**< WEEKLY REPORT FOR WEEK 5 >**

Name: Oh SuJin

Project: Multimodal Sensor Interfacing, Acquisition and Visualization

**I) Project Work Summary**

**Finished:**

* AcceleGlove data acquisition in java
* Use kinect for image capturing for each timestamp

**Ongoing:**

* Explore MS Kinect SDK and various sample applications to gain clear understing of its latest capabilities

**II) Tasks Assigned**

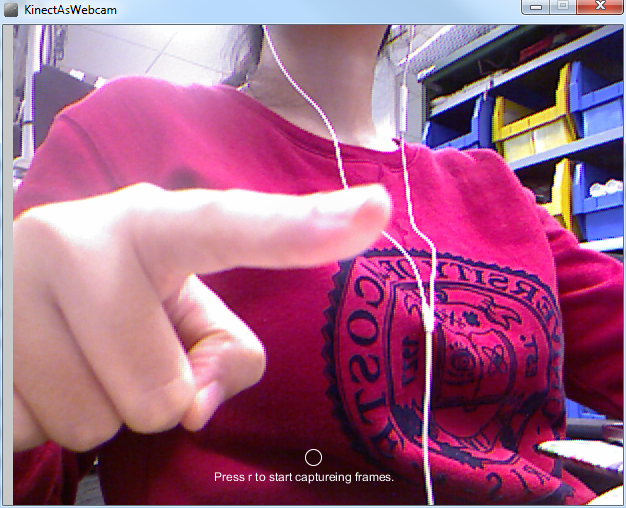
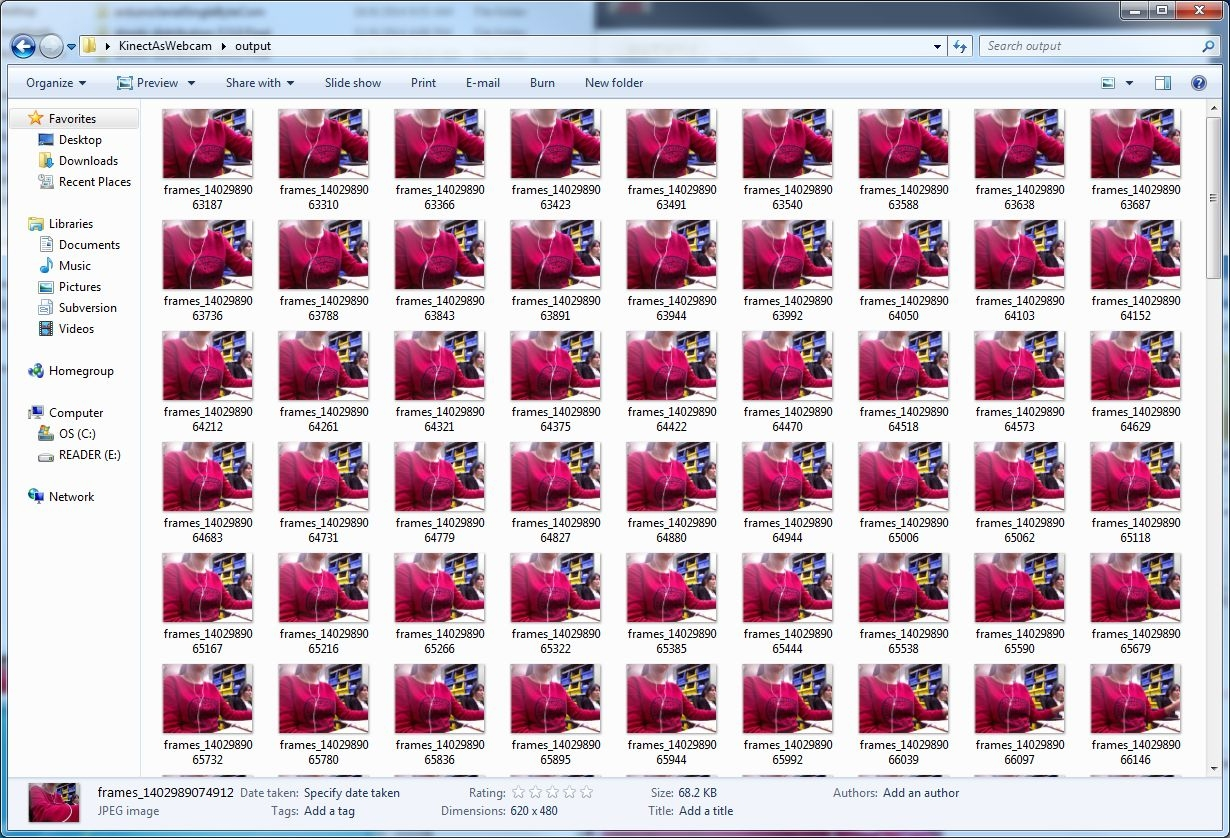
**III) Detailed Activities / Accomplishments**

