ReadMe – Motion analysis with MATLAB UIs

Content

[1. Overview – What is this file about? 1](#_Toc534048530)

[2. video\_gui.mlapp 1](#_Toc534048531)

[1) Purpose/Problem statement: 1](#_Toc534048532)

[2) Manual: 2](#_Toc534048533)

[3) Results: 3](#_Toc534048534)

[3. Motion\_bin\_gui.mlapp 3](#_Toc534048535)

[1) Purpose/Problem statement: 3](#_Toc534048536)

[2) Manual 4](#_Toc534048537)

[3) Results: 4](#_Toc534048538)

# Overview – What is this file about?

This file gives instructions on how to utilize the two UIs *video\_gui.mlapp* and *motion\_bin\_gui.mlapp* to extract the body motion of a mouse that participated in the multisensory experiment.

# video\_gui.mlapp

## Purpose/Problem statement:

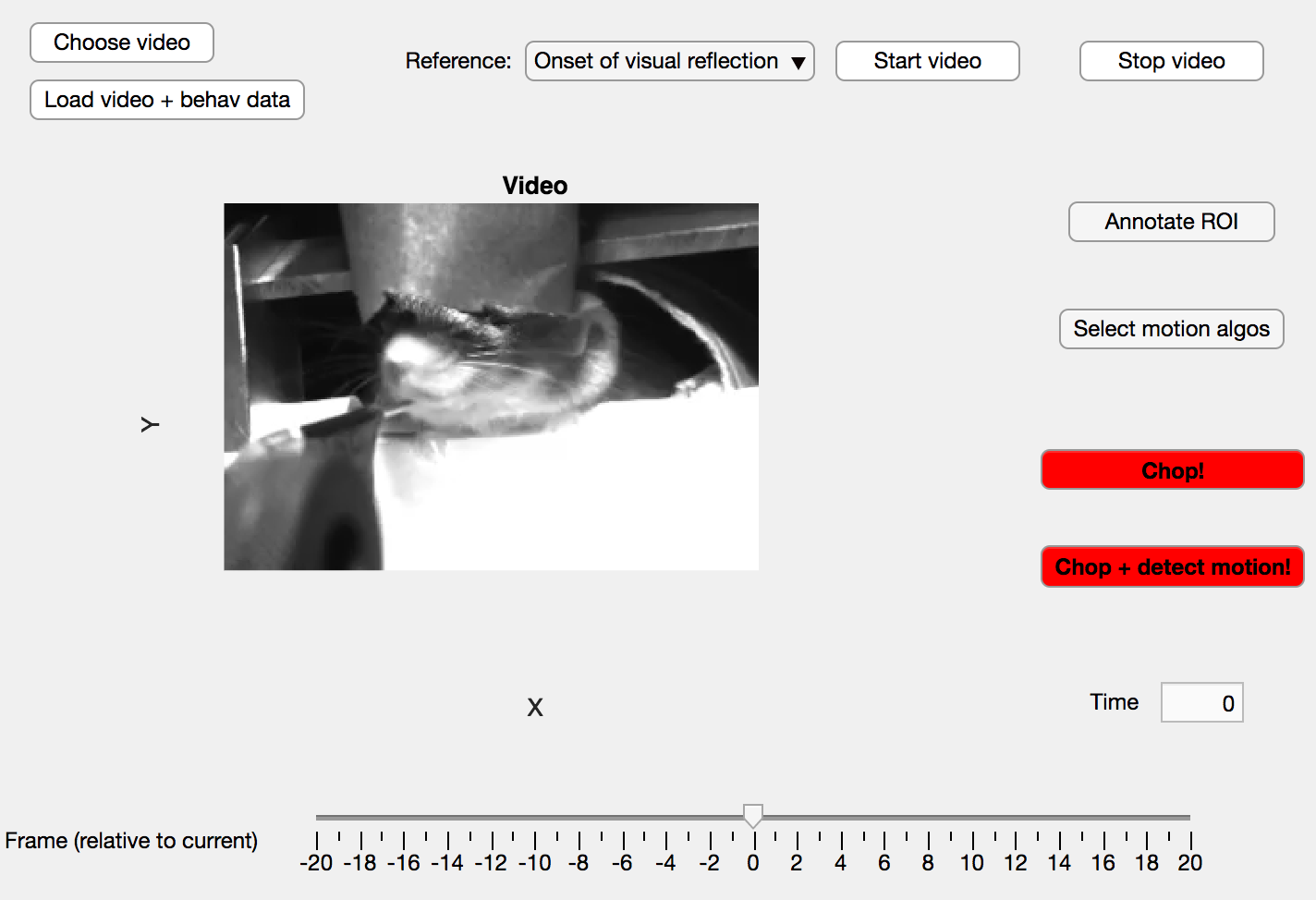
The mice performing the experiment were video-recorded by a static camera for the vast majority of sessions. The produced video files were huge in size (raw up to 100GB per session) and even the compressed versions were heavy to analyze. However, the information about motion about particular limbs, the head or the trunk of the mouse may induce a bias in the widefield (calcium imaging) recordings, in particular in the motion-related areas like M1, SMA etc. To investigate whether or not certain brain regions are activity-modulated by the presence or absence of motion of certain body parts was one of the objectives of this project.

