EyeTracker User Manual

Version 1.2.3, 20221109

By Dr. Jorrit Montijn

Circuits, Structure and Function group

Netherlands Institute for Neuroscience

E-mail: j.montijn@nin.knaw.nl

Contents

Overview

This toolbox was created to work with SpikeGLX to detect the location and size of a pupil. It will run online pupil detection during video acquisition, and automatically synchronize videos by creating a lookup table that logs pairs of SpikeGLX timestamps with video frame numbers. It also provides easy functionality to extract light pulses in the video that can be used to provide zero-lag synchronization between video frames and light pulse onsets (for example, using Acquipix). However, you can also use this toolbox as a standalone recording program without SpikeGLX or Acquipix.

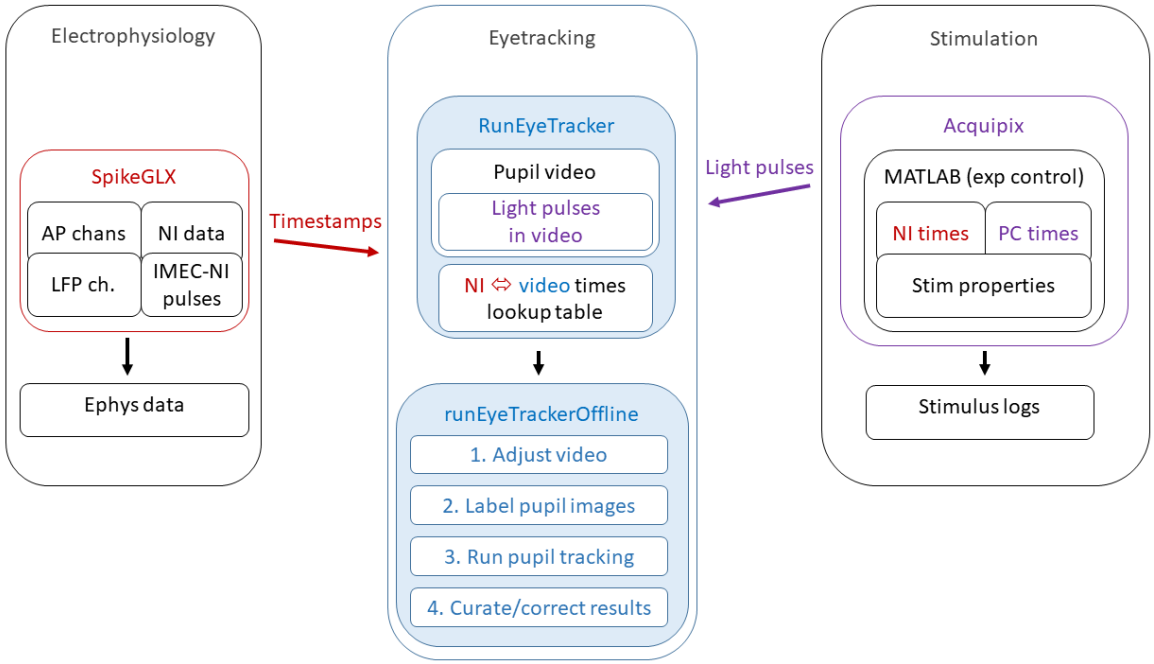


Figure 1. Data flowchart. The program RunEyeTracker saves a video and logs SpikeGLX/video frame pairs for reliable synchronization. After finishing a recording, you can use runEyeTrackerOffline to adjust pre-processing settings, such as video brightness, gamma, blurring, etc. You can then either manually set the eye-tracking parameters, or label a set of images that will train the algorithm and pick the optimal parameters using an initial grid search, followed by gradient descent. After the parameters have been set, you can run the pupil tracking, and finally curate and/or correct the final results.

Installation instructions

Before you start:

1. Make sure your computer is up-to-date, and has compatible (not necessarily the newest) CUDA drivers for your Nvidia GPU. If you don't have an Nvidia GPU, you can still use the eye tracker, but many GPU-accelerated functions will be slower.

Installing the external libraries

1. Install the required interface to stream your camera's data into matlab. If your camera works with the genicam interface, you can find the install package in the root directory.

Installing the eye-tracking repository and its dependencies:

1. Download the EyeTracker repository at https://github.com/JorritMontijn/EyeTracker
2. Download the GeneralAnalysis repository from https://github.com/JorritMontijn/
3. Optional: If you wish to combine the eye-tracker with the neuropixels data pipeline, download SpikeGLX from https://billkarsh.github.io/SpikeGLX/ and follow the installation instructions.

