Radboud University

Faculty of Science

Validation of measuring 3D eye movements by video-oculography.

Thesis BSc Science

June 2021

Author:
Eva Severijnen

Supervisor:
dr. Annemiek Barsingerhorn &
dr. John van Opstal

Second reader:
dr. Marc van Wanrooij

Contents

1 Introduction

For the study of the human motor system, looking at the oculomotor system is a relatively simple option. This is the result of the fact that only six extraocular muscles are contributing to eye movements making this system convenient for the study of the underlying neural mechanism of motor control. However, relatively simple is still complicated in this case. Since movements and control both go really fast and the mechanisms are still complicated even with only six muscles. But with the study on eye movements a lot of knowledge about the control of movements by different parts of the brain was acquired in the past [Purves and Augustine, 2010]. Looking at the eyes and their movements can thus give a lot of information about the brain eventhough it is a complicated task.

Significance in clinical and research fields

Measurements of eye movements are used in both the clinical as well as the research neuroscience field. In the clinical field measurements of the eyes can contribute to early diagnoses of brain diseases like Multiple Scleroses and be used by the monitoring of treatment effects and disease progression in these neurological diseases [Leigh, 2015]. Within research, eye movements are used in studies on for example memory, decision making and perception. In these studies accurate eye tracking is important since this is a major part of their experimental paradigms and the way in which researchers try to reveal underlying mechanisms [Otero-Millan et al., 2015]. It is thus crucial for both fields to be able to measure eye-movements adequately.

Eye movements

The movements of the eyes can be described by only three degrees of freedom. This is due to the fact that the eyes' six muscles are arranged orthogonally in three pairs of antagonist as shown in Figure 1. The horizontal component is generated by the lateral rectus (abduction) and the medial rectus (adduction). Vertical movements are generated by the superior rectus (elevation) and the inferior rectus (depression), but also partly by the superior and inferior oblique muscles. These later muscles are primarily inducing torsional movements: intorsion and extorsion. So, eventually the three rotational axes of the eye can be described as x, y and z as shown in Figure 2. Most muscles are innervated by the axons of lower motor neurons that form the oculomotor cranial nerve, except from the lateral rectus and the superior oblique which are innervated by the abducense and trochlear cranial nerves respectively [Purves and Augustine, 2010].

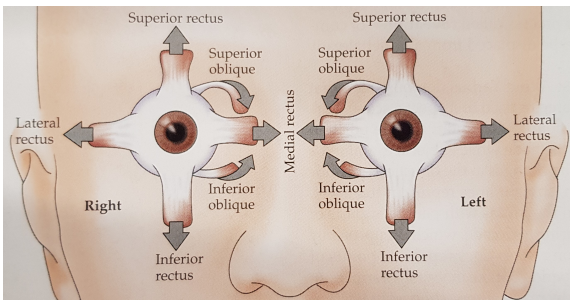

Figure 1: The six muscles of the eye. [Purves and Augustine, 2010].

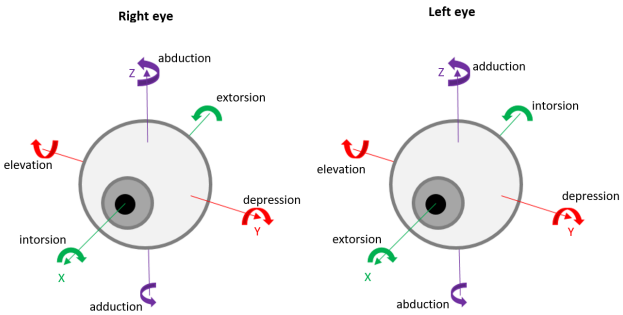

Figure 2: Rotational axes of the eyes and their accompanied movements.