So the purpose of this UI is two-fold; first the video is synchronized with the widefield data and saved separately for every trial and secondly, the motion analysis is performed on the video of every trial.

Various facts hamper the motion analysis of the recorded data, including 0) missing synchronization with the widefield data, 1) varying camera position (across sessions), 2) occlusion of certain ROI across entire sessions, 3) video quality and 4) varying lighting conditions. Hence, the analysis is semi- but not fully-automatized.

The UI is able to run a motion analysis for an experimental session specified by the user. The user has to draw and name the ROI into the video and can select which algorithms are used for motion extraction.

## Manual:



This is the default window of the *video\_gui* interface. The buttons are designed for the user to operate from left to right and top to bottom.

1. **Choose video:** The user clicks on the button, a window of the J: drive pops up (this the default but can be changed) and the user can navigate through the explorer to select a video of interest.
2. **Load video + behave data:** MATLAB attempts to open the selected video, and if this is successful, a preview of the first frame is displayed (see picture). Additionally, the H: drive is searched for the behavioural files of the respective session (e.g. the stimulus and response type of each trial).
3. **Reference:** Synchronization of the video with the imaging data is crucial for all later analysis. For the future, I clearly recommend synchronizing both devices *before* the data analysis (e.g. by connecting the hardware devices and let the video start automatically upon receipt of a time-stamp signal from the widefield device or so). This synchronization was the trickiest part of this UI and it is still error-prone as it relies on the user inserting accurate data.  
   Synchronization is performed by a visual cue present in the video, either “*the onset of visual reflection*” or the “*offset of the stage drive-in*”. The user has to choose this option according to the presence of the cue in the video. Generally, *visual reflection* is preferable, however as the horizontal metal bar that reflects light from the video screen is not visible in this video, the *stage drive-in* needs to be chosen.
4. **Start video:** Next, the user starts watching the video. The box **Time** will indicate the duration of the video up to the current frame (in seconds). The user has to watch carefully(!) the video until he/she detects the visual cue. Oftentimes, nothing interesting will happen for the first 10-30 seconds because the video recording was started prior to the experiment start. However, the onset of the visual reflection indicates the activation of the visual stimulus for the mouse which has a timestep in the widefield data. Similarly, the offset of the stage drive-in is indicative of a somatosensory trial. These timesteps are then compared to the timestep of the first widefield frame to synchronize video and widefield data.
5. **Stop video:** The user has to press this button to stop the video as soon as the visual cue was detected.
6. **Frame (relative to current):** This slider helps the user to select the particular frame that best corresponds to the reference cue. It is important to choose this frame with highest precision, because all posterior motion analysis depends on it.
7. **Annotate ROI:** Clicking this button will start an interactive guide that helps the user to draw up to four ROIs (head, left paw, right paw, back) into the video. This process should be self-explanatory.
8. **Select motion algos:** Here, the user can choose which algorithms should be used to detect motion. The user can select from:
   1. **Absoute difference** This computes the absolute difference of all pixels within a given ROI between time and time . Then, the mean is taken to get one value for the ROI.
   2. **Pearson R** This computes the pearson correlation coefficient between all pixels within a given ROI between time and time .
   3. **Lucas Kanade** This computes the optical flow between all pixels within a given ROI between time and time using the Lucas Kanade method. The directional information is discarded, only the motion magnitude is used and averaged across all pixels in the ROI. The noise threshold is set to 0.009.
   4. **Horn-Schunck** Similarly like Lucas Kanade, but Horn-Schunck optical flow method is applied.