Workflow summary:

To run an experiment:

1. Start SpikeGLX and begin a new acquisition – but do not start the recording yet!
2. Start "RunEyeTracker" in MATLAB
3. Adjust the camera, lighting, and settings to get good online pupil detection
4. Click "Record" in the eyetracker or start the recording in the spikeglx, the eyetracker will then start automatically (as long as you have auto-start enabled)

To pre-process your video:

1. Start "runEyeTrackerOffline" in MATLAB and compile your data library
2. Select the recording you wish to preprocess
3. Adjust your pre-processing parameters

4a. Manually determine your detection settings, or

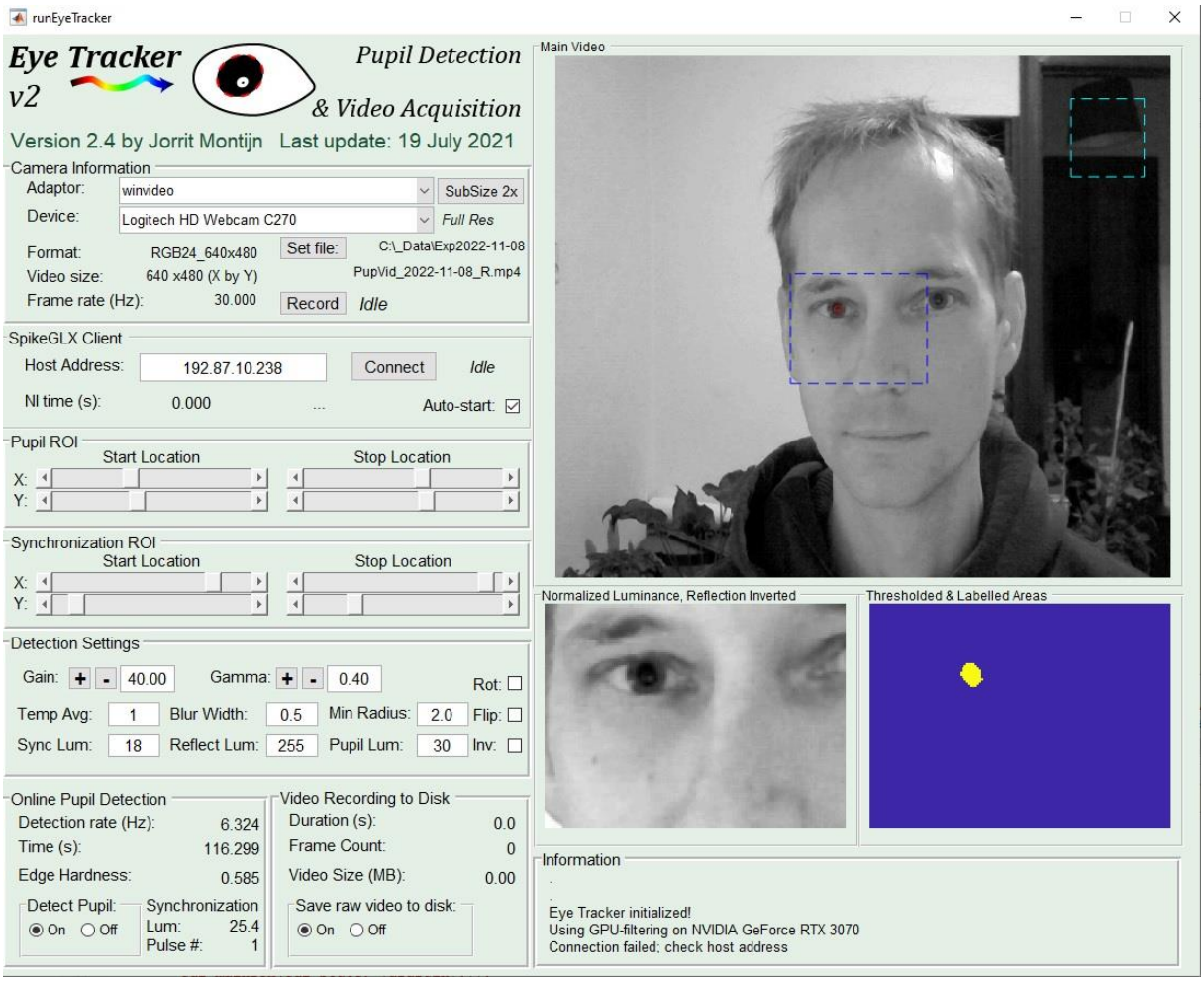
4b. Label images & optimize your settings

5. Run the pupil tracking
6. Curate your results and correct any errors

User guide

Recording a video

To record a video, simply enter “RunEyeTracker” in matlab, and the following will appear (Note: product does not include weary post-doc).



General settings

The program will automatically try to connect to SpikeGLX at the host address that used last time. If it cannot connect to SpikeGLX, it will say “Connection failed” after a 10-second timeout, but you can continue using the program otherwise. If it has connected to SpikeGLX, it will show the recording time and name according to SpikeGLX. If you have “Auto-start” checked, the Eye Tracker will start recording automatically when you start your SpikeGLX recording, so you cannot forget to turn on the eye-tracker. The program will automatically create a date-stamped name for your recording, but you can also set the output file manually.

