Metadata of Motion Data

1. Annotation to Motion Capture Dataset

Intended motion: the motion category the actor intended to convey. The usual motion includes,

run,

run/jog

run/veer left or right

run/sudden stop

walk

walk/slow walk

walk/stride

navigate (forward, backward, sideways)

walk/wander

walk/veer left or right

walk stealthily

walk/hobble

jump

jump/forward

jump/high

hop on one foot

leap

For emotion, there are eleven emotion categories as below,

amusement,

anger,

disgust,

fear,

joy,

neutral,

pride,

relief,

sadness,

shame,

surprise

Intended polarity: the polarity of the emotion the actor was trying to convey (positive/negative/neutral).

Duration: length of the motion in frames.

Speed: average speed for the left and right wrists in meters per second.

Span: average span of the motion in meters (mean distance between the wrists).

Acting task: the acting task the motion comes from (Nonverbal/Sentence/Narration).

Acting subtask: more specific task tag. The following tags are used,

nonverbal\_alone

nonverbal\_social

sentence\_narrator

sentence\_direct\_speech

tale\_blue\_beard

tale\_flower\_princess

tale\_golden\_goose

tale\_hoodie\_crow

tale\_jack\_my\_hedgehog

tale\_owl\_and\_eagle

tale\_six\_swans

tale\_swineherd

tale\_white\_duck

Gender: actor's gender (m/f)

Age: actor's age

Handedness: actor's handedness

Text: the text the actor was pronouncing/acting out during the motion sequence. For nonverbal tasks the text is not pronounced but is a general motivation for the motion.

1. RGBD Dataset (Annotation to Facial Expression Data)

Hierarchy structure

Name: ID and Gender

Identify and label 6 expressions: Angry, Disgust, Fear, Happy, Sad, Surprise

Each expression includes,

1. 3D Mesh Sequences
2. Texture Sequences
3. 3D Feature Points
4. AVI Videos

Label the usual 64 feature points by manual in every frame.

Appendix 1-- Motion Capture Data

Overview

This appendix describes Vicon based motion capture data structure. It focuses on the usual file formats.

Using Marker base MoCap devices to capture something, small grey markers are placed on it. Humans wear a black jumpsuit and have 41 markers taped on. The Vicon cameras see the markers in infra-red. The images that the various cameras pick up are triangulated to get 3D data.

This 3D data can be used in two ways:

1. Marker positions. You can be handed a file of 3D marker positions, a .c3d. This file is relatively clean - i.e., Marker “A” should be labeled Marker “A” throughout the motion. But it is your responsibility to figure out what “A” means and how it relates to the other markers.
2. Skeleton Movement Data will be handed to you as either a.vsk/.v pair or .asf/.amc pair (more on that later). The former element of the pair describes the skeleton and its joints: their connections, lengths, degrees of freedom (free, ball and socket, 2 hinges, hinge, rigid), and mathematical transformations. The latter element of the pair contains the movement data. Notes: If a subject/object was captured in multiple clips, you will be handed several .v's or .amc's. Also, something like a hamburger turner, if that's what you're capturing, can have a "skeleton" - even if it's one bone long.

The rest of this document considers only the latter option (skeleton movement). The popular Vicon software system called "ViconIQ" processes the camera data and ultimately outputs a .vsk/.v. The steps in this process are described in the three subsections following.

The Skeleton Template:

Vicon must be told what skeleton to use, in the form of a .vst, a Vicon Skeleton Template. These can be created in ViconIQ itself, under the modeling tab. The Vicon software comes with documentation on editing them. Visualized, they look like maya skeletons covered in porcupine needles. They specify the skeleton hierarchy, and what markers will be captured to help construct this skeleton. They give approximate bone lengths - the actual length, of course, will depend on the subject/object being captured.

The markers are carefully placed to get maximal information - consider that if you had a hinge joint, 2? 3? markers would define it absolutely. Constraints between markers and joints are also specified, e.g. "the elbow belongs at the y-location of this marker", or "the wrist joint is halfway between these two markers". You get the idea. Constructing .vst's for complex objects requires careful thought and testing.