I recommend recording all of them, so it can be decided at a later point which one is used for further analysis. Speed detriments should be minor.

1. **Chop / Chop + detect motion:** After pressing this button, MATLAB starts the synchronization and chops the video into small videos (one per trial). Generally, the lower button should be used since chopping and motion detection can be done in parallel easily. It is however slower than not running the motion detection and if for some reason only the chopping is required, the user can user the upper red button.

## Results:

The UI will leave a folder called “*chunked*” inside the folder of the original video. *Chunked* contains a series of videos (one per trial).

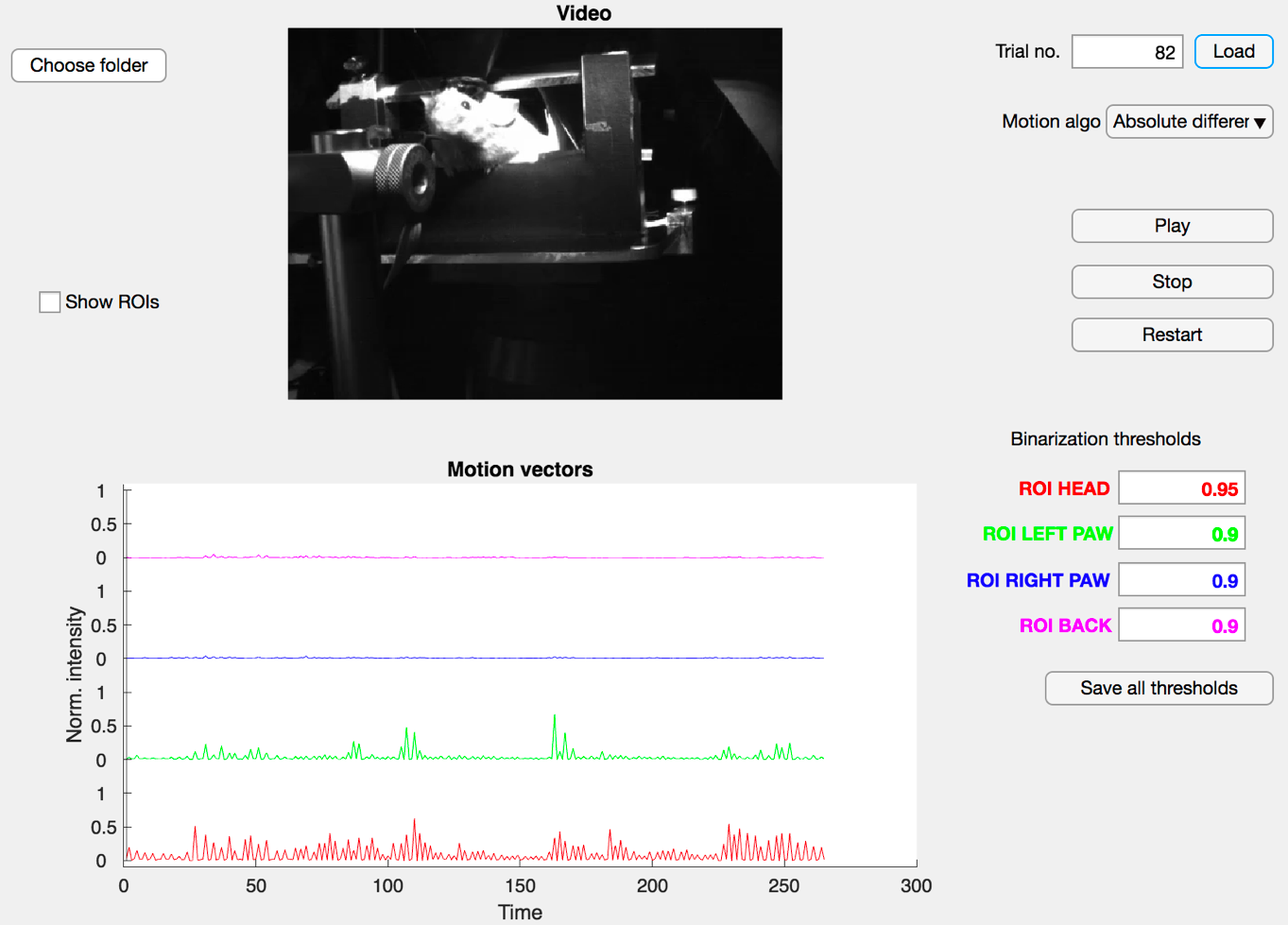
Additionally, a struct *motion* will be saved with the title *motion\_analysis.mat.* It contains all relevant data, extracted for motion analysis, including the list of ROIs, the list of used motion detection algorithms, the motion matrix (of shape *num\_rois x num\_motion\_detection\_algos x num\_frames*) and vectors that indicate the stimulus and response type of each trial. The *bounds* vector indicates which frames of the motion matrix correspond to which trial.