Detection settings

You can select two regions of interest (ROIs): one for the pupil (blue) and for the synchronization light pulses (cyan). As you can see in the picture above, the pupil detection is only applied to the ROI. The larger the ROI, the slower your online detection will be. Note that the online pupil detection is a slimmed-down version of the offline pupil detection algorithm (only detects circles instead of ellipses, less pre-processing, etc). Generally speaking, if the online detection works 95% of the time,

your offline pupil detection will be fine. However, if the online pupil detection is hopping around the image from frame to frame, then you may need to adjust your camera settings. These include:

- Camera gain: adjusts the dynamic range of the camera. **Note: this will affect your raw video!**
- Camera gamma: adjusts the gamma of the video (dark/light mapping unto pixel brightness). **Note: this will affect your raw video!**
- Rot: rotate video 90 degrees. **Note: this will affect your raw video!**
- Flip: flip video upside-down. **Note: this will affect your raw video!**

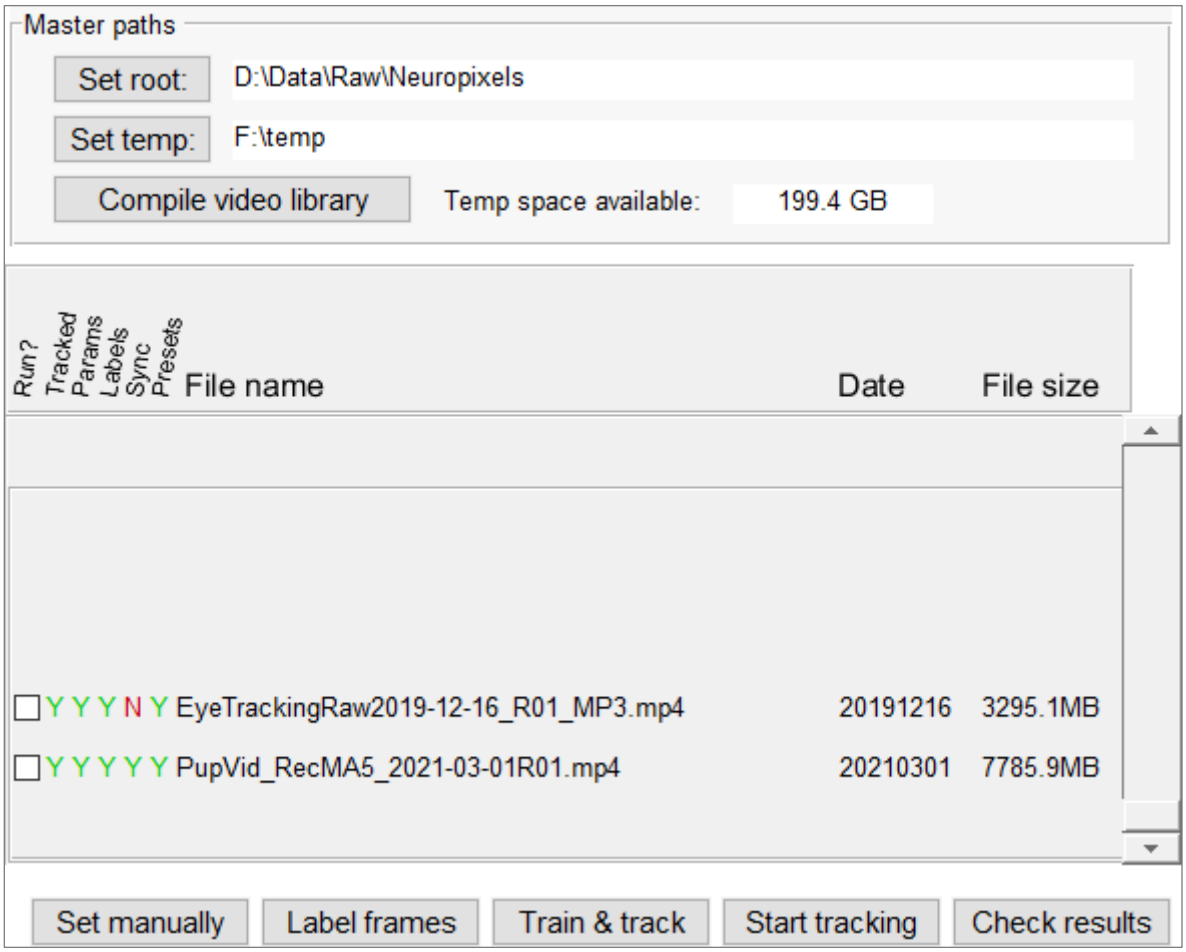
There are also some detection parameters that you can adjust that will not affect the raw video. These parameters are saved with the output video, however, so if you load the recording later it will automatically import these settings. You can always change them later if you wish.

- Inv: if this box is checked, it will invert the brightness values of the video before detecting the pupil. You can use this to detect pupils that are brighter than the background, for example when using two-photon calcium imaging, or sometimes with optogenetics.
- Temp Avg: temporal averaging of frames, i.e. if Temp Avg is set to "2", then every pair of two frames is averaged, and detection is only applied to the averaged image. This may help when your video is noisy, but it will also lower the frame rate and may induce strange artifacts with rapid pupil movements, such as the pupil becoming smeared-out as it moves frame-by-frame. Note that the raw video will be saved at the normal native framerate. This averaging is only applied to the pupil detection.
- Blur width: sd of Gaussian filter to spatially smooth the video (in pixels)
- Min radius: minimum pupil radius (in pixels). This setting only affects the online algorithm.
- Sync Lum: reversal threshold between dark/light pulses to synchronize the video to stimulus onsets (range: 0-255 pixel brightness)
- Reflect Lum: threshold value to remove pixels if they are brighter than this (range: 0-255 pixel brightness)
- Pupil Lum: threshold value that labels pixels as "potential pupil" when they are darker than this (range: 0-255 pixel brightness)