The Labelling:

ViconIQ requires user interaction to start off the skeleton fitting. To process a capture, a segment of motion is loaded onscreen as a point cloud of markers. The user goes through and specifies the correspondence between these markers and the markers in the .vst, e.g. "this white dot is the clavicle marker". From this data ViconIQ can fit a skeleton and determine the skeleton's limb lengths. From here on out the labelling process is automatic. ViconIQ can load up each motion clip and automatically perform a "Kinematic Fit" of the skeleton to the markers. During this time the software uses its knowledge of the skeleton to correct captured marker aberrations. The user can also fix things up by editing the joint rotation/translation graphs directly.

The Exporting:

While this work is going on each motion clip is stored in a .trial file. When the data is clean, it is time to export useful files. A .vsk of the skeleton is exported. Keep in mind that this .vsk is unique to each person, because each person has different limb lengths. Multiple .v's are exported, one for each motion clip the person performed. Using BodyBuilder, these can be turned into asf/amc's.

Documentation on File Formats:

The File Formats - c3d, c3d files are binary; check out http://www.c3d.org/

The File Formats - asf/amc, the asf/amc format is ascii and is reasonable to parse. Angles are in euler angles.

1. What are the lengths in asf/amc files? ASF files have ":units->length" set to 0.45. That is because all the values are multiplied by 0.45 before they are stored to file (I am not sure why). Also ASF files are stored in inches, so to convert to meters you need to multiply all length values by the following scale=(1.0/0.45)\*2.54/100.0.

The File Formats - vsk/v, the .v file format is binary and is, surprisingly, also reasonable to parse. Angles are in axis-angle format (strange how you can parse them as euler angles and it still looks ok. Bang head here. See CMU-Vicon V File format). Here are some questions that came up about the format, and some possibly-correct answers (code referred to is James McCann's "C++ code" in <http://graphics.cs.cmu.edu/software/amc_viewer.r1806.tar.gz>).

1. In The .V file every bone has 6 values: T-X, T-Y, T-Z, A-X, A-Y, A-Z. Why does every bone has translational dofs? Do you know what these values represent? Are they in local coordinate system or in global coordinate system? Bone data in the .V file are local or global, depending on the file - it's in the header and depends on the options you choose when doing the File->Export in the vicon software. The code only reads global .V's.

In the .vsk, offsets in constructing the skeleton are naturally local (i.e. "move down this amount to get to the knee").

1. Does the world translation specify the position of the inboard joint? My suspicion is that the translational dofs in the V-file give the location, in world space, of the inboard joint.
2. Why does every bone has 6 values stored independently on how many dofs it has? Is it just what vicon does? Vicon just stores all 6 values. Some bones have translation, some don't. It's just what Vicon does.
3. How is rotation represented? What is A-X, A-Y and A-Z? AX AY AZ are axis-angle format. http://en.wikipedia.org/wiki/Axis\_angle
4. The position data, is it always in millimetres? Don't remember. You should read over the spec for .V files.
5. What is the reason for adding the phantom bones in the code? Phantom bones are there mostly to frighten the undergrads.

Additionally: In ASF: Store a bone start position, and its extent X.

In VSK: Store a bone start position, and its offset from its parent.

1. Not that this comes up in everyday conversation a lot, but what is the toe length? Bones at the end of kinematic chains may have arbitrary lengths since this information isn't stored in the vsk.

The File Formats – txt, To export mocap data marker positions from a .trial file to a .txt of columns of XYZ positions:

1. After labelling trajectories in ViconIQ, export .trial file as .c3d file
2. Open BodyBuilder software and open .c3d file (icon with red arrow in eclipse)
3. Click File > Write ASCII, then choose options
4. Choose options for exported markers and .txt filename

.BVH format

The name BVH stands for Biovision hierarchical data. This format provides skeleton hierarchy information in addition to the motion data. The BVH format is an excellent all around format, its only drawback is the lack of a full definition of the basis pose (this format has only translational offsets of children segments from their parent, no rotational offset is defined), it also lacks explicit information for how to draw the segments but that has no bearing on the definition of the motion.