# Motion\_bin\_gui.mlapp

## Purpose/Problem statement:

*Video\_gui.mlapp* left us with vectors of continuous motion values, one per motion detection algorithm. To have an easier time during the analysis, it may be beneficial to operate on binary rather than continuous motion values. This UI allows the user to perform a binarization separately for every motion detection method and each ROI. The former is necessary, because each method has different semantics (e.g. high pearson correlation indicates little motion, whereas high absolute difference indicates high motion) and the latter is necessary because different lightning conditions (illumination, occlusion, contrast) may apply to different ROI.

## Manual



This shot shows the *motion\_bin\_gui* in action.

1. **Choose folder:** When clicked, the user can navigate through the explorer to a particular session and select the folder called *chunked* that was produced by *video\_gui.mlapp.*
2. **Trial no. / Load:** Here, the user can chose a particular trial to get a feeling of a suitable binarziation threshold. Before pressing Load, the user should select the desired motion algorithm from the dropdown menu. After pressing *Load*, the first frame of that trial will be displayed in the upper panel and the motion time series of all detected ROIs will be displayed in the lower panel (mind the color code).
3. **Play/Stop/Restart:** Can be used to compare the raw video to the results of the motion algorithm
4. **Show ROIs:**  Overlays the ROIs with the raw video to make the spatial location of the ROI apparent.
5. **Binarization thresholds:** Here, the user inserts the binarization thresholds [0,1], one per ROI. The user may want to investigate multiple trials to get a feeling about the distributions, however the same threshold will be applied to all trials.
6. **Save all thresholds:** Clicking this button produces a preview of the binarization wherein the colored lines are replaced by binary dots. A window with a warning pops up and the user has to confirm the choice prior to the actual binarization.

## Results:

The last step will produce a struct called *Motion\_results\_[motion\_algo\_name]* that is saved in the directory of the particular session (i.e. the same directory like the *Chopped* folder).

Author: Jannis Born, jborn@ethz.ch