Gesture detector: Usage

1. Platform

This project is aiming at detecting gestures and training user-defined gesture templates. The "gesture" here means a position OR a motion of the joints of body. The function is provided in a package for Unity3D platform. It can also be migrated to other platforms.

It reads skeleton data from MS Kinect or similar motion capture devices. To acquire skeleton data, we uses OpenNI 1.5.4 because it is an open source library. You can change it to OpenNI 2.x or MS Kinect SDK, by simply modifying the codes in the `SensorSkeletonReader` class, and maybe something that is only defined in OpenNI 1.5.4.

When training gesture templates, we need to display a 3D avatar. Unity3D is good at doing this, and that's why we used it. Unity3D is a popular platform for 3D games. It can generate programs that can run in many operation systems. If you do not want to use Unity3D, you can make use of the c# scripts in "GestureScripts" folder, and try to change the codes in the UIXXX classes, the `SkeletonAvatarController` class, and the APIs that is Unity3D specific.

2. Usage

Import "OpenNIGesture.unitypackage" into your Unity3D project, open the "testScene" scene, and copy the two folders "gesture data" and "skeleton data" to your project folder (the same level with "Assets" folder), then you can test the gesture detection function.

GestureManager.Update() checks every defined gesture in every frame. When a gesture is detected, m_gestureDetected event will be triggered. You can add your own event handler on that event, see UITrainManager.Start() and UIInfoObserver.OnNewGesture() for an example. In UIInfoObserver.OnNewGesture(), I have tried to use gestures to control the Chrome Internet browser by sending virtual key events (you need to uncomment some codes to try it out).

When initializing, the program will read "gesture data/gesture template names.txt", where the names of the gestures to be detected are listed. For each gesture to be detected, there should be an xml template file in the folder "gesture data". I have trained some common gestures, such as wave and swipe.

In a nutshell, the program splits a gesture into several key postures. When these postures are detected successively, then the gesture event is triggered. I defined two ways of triggering gesture events: 1). only trigger once when the gesture is detected. 2). Continue triggering as long as the user holds the last posture of the gesture. See the comments for GestureDetector.m_frameIntv.

3. Training new gesture templates

The program uses a semi-automatic method to train gesture templates. There are roughly 4 steps: recording skeleton data, marking key frames, training templates, testing.

a) Recording skeleton data. In the "`Observe/Record skel.`" interface, enter the file name in the text field, then press "`Start recording`". The program will start to record the angle of each joint defined in SkeletonRecorder.m_jointIdx with the interval not shorter than SkeletonRecorder.m_timeInterval. Now we can do the gesture several times in front of the Kinect. After that, press "`Stop recording`". The recorded skeleton data will be stored into a .skeleton file.

b) Marking key frames. This is the important manual step for training gesture templates. In the "`Mark key frames`" interface, open a recorded skeleton file, then you can use mouse wheel to observe the skeleton frame by frame. Now we need to decide how to define key postures in the gesture. For instance, the hand goes from left->middle->right->middle in a wave gesture, so we can define these 4 postures as key postures. A frame containing a key posture is called a key frame. Mark these key frames according to the hint on the interface, then press "`Save to file`". Now the origin skeleton is overwritten, with the key posture tags added to key frames.

c) Training templates. First, we need to do a little configuration in "skeleton data/train config.txt". Then, press "`Load train. config`", "`Train and save templ.`", and the trained xml template file will be saved in "gesture data" folder.

d) Testing. In order to test the detection accuracy, we first add the name of the gesture in "gesture data/gesture template names.txt", then go to the "`Observe/Record skel.`" interface, press "`Reimport initial gesture templates`". Now you can do the gesture in front of the Kinect and see if the program can detect it correctly.

The "`Play skel.`" Interface can "play" a skeleton data file recorded before. This is useful when you forget how a gesture was recorded.

4. Remarks

- When using Kinect, according to my experience, it's better to keep the room bright. Stand not too close to the Kinect. Make sure that the limbs to be detected in the center of the picture.
- To improve the performance of the detection, first, we should design gestures that are well distinguishable from each other. For example, do "swipeLeft" with your right hand and "swipeRight" with your left hand.
- Different user do the same gesture differently. So either train a set of template for each user, or collect the skeleton data of the same gesture from several users before training the template.
- When recording skeleton data for a gesture, do the gesture several times. And in each time, do it slightly differently from other times, such as arm higher/lower, speed faster/slower, movement larger/smaller, etc. This keeps the variety of the training samples and increases the robustness of the template.
- After training, if you think the template is too sensitive/insensitive, you can open the xml file and manually change the thresholds. The larger the thresholds, the easier the gesture will be detected, and the more false alarms. The m_thresMulPerPost parameters are multipliers for the thresholds of each key posture. The m_thresMulPerJoint parameters are multipliers for the thresholds of each joint. For example, if you think the first key posture is hard to detect, you may change the first parameter of m_thresMulPerPost from 1 to 2. There are also parameters controlling the time interval thresholds between key postures. For example, the time interval between key postures 1 and 2 should be within 0.1s±0.05s. If you change the first parameter of m_thresMulTimeIntv from 1 to 2, then it becomes 0.1s±0.1s.
- Mark key frames carefully, don't mix the tags. Choose those typical postures in a gesture as key postures. 2~4 key postures are preferable. It's also OK if you want to define a static

gesture with only one posture.

- It is recommended that the library is changed from OpenNI 1.5.4 to OpenNI 2.x or MS Kinect SDK, since the latter ones support the wrist joint.