**DAY1**

* Serial communication between java and Arduino(refer to JavaToArduino.ino)
* Continue studying Rules Engine

**DAY2**

* Kinect for image capturing: Using Processing 2.2+ SimpleOpenNI library, a simple video and image capturing application has been developed. Using Kinect like webcam, it does the followings:
* Once it starts running, it automatically saves the video in .oni format at KinectAsWebcam/data/video.oni.
* When ‘r’ is pressed, the application starts capturing images. Frame rate can be set as user wishes. In this project, I use 10 frames per second. Respective jpeg files are saved under KinectAsWebcam/output, while each jpeg file is named as its 13 digit unix time stamp when it is taken.

* At the same time, on the Leap Motion side, each frame data is saved with its unix time stamp. Along jpeg images from Kinect, this allows easier data comparison and analysis.
* List of exercises to be tested with this KinectAsWebcam application above.

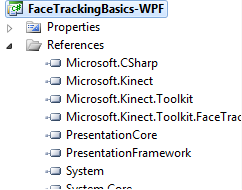
**DAY3**

* Explore various capabilities of real time Face Tracking (Kinect):
* By analyzing input from a Kinect camera, the face tracking engine can detect head pose and facial expressions and provide such information to an application in real time.
* How to set up:
* Add the following two projects to your C# solution

Microsoft.Kinect.Toolkit;

Microsoft.Kinect.Toolkit.FaceTracking;

* Also, add the two projects as references to your project:



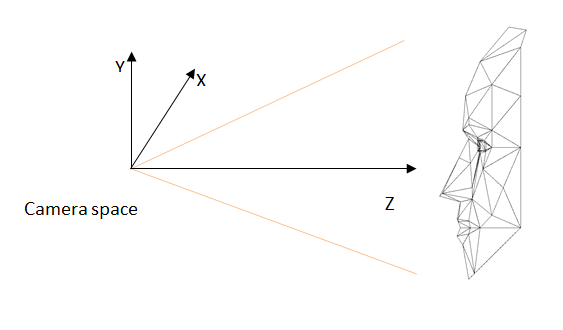
* Consider adding the following namespaces:

using Microsoft.Kinect;

using Microsoft.Kinect.Toolkit;

using Microsoft.Kinect.Toolkit.FaceTracking;

* Coordinate system: The Face Tracking SDK uses the Kinect coordinate system to output its 3D tracking results. The origin is located at the camera’s optical center (sensor), Z axis is pointing towards a user, Y axis is pointing up. The measurement units are meters for translation and degrees for rotation angles.



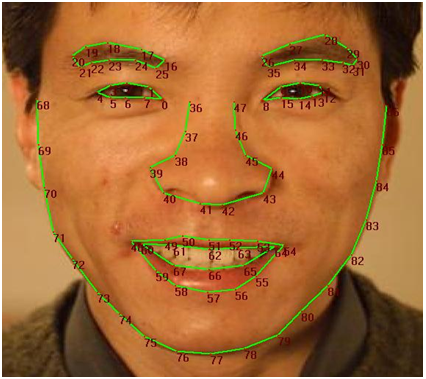
## Face Tracking Outputs:

This section provides details on the output of the Face Tracking engine. Each time you call **StartTracking** or **ContinueTracking**, **FTResult** will be updated, which contains the following information about a tracked user:

* Tracking status
* 2D points
* 3D head pose
* AUs

**2D Mesh and Points**

The Face Tracking SDK tracks the 87 2D points indicated in the following image (in addition to 13 points that aren’t shown in Figure 2 - Tracked Points):



These points are returned in an array, and are defined in the coordinate space of the RGB image (in 640 x 480 resolution) returned from the Kinect sensor.

The additional 13 points (which are not shown in the figure) include:

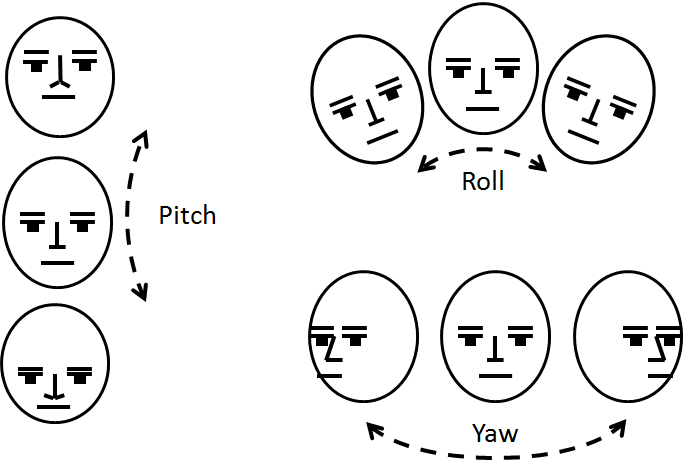
* The center of the eye, the corners of the mouth, and the center of the nose
* A bounding="" box around the head

**3D Head Pose**

The X,Y, and Z position of the user’s head are reported based on a right-handed coordinate system (with the origin at the sensor, Z pointed towards the user and Y pointed UP – this is the same as the Kinect’s skeleton coordinate frame). Translations are in meters.

