**Guide for Using the Whisker Behavior Video Processing Pipeline**

**Before Start**

* Make sure you have MATLAB **R2014a** for maximal compatibility.
* Make sure **packages** folder and its subfolders are added to MATLAB search path.
* Make sure **Many functions** folder and its subfolders are added to MATLAB search path.

**General Organization of the Pipeline**

**Pipeline\_script.m has four functionalities separated by sections**

* Pipeline batch processing (based on ManyWhiskers class) whisker linking, bar detection, face masking, physical quantities computation, and contacts detection.
* Process single trial (based on OneWhisker class).
* Use preprocessed training and testing data to create and benchmark Random Forest classifier (based on ContactTrainer class).
* Create image templates for bar detection (based on Bartender class).

**ManyWhiskers class**

Before running

* Make sure you have the **session\_dictionary.xlsx** Excel file containing all necessary session names with corresponding whisker ID of interest and distance (in pixel) of facemask to face. For trials with more than one whisker of interest, you can duplicate the row and assign different values to each. The pipeline will retrieve all information but only process the first listed whisker.
* If you need to do bar detection, make sure the **pole\_template.mat** has been created or obtained, otherwise set ‘bar’ property to false in the constructor (explained in next section).
* If you need to do contact detection, make sure the **classifier.mat** is obtained, otherwise set ‘contact’ property to false in the constructor (explained in next section).
* A default version of all the above files can be found in **Default session info** folder in the Data3 NAS. This default pole\_template.mat was obtained from training dataset and the default classifier.mat is an empty dummy file at this point.

Running from scratch

* Construct and initialize a ManyWhisker object. In the constructor, parameter-value pairs can be used to set which step of processing (e.g. ‘bar’, ‘facemasks’, ‘physics’) will be carried out.
* Use Start() method to start the pipeline. However, you may also want to navigate into the variable, mw.pipressObj.pipress, with MATLAB Variable window to verify the details before start.
* The record of pipeline progress, pipress, will be automatically saved to the main folder at the end of processing. If you used Ctrl+C to abort the execution, the current progress will be lost and nothing is saved.
* The pipress.mat contains all information about the pipeline including progress, errors, diagnosis, etc., for inspection and/or debugging. The pipress.xls contains only trial information and the progress portion of pipress.mat and is for you to modify interactively (see next section).

Running based on pipress

* Keywords in pipress

‘found’: The file is found although the validity cannot be guaranteed.

‘computed’: The processing was done and the result passed the quality control (if any) and was saved.

‘error: …’: MATLAB exception occurred or the result failed the quality control. In either case you can find detailed information in the corresponding report column (\*\_r).

‘none’: The processing is yet to be executed because not all necessary data is available.

* Modifying pipress.xls

Using Excel, you can easily change any status entry to ‘redo’ to force the pipeline computing that specific step of that specific trial again. Note that ‘redo’ will remain if the batch processing failed to reach it. Keyword ‘skip’ (or ‘skip: …’) can be used to ignore certain step of certain trials.

An error will occur when MATLAB accesses pipress.xls under use. So, do not keep it open.

Run again

* If you use pipress generated by another computer, the file paths for localizing the data may not be valid on your computer (unless data files are specified by relative network paths, e.g. \\OCONNORDATA2\data2, which is preferable and recommended). Try substituting part of the paths in Excel when feasible. However, if the folders of interest are scattered around you can “reorganize” them as if they are being “copied” by creating hard directory junctions in file system (in command window, type “mklink /j <Link> <Target>”).

**OneWhisker class**

* See “Single trial processing” section in pipeline\_script.m for an example of use.
* Essential properties in an OneWhisker object

**checkTable:** It keeps track of the progress including the status and detailed information of each processing step. A later step will come to this table to check if all necessary data is ready. Therefore, you have to run methods in a reasonable order. However, you do not have to force each method to compute the result again (see ‘Force’, true/false parameter pair below).

**objStruct:** Every steps of processing are also object-oriented and all objects created along the way are saved in this structure variable (e.g. objStruct.measurements, objStruct.facemasks, etc.). As a result, you have the access to a lot of handy methods associated with them. However, an object will not be created if the step is not actually computed during the current run (see ‘Force’, true/false parameter pair below).

**Other properties:** Refer to comments in code following the property definition.

* Parameters of methods

**‘Force’, true/false parameter pair:** It specifies whether or not to force re-computing the subject. If set to false, the method will try to find existing file and only start to compute it when the file cannot be found.

**DetectBar('PosImgDir', dirPath):** Since the quality of bar detection cannot be evaluated automatically, an image marked with the detected pole center will be saved in the folder specified by dirPath.

**See comments in the code for explanations of more method specific parameters.**

* If you are only interested in certain processing step, such as making facemasks, it is usually easier to use individual processing modules (classes) to achieve that (see next section).

**Individual Processing Modules**

**WhiskerRegistration class**

**Bartender class**

**Masquerade class**

**WhisPhys class**

**ManyContacts class**