Preprocessing your data

Compiling the library

When you type “runEyeTrackerOffline” in the matlab prompt and click “Compile video library”, you will see a window like below:



In this example, there are two recordings, and the list displays the name, date, and file size of the movies. It also shows the following information for each movie:

- Tracked: green Y if the pupil has been tracked for this movie, otherwise it will show a red N
- Params: shows whether the tracking parameters have been set
- Labels: shows whether a training set has been labelled
- Sync: shows whether a synchronization lookup-table is present
- Presets: shows whether parameters were found that were set during recording for the online pupil tracking algorithm

You can select a movie by checking the box that says “Run?” and then clicking on one of the following buttons:

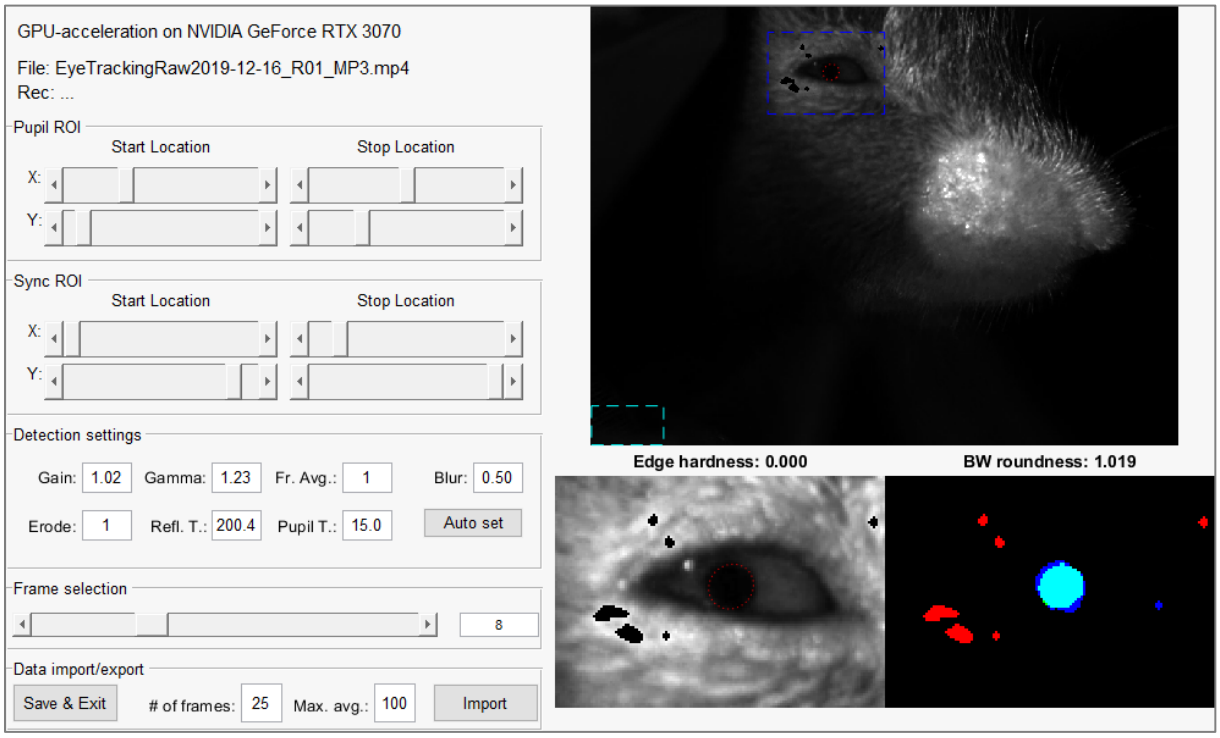
- Set manually: will open the tracking GUI where you can adjust the ROIs, change the video pre-processing settings, and set the parameters for pupil tracking
- Label frames: you can create an image training set by labelling the pupil in 25 images taken from the entire recording
- Train & track: after labelling, you can select multiple recordings in the list, click this button and go home (or on vacation, depending how many videos you have). The program will then

optimize the parameters for each recording and perform pupil tracking of all movies you selected.

- Start tracking: same as above, but without the training step in case you have already set the tracking parameters you wish to use
- Check results: this will open the TrackerChecker GUI so you can curate the data, confirm blinking epochs and correct any errors.

Setting the variables and parameters

You can set the parameters for the pupil detection in a dedicated GUI, or you can run the automated parameter selection after labelling pupils in example images (see next section).