Parsing the file:

A BVH file has two parts, a header section which describes the hierarchy and initial pose of the skeleton; and a data section which contains the motion data. Examine the example BVH file called "Example1.bvh". The start of the header section begins with the keyword "HIERARCHY". The following line starts with the keyword "ROOT" followed by the name of the root segment of the hierarchy to be defined. After this hierarchy is described it is permissible to define another hierarchy, this too would be denoted by the keyword "ROOT". In principle, a BVH file many contain any number of skeleton hierarchies. In practice the number of segments is limited by the format of the motion section, one sample in time for all segments is on one line of data and this will cause problems for readers which assume a limit to the size of a line in a file.

The BVH format now becomes a recursive definition. Each segment of the hierarchy contains some data relevant to just that segment then it recursively defines its children. The line following the ROOT keyword contains a single left curly brace '{', the brace is lined up with the "ROOT" keyword. The line following a curly brace is indented by one tab character, these indentations are mostly to just make the file more human readable but there are some BVH file parsers that expect the tabs so if you create a BVH file be sure to make them tabs and not merely spaces. The first piece of information of a segment is the offset of that segment from its parent, or in the case of the root object the offset will generally be zero. The offset is specified by the keyword "OFFSET" followed by the X,Y and Z offset of the segment from its parent. The offset information also indicates the length and direction used for drawing the parent segment. In the BVH format there isn't any explicit information about how a segment should be drawn. This is usually inferred from the offset of the first child defined for the parent. Typically, only the root and the upper body segments will have multiple children.

The line following the offset contains the channel header information. This has the "CHANNELS" keyword followed by a number indicating the number of channels and then a list of that many labels indicating the type of each channel. The BVH file reader must keep track of the channel count and the types of channels encountered as the hierarchy information is parsed. Later, when the motion information is parsed, this ordering will be needed to parse each line of motion data. This format appears to have the flexibility to allow for segments which have any number of channels which can appear in any order. If you write your parser to handle this then so much the better, however, I have never encountered a BVH file that didn't have 6 channels for the root object and 3 channels for every other object in the hierarchy.

You can see that the order of the rotation channels appears a bit odd, it goes Z rotation, followed by the X rotation and finally the Y rotation. This is not a mistake, the BVH format uses a somewhat unusual rotation order. Place the data elements into your data structure in this order.

On the line of data following the channels specification there can be one of two keywords, either you will find the "JOINT" keyword or you will see the "End Site" keyword. A joint definition is identical to the root definition except for the number of channels. This is where the recursion takes place, the rest of the parsing of the joint information proceeds just like a root. The end site information ends the recursion and indicates that the current segment is an end effector (has no children). The end site definition provides one more bit of information, it gives the length of the preceding segment just like the offset of a child defines the length and direction of its parents segment.

The end of any joint, end site or root definition is denoted by a right curly brace '}'. This curly brace is lined up with its corresponding right curly brace.

One last note about the BVH hierarchy, the world space is defined as a right handed coordinate system with the Y axis as the world up vector. Thus you will typically find that BVH skeletal segments are aligned along the Y or negative Y axis (since the characters are often have a zero pose where the character stands straight up with the arms straight down to the side).

The motion section begins with the keyword "MOTION" on a line by itself. This line is followed by a line indicating the number of frames, this line uses the "Frames:" keyword (the colon is part of the keyword) and a number indicating the number of frames, or motion samples that are in the file. On the line after the frames definition is the "Frame Time:" definition, this indicates the sampling rate of the data. In the example BVH file the sample rate is given as 0.033333, this is 30 frames a second the usual rate of sampling in a BVH file.