Eye movements are necessary for creating a stable visual perception. Only a small part of the human retina generates high visual acuity, namely the fovea. The fovea is a small circular region of about 1.5 mm in diameter close to the center of the retina. To be able to direct the fovea to a target of interest (foveation) or to keep a stable image of a target while it or the body is moving, the human eye makes five types of movements; saccades, smooth pursuit movements, vergence movements, vestibulo-ocular movements and optokinetic eye movements. Saccades are fast ballistic movements that can be both small as large in amplitude and are made voluntarily as

well as involuntarily. Smooth pursuit movements make it possible to track a moving object and is a voluntary movement. However making this movement without a moving object is for most people impossible. Vergence movements converge or diverge the eyes to be able to see targets close by but also far away. Lastly, the vestibulo-ocular movements and the optokinetic eye movements which are together responsible for keeping a stable gaze by compensating for head movements and movements of the environment. These last occur with slow movement of big parts of the visual field and build up slowly, while the first are generated by the vestibulo-ocular reflex (VOR) following small brief changes in head position. [Purves and Augustine, 2010]. An example of this kind of movement is the ocular counter-roll that arises by tilting of the head.

Mathematically defining eye movements

Mathematically eye movements can be described in multiple ways. The axes of rotation can for example be described as head fixed or eye fixed. With the eye fixed approach the axes are 'fixed' to the eye, meaning when the eye turns, the axes change position as well and with head fixed axes the axes stay at their position when the eye is turning in the eye socket. Additionally, the rotation over the different axes are non-commutative. This means that the order of rotation around the axes is important for the final orientation of the eye. So, a rotation around the vertical axis and then around the horizontal axis, does not lead to the same orientation as first the rotation around the horizontal axis and then the vertical axis. The main differences in the mathematical description of the movements are therefore the order of rotation around the axes and their references frame. Euler's theorem states that a single-axis rotation can describe a rigid body displacement if one point of the body is fixed in space, with a rotation axis through this fixed point. Scientists Fick and von Helmholtz did both choose a different order of rotations to describe eye movements in the eye fixed system, but with three axes there is a total of twelve possibilities [van Opstal, 2004]. Figure 3 visualizes the systems of Fick [Fick, 1854] and von Helmholtz [von Helmholtz, 1866].

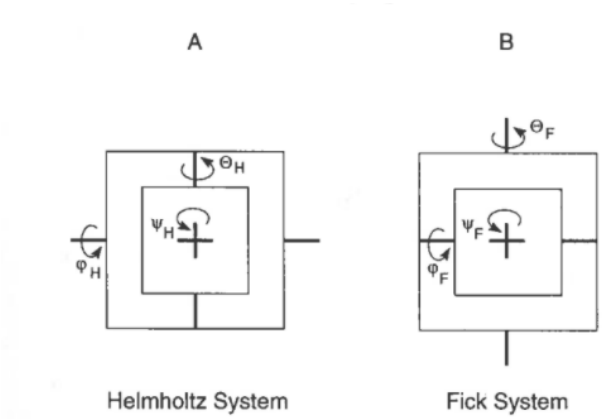

Figure 3: The Fick (A) and Helmholtz (B) gimbals, showing the different orders of axes rotation [van Opstal, 2004].

In this thesis the eye movements are described by rotation vectors and quaternions. Quaternions have a useful similarity with fixed-axes rotations namely non-commutativity and are therefore useful when describing eye movements. A quaternion is a four-component complex quantity, $q = q_0 + \mathbf{I} \bullet \mathbf{q}$, with a vector part($\mathbf{q} = (q_1, q_2, q_3)$) and a scalar part($q_0$). Which can also be written as, $q = cos(\theta) + sin(\theta)(q_1 i + q_2 j + q_3 k)$ (with the constraint that the length of the vector is taken to be equal to one) and a quaternion has the following characteristics:

$$i^2 = j^2 = k^2 = ijk = -1$$

Quaternion multiplication is then defined as follows with $pq \neq qp$ for p and q not parallel:

$$pq = p_0 q_0 - \mathbf{p} \bullet \mathbf{q} + (p_0 \mathbf{q} + q_0 \mathbf{p} + \mathbf{p} \bullet \mathbf{q}) \times \mathbf{I}$$

This multiplication is computationally a lot easier than rotation matrix multiplication and therefore useful when describing rotations while programming. When describing rotations of a

vector ($\mathbf{u}$) this can now be described as follows:

$$\mathbf{u}' = q\mathbf{u}q^{-1}$$

A different approach of describing 3D-movements is with rotation vectors, this is actually a quaternion divided by its scalar part [van Opstal, 2004]: $\mathbf{r} = \mathbf{q}/q_0$. This means they are normalized and therefore a useful tool in graphically visualizing the eyes orientation.