Here can adjust the ROIs, as described in the online pupil detection section. You can also adjust the following parameters that are used by the offline pupil detection algorithm:

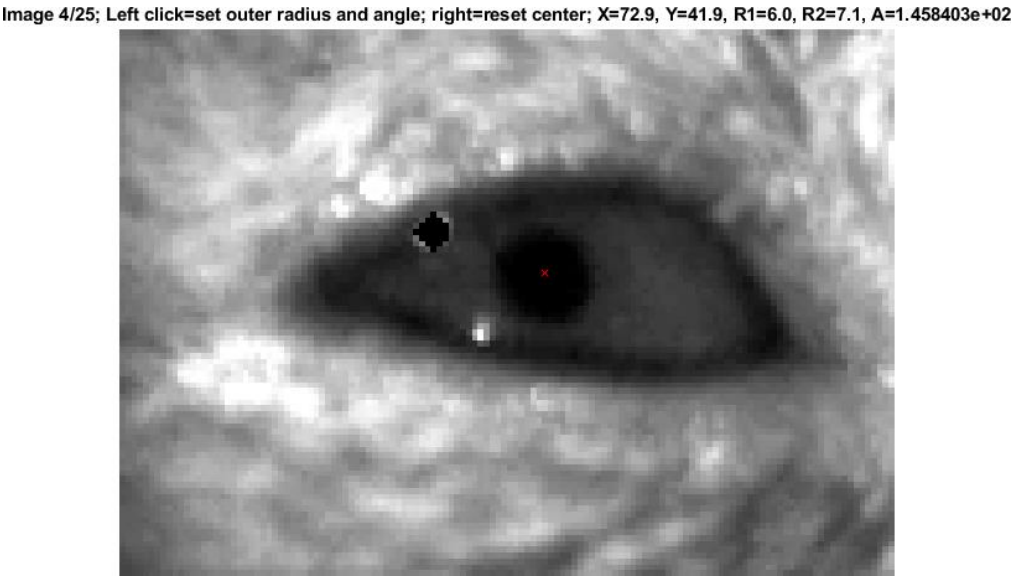
- Gain: change the video brightness (value is a multiplicative factor)
- Gamma: change the video gamma
- Fr. Avg.: frame averaging before detection
- Blur: sd in pixels for spatial Gaussian smoothing
- Erode: border erosion strength to remove small specks of potential pupil areas
- Refl. T: threshold brightness above which pixels are removed (red areas in bottom right)
- Pupil T: threshold brightness below which pixels are marked as potential pupil regions (blue areas in the bottom right)

The detected pupil is shown as a red dotted ellipse in the top raw image view, lower left pre-processed image, and as a cyan region in the bottom right view that shows the marked areas (reflection: red, potential pupil: blue, detected pupil: cyan).

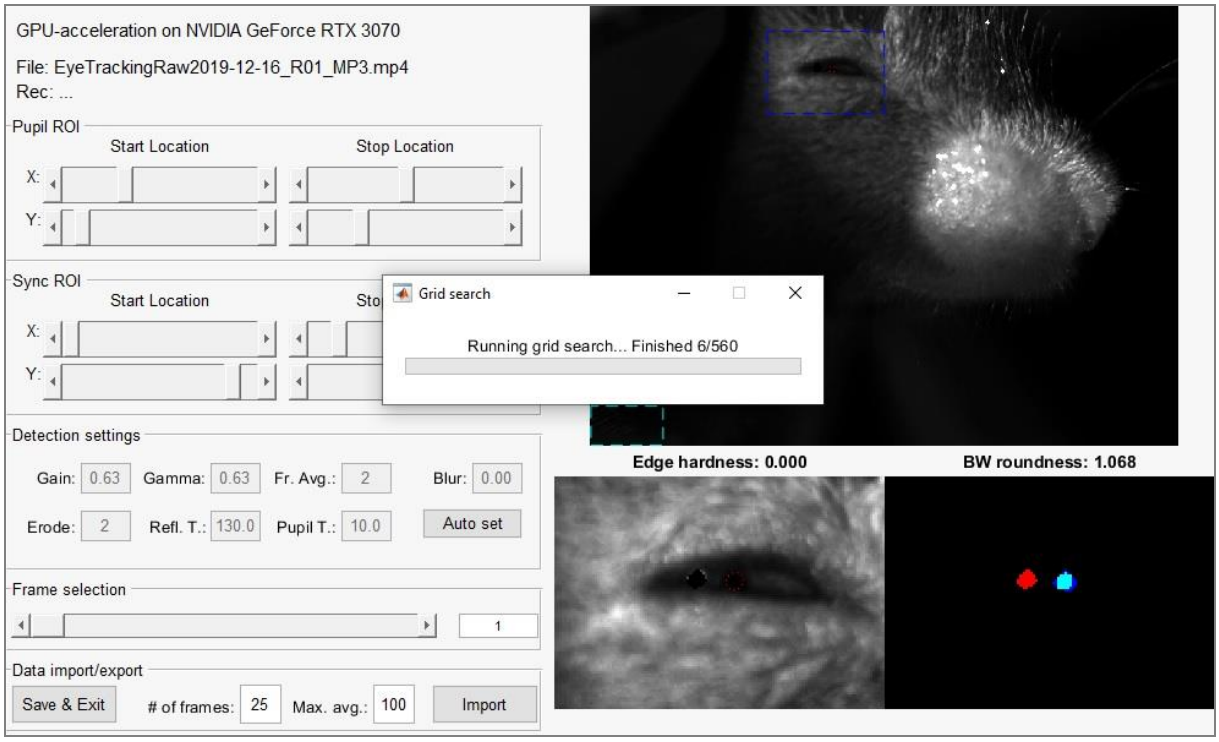
In the bottom of the screen, you can also adjust the number of example images (default: 25) and the maximum frame averaging (default: 100). As the program pre-loads all images from the video, increasing either the number of images or the maximum frame averaging will increase loading times.

