notation as in [5].

Header : The header is a letter P,K,M specifing the matrix.

Data :

1. Intrinsic parameters $\mathcal{K}$:

$$\mathcal{K} = \begin{pmatrix} fc_1 & 0 & cc_1 \\ 0 & fc_2 & cc_1 \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

   with *Focal length* $f_c = [fc_1, fc_2]^T$ *and* Principal point $cc = [cc_1\ cc_2]^T$. Note: The focal length is stored in the 2x1 vector fc and the principal point are stored in the 2x1 vector cc in pixel units.
   Important Convention: Pixel coordinates are defined such that [0;0] is the center of the upper left pixel of the image. As a result, [nx-1;0] is center of the upper right corner pixel, [0;ny-1] is the center of the lower left corner pixel and [nx-1;ny-1] is the center of the lower right corner pixel where nx and ny are the width and height of the image (for the images of the first example, nx=640 and ny=480).
   One matlab function provided in the toolbox computes that direct pixel projection map. This function is project_points2.m. This function takes in the 3D coordinates of a set of points in space (in world reference frame or camera reference frame) and the intrinsic camera parameters (fc,cc,kc,alpha_c), and returns the pixel projections of the points on the image plane. See the information given in the function.

2. Extrinsic parameters $\mathcal{K}$:

$$\mathcal{M} = \begin{pmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1 \end{pmatrix} \tag{2}$$

$\mathcal{M}$ is the transformation from the world coordinate system (the calibration cube) and the camera frame. If $\mathcal{M} = [A|b]$ then the *camera center* in world coordinates is $C^{\mathcal{W}} = -A^{-1}b$;

3. Intrinsic + Extrinsics camera projection P :

$$P = [\mathcal{K}|0]\mathcal{M} \tag{3}$$

The matrix P maps a point in workd coordinates $X^{\mathcal{W}}$ to pixel coordinates x.

1.4 Silhouette files

This files are located in the `Silhouettes` directory. The silhouettes are stored in a matlab file. This file `MPI_<actor>_<seq>_<take>_<frameRate>_<view>.mat` contains all the silhouette images corresponding to a full take for one camera view. Therefore, in the MPI08 database there is one of this files per view and take.

Data : The matlab file consists of 5 fields:

- `frameHeight`: y-size of the image
- `frameWidth`: x-size of the image
- `nframes`: Number of frames of the take
- `sampling Rate`: sequence frame rate
- `frameRLE`: cell array of dimension $1 \times nframes$. Each cell contains the image run-length encoded. The images can be decoded with the matlab function `runLengthDecode.m` included in the matlab demo file `MPI08_demo.m`.

Besides the matlab files the silhouettes are also included as text files `MPI_<actor>_<seq>_<take>_<frameRate>_<view>.dat` in the same format for easy parsing from c/c++ and for compatiblity with systems without matlab license.

References

[1] G. Pons-Moll, A. Baak, T. Helten, M. Müller, H.-P. Seidel, and B. Rosenhahn, "Multisensor-fusion for 3d full-body human motion capture," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2010.

[2] A. Baak, T. Helten, M. Müller, G. Pons-Moll, B. Rosenhahn, and H.-P. Seidel, "Analyzing and evaluating markerless motion tracking using inertial sensors," in *3rd Workshop on Human Motion. In Conjunction with ECCV 2010.*, Sept. 2010.

[3] J. Lewis, M. Cordner, and N. Fong, "Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 165–172, ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 2000.

[4] G. Pons-Moll and B. Rosenhahn, "Ball joints for marker-less human motion capture," *IEEE Workshop on Applications of Computer Vision (WACV)*, dec 2009.

[5] D. Forsyth and J. Ponce, *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.