**Iteration 5: Gesture Editing with Fflipper**

**Social gesture coding manual**

*created by*: Tina Chen, Quinn Liu, Rain Liu, Elliott Lin, Samantha Feder-Trosclair, Claire Song, Eddie Mondeja, Sadie Camilliere, Cassidy Salentine, Avery Kirschbaum, Callie Rosenzweig, Zahira Flores, Yihan Qian, Mufeng Li, Cameron Freidman, Christine St Jeanos, Anjana Lakshmi

*last updated*: Jan 2nd, 2024

The purpose of this analysis is **<blind>**

**A.** **Summary of ELAN tiers:**

**Important:** This round of coding only pertains to Gesture Content tiers. At the time of this iteration, you have finished coding your first round of gesture content tiers.

In this iteration, you will work with an ELAN file named in the format “xyzabc\_GC4\_ForFlipper.eaf” wherein “xyzabc” is a placeholder for the participant ID and “\_GC4\_ForFflipper” indicates that this has completed two levels of gesture coding and is ready for clipping with Fflipper (see manual for iteration 5).

In order to double-check the annotated gestures, in this iteration, you will be using **Fflipper** to clip out annotated gestures from this ELAN file and compare the gestures tagged for each axis alongside other gestures that were tagged for this axis.

After comparing the gestures, you will make edits to the annotations you had made in iteration 4 (gesture content and axes) to the ELAN file xyzabc\_GC4\_ForFlipper.eaf.

**B.** **Description of ELAN tiers:**

In order to run the Fflipper program, you will annotate onto a tier in the ELAN file, named as “Fflipper”.

Next, you will annotate each of the gestures you have annotated at iteration 4, using the following format: #Gesture\_content wherein # is the serial number (MUST be different for each annotation you make on the Fflipper tier), and Gesture\_content refers to the axis / axes decisions you have made for this annotation.

For example, for the first annotated gesture in the timeline (say it is a Left gesture), you may annotate 1Left. Then, the next annotated gesture (say it is an Up-and-Down gesture) may be 2UpDown. The numbering is to ensure you can reference back to it when needing to edit, but it is also an absolute necessity for the Flipper App that you will be introduced to, in the next paragraph.

**C.** **Installing Fflipper (First-time user):**

Go to the Fflipper website:<https://github.com/jonkeane/fflipper>, specific instructions are shown in the README.md. Here is a summary of the installation process:

1. To install on Mac OS X, go to Applications > Terminal to type commands.

**Note**: For commands, type out (or copy/paste) the exact text and press enter. Some installations take a while, don’t do the next one until you see a dollar sign ($) appear

2. [Install Homebrew](https://brew.sh/) and then [Install Python2.7](https://www.python.org/downloads/release/python-2718/)

3. Install tk from brew: **brew install tcl-tk**

4. Install ElementTree for python

use the command:

**python**

and then in the new line that appears type:

**import xml.etree.ElementTree as ET**

5. Download fflipper, pyelan, and clipper.

Move these into the fflipper folder, and make sure that each folder is named just pyelan, flipper, and clipper respectively (ie remove the -master, if you downloaded them from GitHub as archives.)

6. Also install ffmpeg

**brew install ffmpeg**

**D.** **Using Fflipper:**

Use the command **python /path/to/fflipper/fflipper.py** (or flipper-master/flipper.py if you didn’t change the folder name) in the terminal to run Fflipper, remember to change the /path/to part to the location of Fflipper on your machine (that is, delete /path/to/).

**How to find the path:**

type: **python**

Then press space, and using Finder, find the flipper.py file in the flipper-master folder you downloaded and drag and drop it into the terminal window. Then the full ‘path’ should appear, and you can just press enter

Example of command: python desktop/flipper-master/flipper.py

**Additional information**:

You will use serial numbers here because Fflipper needs you to give each annotation a string as a name (probably a unique string).

You can choose to “append the tier name to the annotation” in making the clip file names. You will need to pick the folder you want to save to.

When clipping, nothing seems to be happening in the actual app itself (the box saying progress does nothing >.>) you can watch the progress in the terminal window and will see the clips appear in the destination folder. It’s not a fast program.

You know it is finished when you look at the terminal and see the last line is like:

“Resetting to the next batch of processes.”

**E.** **Comparing Videos**

Create subfolders titled “UP”, “DOWN”, “UPDOWN”, “VERTICAL\_DISTANCE”, “LEFT”, “RIGHT”, “LEFTRIGHT”, “HORIZONTAL\_DISTANCE”, “IN”, “OUT”, “INOUT”, “SAGITTAL\_DISTANCE”, “COMPLEX”, “UNCLEAR”, “GESTURE2”, and “IDIOSYNCRATIC” within the folder that has the clipped participant videos.

Move clipped videos into appropriate folders. If a gesture spans more than one axis, choose which subfolder you’d put it into based on which axis seems intentional in that gesture. If both seem equally intentional, load it into the “COMPLEX” subfolder. Note at this point in time, subfolders “UNCLEAR”, “GESTURE2” and “IDIOSYNCRATIC” will be empty.

Next go to each subfolder and view the videos in that subfolder one after the other. First go through them quickly to get a sense of patterns within the participant. Then one by one. **Ask yourself whether**:

1. **This is indeed a gesture of primary interest to this study**. Note that all but for IN gestures tend to be linear, and even IN gestures tend to be curvilinear but not circular. There are usually no closed shapes (ex. triangles) / symbols (ex. finger quotes) involved in gestures of primary interest to this study. If you find one of these, move the file to the “GESTURE2” subfolder.
2. **This gesture indeed belongs in this subfolder and not any other subfolder**. If you feel it belongs in another subfolder, move it to the subfolder. Note that it doesn’t matter what you previously annotated this gesture as. If you strongly believe this is a different axis, move it. If unclear, move it to the “UNCLEAR” subfolder.
3. **This gesture is a clear enough signal given how this participant tends to perform with this particular category of gestures**. If it is as clear in intention as some of the more clear and deliberate gestures in this subfolder, leave it be. Else move it to the “UNCLEAR” subfolder.
4. **An independent coder (who did not do Level-1 gesture coding with you) would also place this gesture in this subfolder**. If you feel strongly that they will, leave as is. Else move it to the “UNCLEAR” subfolder.
5. **This gesture (especially if it looks like a Beat / PUG) is performed consistently across the participant video or only sections of the participant video.** You can tell from the serial number on the file name – does it span across serial numbers? If so, ask yourself whether this still belongs in this subfolder or is an idiosyncratic gesture the participant makes. If you strongly believe this video belongs to this subfolder, leave as is. Else, move it to the “IDIOSYNCRATICS” subfolder.

**F.** **Making final edits and saving output**

Use the final files in these subfolders to make edits to annotations in the xyzabc\_GC2\_ForFlipper.eaf ELAN file. You may delete some entries because you found them unclear and you may edit some entries from one axis to another.

You may also add annotations to the GESTURE PRESENCE 2 tier if you identified any. Inform the graduate student coder so this can be tracked (since GESTURE PRESENCE 2 was a Level-1 gesture code and not Level-2.

Save the ELAN file named in the format “xyzabc\_GC4\_ForReliability.eaf” wherein “xyzabc” is a placeholder for the participant ID and “\_GC4\_ForReliability” indicates that this has completed two levels of gesture coding and is ready for reliability checks. This should save as a “.eaf” file because it is an ELAN file. Upload this file to your individual folder on Box.