Automatic parameters: labelling & training

After clicking on “label frames”, the program will show you a zoomed-in image of the pupil ROI, where you can select the center of the pupil (left click), then set the first axis of the pupil ellipse (second left click), and the second axis of the pupil ellipse (third left click). You can correct your previous left by right clicking in the image:

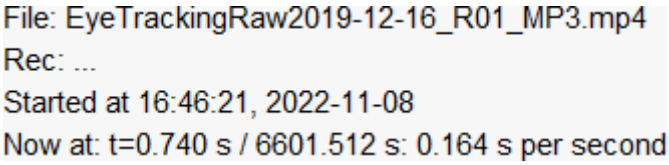


When you click either “auto set” or “Train&Track”, the program will run an initial grid search to optimize the parameters for the pupil detection, followed by a local optimization:



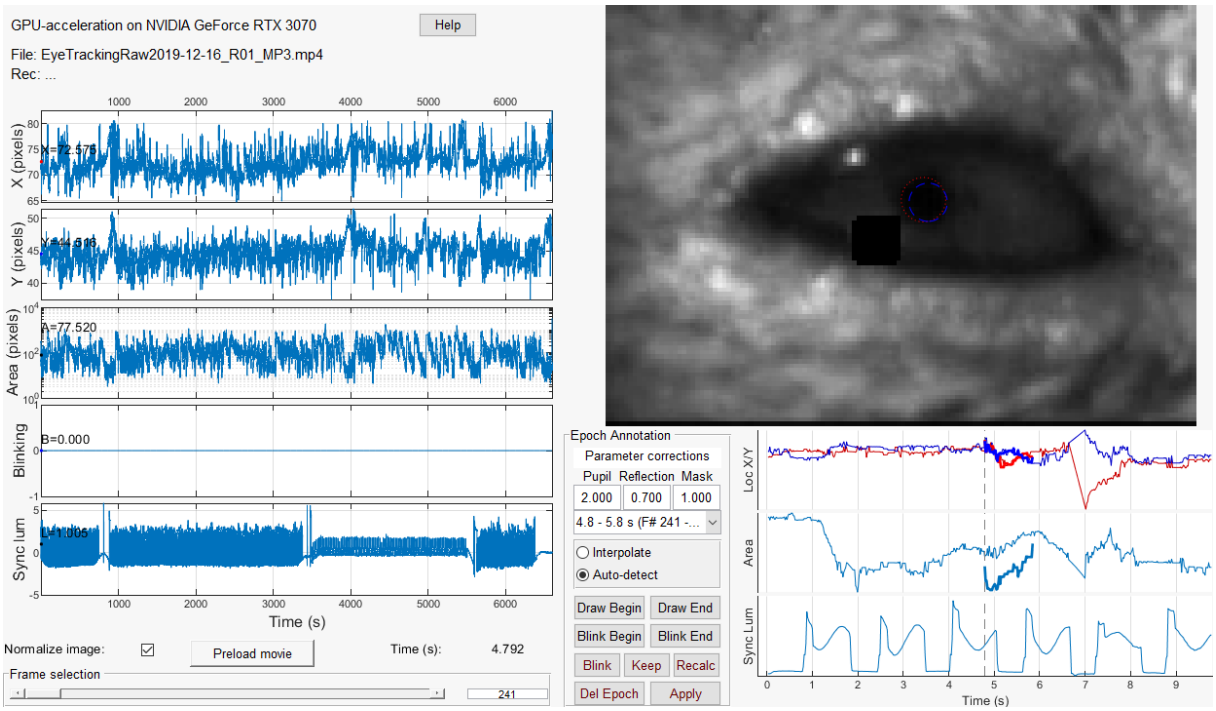
Tracking

While the program is perform pupil tracking, you can keep track of its progress in the following view:



Checking the results: curation and correction

After the pupil tracking has finished, you can check the results to curate the data and correct any errors:



On the left, you can see five time-series data plots that show the entire movie. The top two show the detected pupil location in X and Y pixels, and the third plots shows the pupil area in pixels. The fourth plot shows which periods you marked as “blinking”, i.e. where the mouse has closed its eye. The bottom shows the average luminance of all pixels within the synchronization ROI you selected. You can move through the recording by clicking on any of the plots, moving the frame selection slider at the bottom, or entering a frame number manually.

The right hand side of the GUI shows the data around the selected frame. The top right shows the pre-processed image with an overlay of the detected pupil. The bottom shows three zoomed-in plots around this frame, which display the X/Y location of the pupil (top, red/blue curves), the area (middle), and the synchronization luminance (bottom).

