# Summary of gaze data analyzation process

## Detailed program flow

* get\_calibration\_point\_intervals.py
* get\_calibration\_error.py
  + filter\_gaps.py
  + detect\_outliers.py
* visualize\_error.py

## Calculating calibration error

During the eye-tracking experiment, a subject was shown calibration videos between the actual test videos. From the calibration videos, we need to derive how much error was introduced to the gaze points during the test. We assume that the error increases on both x- and y-axes as the test progresses.

## Process

### Gap filtering

There are gaps (missing measurements) in the data caused by blinks and other random occurrences (for example the pupil being momentarily obscured by eyelashes). Usually gaps are clustered close together and the data near the gaps is unreliable. In the first phase of processing, we detect and filter out gaps and the data in close proximity to the detected gaps.

Current method marks an area as a gap if there are five or more missing measurements in a short time period. After that, all data points inside the gap and in vicinity are pruned.

Current problems: The method for filtering gaps is not backed by any meaningful data. We need to explain why the different thresholds (gap amount, time between gaps) were chosen and we need to check if the chosen thresholds are meaningful.

### Automatic measuring of calibration point visibility intervals

There is some variance in the visibility time periods of calibration points between different calibration videos. We need to know exactly when a calibration point is visible, so we know which gaze measurement points are relevant. Gaze data outside of any calibration point being visible is ignored. Currently we measure if a calibration point is visible by checking the pixel darkness of the area it is supposed to appear in.

### Getting calibration errors

After we know which measurement points are relevant, we measure the difference between the measurement point and its corresponding calibration point. The errors on both axes are processed separately. Since there may be multiple measurement points for each given frame, we calculate the average of said points, thus compressing the data to one measurement per frame.

Current problems: The calibration point locations vary between videos introducing some error to our error calculations. We assumed that the point location was constant. We need to derive a margin of error caused by the inconsistencies of the calibration point locations.

### Detecting outliers

Even after filtering gaps, some erroneous points remain in the data, skewing the average error for a given calibration point. These outlying points need to be detected and removed from the data set.

Current method checks if each point has enough similar neighbors. If it has three or more, it is allowed to stay in the data set. This method considers only spatial neighborhood. Temporal outliers remain undetected.

Current problems: The current method is very bare bones and does not seem robust enough. The lack of temporal outlier detection is a significant issue. In addition, the current detection method is based on the combined sum of error of both x- and y-axes. The correctness of this method needs to be checked.

More problems: since the video dimensions are 16:9, the relative error of y-axis is far greater than on the x-axis even though the pixelwise error would be the same. This is not taken into account. Note: this may actually not even be a problem, but a good thing to keep in mind.

### Error summary line fit

After point cleanup, we calculate the average error of x- and y-axes for each calibration point separately. Using these averages, we can fit a line for the error on both axes. From this line, we can check how the error develops over the experiment. This is done for each subject separately. Based on the fitted line, we derive the correction coefficient for both axes. After each calibration video, the gaze points on the following test videos are corrected using the derived coefficients.

Current problems: Still in this stage there are outlying points which skew the fitted lines resulting in erroneous correction coefficients. We need to either eliminate the outliers in this phase or we need to correct the gaze points before this phase.

Suggestions:

Calculating the average: We could weigh the average for each error point by weighting the data by Gaussian distribution. Thus, we would minimize the weight of erroneous points at the start and the end of the data group. Other method would be to prune the first ten and last ten points which seem to be usually erroneous. Note: the correctness of these methods need to be verified.

## Light method for processing

### Deriving the error threshold

Derive the error threshold for the eye-tracking device. According to the manufacturer web page, the gaze accuracy error is 0.60 degrees. The distance from the screen was approximately one meter. From this we can approximate the error threshold for gaze points. Other source for error is the camera shaking, which causes some error to the gaze projection surface which is calculated by the Pupil software.

### Point correction

From the gaze points, we calculate a similar surface as the one calculated by the Pupil software. This