

( Note: we find that .rar compressed file can achieve a much smaller space than .zip file for our material, so we decide to compress the material into .rar first and pack the .rar file into a .zip file. )

## Supplementary Material Description – CVPR 2008 submission #1545

### 1. Overview

This supplementary material contains the dataset that is used in conducting the experiment presented in Section 3.2 in the paper.

The dataset contains motion capture data from the CMU Motion Capture Database (<http://mocap.cs.cmu.edu/>) and the synthesized silhouettes at different yaw angles generated by us.

The motion capture data is in the acclaim .asf/.amc file format. This is a dual file format, where the .asf file defines the static human skeleton and the .amc file defines the motion. Each subject has a .asf file, and each motion clip has a .amc file. Thus, multiple motion clips for a single subject correspond to one .asf file and multiple .amc files. For an explanation of .asf/.amc file format, see:

<http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/ASF-AMC.html>.

The silhouette data is in .yawsil format defined by us. The .yawsil file format will be described in Section 3.

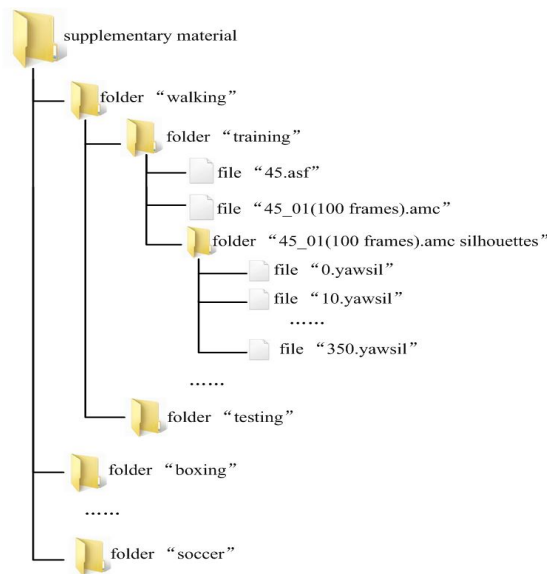
Below is the summarization of the motions in CMU database that is used in our experiment.

<b>motion type</b>	<b>training /testing</b>	<b>subject #</b>	<b>motion clip #</b>	<b>frames contained</b>
Walking (750 frames)	training (600 frames)	45	45_01	100
		49	49_01	100
		55	55_04	100
		56	56_01	300
	Testing (150 frames)	46	46_01	150
Jumping (750 frames)	training (600 frames)	16	16_06	100
		16	16_07	100
		16	16_09	100
		16	16_10	100
		16	16_11	100
		16	16_13	100
	testing(150 frames)	13	13_05	50
		13	13_19	100
Soccer (750 frames)	training (600 frames)	10	10_01	200
		10	10_02	148
		10	10_03	52
		10	10_05	100
		10	10_06	100
	testing(150 frames)	11	11_01	150
Boxing (750 frames)	training (600 frames)	13	13_17	600
	testing(150 frames)	14	14_02	150

## 2. Supplementary materials structure

The materials are arranged in folders, with each folder contains data for a motion type. Note that we use four motion types in experiment: walking, jumping, soccer shooting and boxing. **However, due to the 30 MB limit, we are only able to contain three types: walking, soccer and boxing in this material.**

Within each motion type folder, there are two subfolders called “training” and “testing”. Consistent to the names, they contain training and testing data. The “training” or “testing” subfolders contains the .asf/.amc files and corresponding silhouettes. The silhouettes are grouped in .yawsil files. Each .amc file has 36 corresponding .yawsil files (0.yawsil, 10.yawsil, ..., 350.yawsil), grouping the silhouettes according to the yaw angle. Note that the .yawsil files are contained in subfolders that have the same name as .amc file (with “silhouettes” added in the end). Below is a figure showing the supplementary material structure.



## 3. The .yawsil file format

The pixel wise storage of silhouettes take a lot of space and we store the silhouettes compactly in .yawsil files. Each .yawsil file contains the silhouettes for a .amc file at a certain yaw angle. For example, suppose the file “45\_10.amc” contains 100 frames, then its “10.yawsil” file contains 100 silhouettes at the yaw angle of 10 degree.

The .yawsil files are binary, with the bytes interpreted as follows. Note that the integers are written in low-endian order (consistent to Intel platforms). For high-endian platforms the bytes have to be swapped.

[offset (bytes)]	[type]	[description]
0000	16 bit integer	the yaw angle for this file
0002	32 bit integer	the number of frames (silhouettes)
0006	16 bit integer	image width (256 in this data)
0008	16 bit integer	image height (256 in this data)
0010	silhouettes data starts here	

The silhouettes data are stored sequentially, with each silhouette represented by a set of 32 bit integers. The number of integers that is used to represent each silhouette is different (although the silhouettes are of the same size). There is a 32 bit integer preceding each silhouette data that indicates the number of 32 bit integers that belongs to the silhouette. Below is a figure that gives an example. Here, silhouette 1 has 100 32-bit integers (equivalent to 400 bytes) and silhouette 2 has 80 32-bit integers (320 bytes). Note that the 4 bytes of indicator integer itself are not counted as the silhouette data.

100 (4 bytes)	Silhouette 1 data (32-b integer*100 = 400 bytes)	80 (4 bytes)	Silhouette 2 data (32-b integer*80 = 320 bytes)	... ..
------------------	---	-----------------	--	--------

The data of each silhouette is derived by scanning the silhouette from top-left to bottom-right (the origin of coordinate frame lies at the silhouette's top-left) and recording the pixel indices where the foreground/background property changes. This can be expressed by the following pseudo code:

**Input:** silhouette  $S$  of width  $W$  and height  $H$

**Output:** a list  $L$  of 32-bit integers as silhouette data

**begin initialize:** Boolean flat  $f = 0$ ; list  $L = \text{empty}$ ;

**for**  $y = 0 \sim H - 1$

**for**  $x = 0 \sim W - 1$

**if**  $f == 0$  &&  $(x, y)$  is background

do nothing;

**endif**

**if**  $f == 0$  &&  $(x, y)$  is foreground

$f = 1$ ;

add  $y * H + x$  to  $L$ ;

**endif**

**if**  $f == 1$  &&  $(x, y)$  is background

$f = 0$ ;

add  $y * H + x$  to  $L$

**endif**

**if**  $f == 1$  &&  $(x, y)$  is foreground

do nothing;

**endif**

**endfor**

**endfor**

**end**

The integer list  $L$  is then stored into the .yawsil file.

We also include a Matlab .m function (“*read\_silhouettes.m*”) that reads .yawsil files in this supplementary material. Although it is slow in efficiency (the nested loops are famously slow in Matlab), it works.