Interacting with epochs

The button panel allows you to interact with “epochs”. These are short periods during the movie that you can edit by marking them as a “Blinking Epoch”, recalculate the pupil detection, or interpolate the pupil size and position between the beginning and end frames of the epoch. When you load the

recording after the detection has finished, it will probably contain some epochs. These are periods where the algorithm was unsure about the pupil location, and the mouse was possibly blinking. You can select an epoch from the list and then interact with it as follows. You can:

1) Mark an epoch as a blinking episode by either clicking the button “Blink” or by pressing “b” on the keyboard.
2) Keep the epoch data (the traces shown in bold), remove any blinking and overwrite the old data by clicking “Keep” or pressing “n” (for non-blink) or “k” (for keep)
3) Recalculate the detection during this epoch by clicking “Recalc” or pressing “r”
4) Delete the epoch data and keep the old data by clicking “Del Epoch” or pressing “d”
5) Apply any marked blinking and pupil detections of this epoch and overwrite the old data by clicking “Apply” or pressing “a”

Note that the difference between “Keep” and “Apply” is that “Keep” will remove any periods marked as blinking, while “Apply” will also apply the blinking periods.

Parameter corrections

Sometimes the global parameters do not work well with a particular epoch. You can therefore change three values that affect the adaptive correction algorithm:

1) Pupil: this value is a multiplicative factor applied to the pupil luminance threshold. If the detected pupil is too small, you can increase this value. Likewise, if the pupil detection included areas that are outside the pupil, decreasing this value may help
2) Reflection: multiplicative factor for the reflection threshold. Lower values will exclude more of the image, higher values will exclude less of the image
3) Mask: any value lower than 1 will create a circular mask that excludes any area of the image outside the circle. This can be useful if the detection gets stuck at a dark corner of the image.

Creating new epochs

You can also create new epochs. Using the buttons “Draw Begin” and “Draw End” you can draw the pupil outline at the currently selected frames. Depending on whether you have selected “Interpolate” or “Auto-detect”, the program will now interpolate or run a pupil detection between these two points. An alternative way to add an epoch is simply right clicking on two time points in a zoomed-in plot to mark a beginning and end of an epoch. The program will now use the detected pupil at these two time points as if you had drawn them manually, and again either interpolate or auto-detect the pupil. Either way, this will create a new epoch that you can interact with as described above.

Finally, you can also mark periods where the mouse is blinking by using the “Blink Begin” and “Blink End” buttons, or by clicking with the middle mouse button at two time-points. Blinking epochs are shown by a black bar above the plots. You can similarly remove blinking periods by pressing the CTRL button while using the middle mouse button. To-be-removed blinking epochs are shown as grey bars.

Output file

If the video file is called “something_.mp4”, the output file will be called “something_Processed.mat”. It contains the sPupil structure with various fields.

```
                  sSyncData: [1×1 struct]
        vecPupilFullSyncLum: [1×370498 double]
       vecPupilFullSyncLumT: [1×370498 double]
                 strVidFile: 'PupVid_Rec1_2022-01-25R01.mp4'
                 strVidPath: 'E:\_TempData'
               sTrackParams: [1×1 struct]
             strMiniVidPath: 'E:\_TempData'
             strMiniVidFile: 'PupVid_RecT1_2022-01-25R01MiniVid.mj2'
                       name: 'PupVid_RecT1_2022-01-25R01Processed.mat'
                strProcFile: 'PupVid_RecT1_2022-01-25R01Processed.mat'
                strProcPath: 'P:\Exp2022-01-25T1\EyeTracking'
              vecPupilTime: [1×185249 double]
           vecPupilVidFrame: [1×185249 double]
            vecPupilSyncLum: [1×185249 double]
            vecPupilCenterX: [1×185249 double]
            vecPupilCenterY: [1×185249 double]
             vecPupilRadius: [1×185249 double]
            vecPupilRadius2: [1×185249 double]
              vecPupilAngle: [1×185249 double]
       vecPupilEdgeHardness: [1×185249 double]
       vecPupilMeanPupilLum: [1×185249 double]
          vecPupilAbsVidLum: [1×185249 double]
        vecPupilSdPupilLum: [1×185249 double]
    vecPupilApproxConfidence: [1×185249 double]
     vecPupilApproxRoundness: [1×185249 double]
        vecPupilApproxRadius: [1×185249 double]
      vecPupilNotReflection: [1×185249 double]
       vecPupilFixedCenterX: [1×185249 double]
       vecPupilFixedCenterY: [1×185249 double]
        vecPupilFixedRadius: [1×185249 double]
       vecPupilFixedRadius2: [1×185249 double]
         vecPupilFixedAngle: [1×185249 double]
        vecPupilFixedPoints: [1×185249 logical]
        vecPupilFixedBlinks: [1×185249 logical]
           vecPupilIsEdited: [1×185249 logical]
         vecPupilIsDetected: [1×185249 logical]
        vecPupilFiltSyncLum: [1×185249 double]
      vecPupilFiltAbsVidLum: [1×185249 double]
                    sEpochs: [1×0 struct]
```

The synchronization table

sPupil.sSyncData.matSyncData is a 4 x t matrix with t entries that provide simultaneous timestamps of the video time in seconds, the frame number of the video, the time of the national instruments clock in seconds, and the sample number of the national instruments clock. As you can also see in sPupil.sSyncData.MatrixEntries, the entries are: matSyncData([TimeVid FrameVid TimeNI FrameNI],t)

Raw synchronization luminance

`vecPupilFullSyncLum` and `vecPupilFullSyncLumT` provide, respectively, the average luminance per video frame in the synchronization ROI and the video time of that frame according to the camera.