The rest of the file contains the actual motion data. Each line is one sample of motion data. The numbers appear in the order of the channel specifications as the skeleton hierarchy was parsed.

Interpreting the data:

To calculate the position of a segment you first create a transformation matrix from the local translation and rotation information for that segment. For any joint segment the translation information will simply be the offset as defined in the hierarchy section. The rotation data comes from the motion section. For the root object, the translation data will be the sum of the offset data and the translation data from the motion section. The BVH format doesn't account for scales so it isn't necessary to worry about including a scale factor calculation.

A straightforward way to create the rotation matrix is to create 3 separate rotation matrices, one for each axis of rotation. Then concatenate the matrices from left to right Y, X and Z.

vR = vYXZ

An alternative method is to compute the rotation matrix directly. A method for doing this is described in Graphics Gems II, p 322.

Adding the offset information is simple, just poke the X,Y and Z translation data into into the proper locations of the matrix. Once the local transformation is created then concatenate it with the local transformation of its parent, then its grand parent, and so on.

vM = vM\_child M\_parent M\_grandparent…

Appendix 2—RGBD Data (facial expression)

Depth information has been proved to be very effective in Image Processing community and with the popularity of Kinect since it is introduced. RGB-D has been explored extensively for various applications. Therefore, the need for the development of Kinect image & video database is crucial.

The effort is to create a Kinect Face database of images of different facial expressions in different lighting and occlusion conditions to serve various research purposes.

The dataset consists of the multimodal facial images obtained by Kinect. The data is captured in two sessions happened at different time period. In each session, the dataset provides the facial images of each person in several states of different facial expressions, different lighting and occlusion conditions: neutral, smile, open mouth, left profile, right profile, occlusion eyes, occlusion mouth, occlusion paper and light on. All the images are provided in three sources of information: the RGB color image, the depth map (provided in two forms of the bitmap depth image and the text file containing the original depth levels sensed by Kinect) as well as 3D.

In addition, the dataset usually comes with the manual landmarks of 6 positions in the face: left eye, right eye, the tip of nose, left side of mouth, right side of mouth and the chin. Other information of the person such as gender, year of birth, glasses (this person wears the glasses or not), capture time of each session are also available.

ACQUIRING PROCESS

The recording takes place in an indoor environment. Each person is recorded twice in two sessions happening at different time. In each session, the person is asked to perform different face states in front of the Kinect camera at a distance of around 1 meter. Images are then normalized by cropping at the size of 256x256 centered by the face.

Data Structure

The structure of the database is illustrated in the following hierarchy,

Name: Person Identifier Number, each person is determined by an identifier of 4 digits.

Info.txt: some information about the person (ID, Gender, Year of born, Glasses (this person wears the glasses or not), Capture time of each session (format: yyyy:mm:ddhh:mm:ss))

S1/2: Session 1/2 (there are 2 session: S1 and S2)

RGB: Contains the RGB images, in the .bmp format, e.g. rgb\_personIdentifier\_session\_faceStatus.bmp

Depth: Contains the Depth information, there are two formats,

DepthBMP: Contains the .bmp depth image;

DepthKinect: Contains the .txt files of the depth information from the sensor of each pixel in the original coordinates (before cropping at size 640x480), file names format:depth\_personIdentifier\_session\_faceStatus.txt

3DObj: Contains the .obj 3D model Object files, format of the file names: depth\_personIdentifier\_session\_faceStatus.obj

Mark: Contains the information of the marked points, there are three formats,

Mark3DObj: Contains the .txt files composed of the coordinates in 3D Object space, file names format: depth\_personIdentifier\_session\_faceStatus\_Points\_OBJ.txt

MarkDepth: Contains the .txt files composed of the coordinates in the original Depth image coordinates (before cropping at size 640x480), file names format:depth\_personIdentifier\_session\_faceStatus\_Points\_TXT.txt

MarkRGB: Contains the .txt files composed of the coordinates in 2D RGB image space, file names format: rgb\_personIdentifier\_session\_faceStatus\_Points.txt