The user’s head pose is captured by three angles: pitch, roll, and yaw.

**Figure 3.  Head Pose Angles**



The angles are expressed in degrees, with values ranging from -180 degrees to +180 degrees.

|  |  |
| --- | --- |
| **Angle** | **Value** |
| Pitch angle  0=neutral | -90 = looking down towards the floor  +90 = looking up towards the ceiling  Face Tracking tracks when the user’s head pitch is less than 20 degrees, but works best when less than 10 degrees. |
| Roll angle  0 = neutral | -90 = horizontal parallel with right shoulder of subject  +90 = horizontal parallel with left shoulder of the subject  Face Tracking tracks when the user’s head roll is less than 90 degrees, but works best when less than 45 degrees. |
| Yaw angle  0 = neutral | -90 = turned towards the right shoulder of the subject  +90 = turned towards the left shoulder of the subject  Face Tracking tracks when the user’s head yaw is less than 45 degrees, but works best when less than 30 degrees |

**Animation Units**

The Face Tracking SDK results are also expressed in terms of weights of six AUs and 11 SUs, which are a subset of what is defined in the Candide3 model (http://www.icg.isy.liu.se/candide/). The SUs estimate the particular shape of the user’s head: the neutral position of their mouth, brows, eyes, and so on. The AUs are deltas from the neutral shape that you can use to morph targets on animated avatar models so that the avatar acts as the tracked user does.

The Face Tracking SDK tracks the following AUs. Each AU is expressed as a numeric weight varying between -1 and +1.

|  |  |  |
| --- | --- | --- |
| **AU Name and Value** | **Avatar Illustration** | **AU Value Interpretation** |
| Neutral Face  (all AUs 0) | JJ130970.k4w_face_neutral_face2(en-us,IEB.10).png |  |
| AU0 – Upper Lip Raiser  (In Candid3 this is AU10) | JJ130970.k4w_face_upper_lip_raiser(en-us,IEB.10).png | 0=neutral, covering teeth  1=showing teeth fully  -1=maximal possible pushed down lip |
| AU1 – Jaw Lowerer  (In Candid3 this is AU26/27) | JJ130970.k4w_face_jaw_lowerer(en-us,IEB.10).png | 0=closed  1=fully open  -1= closed, like 0 |
| AU2 – Lip Stretcher  (In Candid3 this is AU20) | JJ130970.k4w_face_lip_stretcher(en-us,IEB.10).png | 0=neutral  1=fully stretched (joker’s smile)  -0.5=rounded (pout)  -1=fully rounded (kissing mouth) |
| AU3 – Brow Lowerer  (In Candid3 this is AU4) | JJ130970.k4w_face_brow_lowerer(en-us,IEB.10).png | 0=neutral  -1=raised almost all the way  +1=fully lowered (to the limit of the eyes) |
| AU4 – Lip Corner Depressor  (In Candid3 this is AU13/15) | JJ130970.k4w_face_lip_corner_depressor(en-us,IEB.10).png | 0=neutral  -1=very happy smile  +1=very sad frown |
| AU5 – Outer Brow Raiser  (In Candid3 this is AU2) | JJ130970.k4w_face_outer_brow_raiser(en-us,IEB.10).png | 0=neutral  -1=fully lowered as a very sad face  +1=raised as in an expression of deep surprise |

**Shape Units**

The Face Tracking SDK tracks the following 11 SUs in IFTFaceTracker. They are discussed here because of their logical relation to the Candide-3 model.

Each SU specifies the vertices it affects and the displacement (x,y,z) per affected vertex.

|  |  |
| --- | --- |
| **SU Name** | **SU number in Candide-3** |
| Head height | 0 |
| Eyebrows vertical position | 1 |
| Eyes vertical position | 2 |
| Eyes, width | 3 |
| Eyes, height | 4 |
| Eye separation distance | 5 |
| Nose vertical position | 8 |
| Mouth vertical position | 10 |
| Mouth width | 11 |
| Eyes vertical difference | n/a |
| Chin width | n/a |

In addition to the Candide-3 as described at http://www.icg.isy.liu.se/candide/, face tracking supports the following:

* Eyes vertical difference
* Chin width

Face tracking does not support the following:

* Cheeks z (6)
* Nose z-extension="" (7)
* Nose pointing="" up (9)

**DAY4&5**

* AcceleGlove: data acquisition in java. XYZ vector position of each finger, as well as palm position can be recorded in csv file format.