Metadata
Several fields contains metadata:

- strVidFile: name of the original video file
- strVidPath: path of the video file
- sTrackParams: tracking parameters
- strMiniVidPath: path to the mini video file that contains only the ROI
- strMiniVidFile: name of the mini video file that contains only the ROI
- name: original name of the output file
- strProcFile: name of the processed file (should be the same as above)
- strProcPath: path of the processed file

Tracking data
The remaining fields are [1 x n] vectors, where n is the number of detected frames. Note that n here is not necessarily equal to the number of frames in the video: if you set frame averaging to a value of *x*, pupil detections only occur on the averaged image of *x* consecutive frames. As you can see in the example above, I set frame averaging to 2, so pupil detections ran at 50 Hz, while the camera frame rate was 100 Hz. This is also explains why the raw luminance vectors are twice as long. The following variables are all [1 x n] vectors, with each entry corresponding to a single pupil detection:

- vecPupilTime: video time in seconds
- vecPupilVidFrame: video frame number
- vecPupilSyncLum: averaged synchronization ROI luminance
- vecPupilCenterX: pupil location in X-pixels from the left *[before corrections]*
- vecPupilCenterY: pupil location in Y-pixels from the top *[before corrections]*
- vecPupilRadius: first ellipse radius (semi-major or semi-minor axis) *[before corrections]*
- vecPupilRadius2: second ellipse radius (semi-minor or semi-major axis) *[before corrections]*
- vecPupilAngle: angle of ellipse's focal points relative to vertical *[before corrections]*
- vecPupilEdgeHardness: steepness of brightness change around the pupil edge
- vecPupilMeanPupilLum: average luminance of all pixels inside the pupil
- vecPupilAbsVidLum: average luminance of all pixels inside the entire ROI
- vecPupilSdPupilLum: standard deviation of luminance of all pixels inside the ROI
- vecPupilApproxConfidence: confidence of first pupil approximation
- vecPupilApproxRoundness: roundness of first pupil approximation
- vecPupilApproxRadius: radius of first pupil approximation
- vecPupilNotReflection: number of pixels inside the pupil that are not excluded due to reflection
- vecPupilFixedCenterX: pupil location in X-pixels from the left *[after corrections]*
- vecPupilFixedCenterY: pupil location in Y-pixels from the top *[after corrections]*
- vecPupilFixedRadius: first ellipse radius (semi-major or semi-minor axis) *[after corrections]*
- vecPupilFixedRadius2: second ellipse radius (semi-minor or semi-major axis) *[after corrections]*
- vecPupilFixedAngle: angle of ellipse's focal points relative to vertical *[after corrections]*
- vecPupilFixedPoints: was this frame marked for manual curation (1) or not (0)
- vecPupilFixedBlinks: is the frame marked as a blinking episode (1) or not (0)
- vecPupilIsEdited: was this frame edited during curation & correction (1) or not (0)
- vecPupilIsDetected: was the pupil detected algorithmically (1) or interpolated (0)
- vecPupilFiltSyncLum: high-pass filtered synchronization luminance
- vecPupilFiltAbsVidLum: high-pass filtered video luminance

Note that the most important variables are therefore:

- Pupil detection variables: **vecPupilFixedCenterX, vecPupilFixedCenterY, vecPupilFixedRadius, vecPupilFixedRadius2,** and **vecPupilFixedAngle**
- Exclusion if the mouse is blinking: **vecPupilFixedBlinks == 1**
- Exclusion if the pupil detection failed: **vecPupilIsDetected == 0**

Troubleshooting

Question ("actually, it's more of a comment"): *It doesn't work*

Answer: Restart your PC

Q: *I downloaded everything, but it says files are missing*

A: Double check you have added all folders to the path in Matlab, you have the required Matlab toolboxes installed (Curve Fitting, Parallel Computing, Image Processing and Image Acquisition), and you're using a supported matlab version: R2019b is tested and works; anything earlier than R2016b will fail for sure; other versions might work. If it's still not working after you've tried the above, google the filename and reinstall its source repository. If it still fails, create a report here: https://github.com/JorritMontijn/EyeTracker/issues.

Q: *I cannot run any GPU code in matlab (i.e., gpuArray() fails)*

A: Make sure that you have the correct CUDA drivers installed for your GPU. Note that if you're using anything other than a (modern) Nvidia GPU, you cannot run CUDA.

Q: *I found a bug*

A: Great! Or at least, it's great that you found it, not that it's there. If you've fixed it, you can make a pull request, otherwise you can create a bug report here: https://github.com/JorritMontijn/EyeTracker/issues. Please copy/paste the matlab error message and as much detail as you can about what you were doing when it happened. If I cannot recreate the issue, I probably won't be able to fix it.