Donders' law and Listing's law

Donders' law says that eye torsion is always defined by the horizontal and vertical components of the gaze direction. So for each orientation of the eye, no matter what the path to this orientation was, the torsion is identical. Torsion can thus be described as a function only depending on the horizontal and vertical position of the eye at that moment. For example, if the eye rotates to the right and then up or first up and then to the right, as long as you end up in the same position the torsion of the eye will always be equal in that position. However, this is not what is expected mathematically, since accumulation of torsion should occur. Listing's law is a specification of this law and states that there is no torsion in a head fixed with no convergence and accommodation. When the orientation of the eye is then described as its rotation vector, with the primary position (r = (0,0,0)), as the origin of the reference frame, all of the rotation vectors will form a plane. This plane is called Listing's plane. For example (0,0.2,0) is a orientation of 20 degrees in y direction because, $r_j = \tan(20/2)$=0.2. The division by two originates from the fact that a vector is multiplied with both q and q^{-1}. When Listing's law is obeyed, the vectors form a plane in the y-z plane, so x is zero, meaning no torsion is occurring. A visualization of listing's plane is shown in Figure 4.

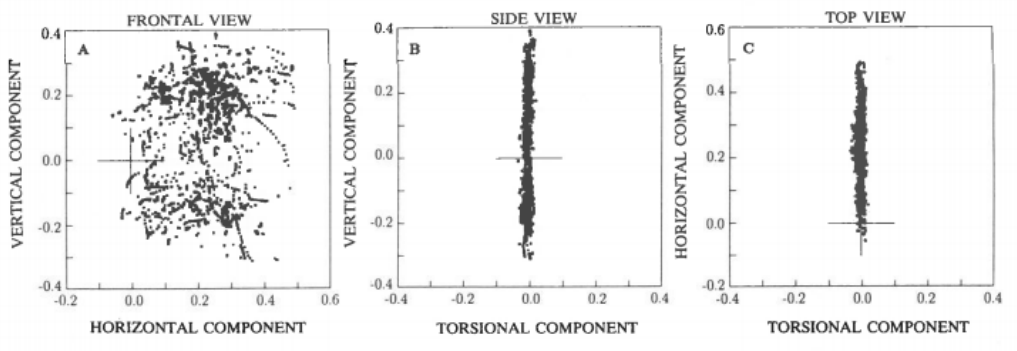

Figure 4: Listing's plane visualization of data of a monkey making saccadic eye movements [van Opstal, 2004].

There is still an ongoing discussion about where this phenomenon is generated. The question raises if this phenomenon it purely mechanical. Van Opstal provided evidence that it is for sure after or within the deep layers of the Superior Colliculus in the brain [van Opstal et al., 1991]. But, they also stated that it cannot be purely dependent on the mechanical properties of the eye's system [Van Opstal et al., 1996].

Current methods for measuring eye movements

Currently there are three main approaches for measuring eye movements. The scleral search coil technique uses a annulus shaped contact lens containing an inductive coil and external magnetic fields to measure the orientation of the eye [Collewijn, 1985], which gives a good spatial and temporal resolution. However, slippages of the lens leads to inconsistencies and because of discomfort due to the lens this method is not fit for clinical practice [Otero-Millan et al., 2015] [Ong and Haslwanter, 2010]. A second method used is electro-oculograpy (EOG) and is based on a voltage difference between the cornea and the retina [Brown and Marmor, 2006]. However, only horizontal and vertical movements can be measured in this way and the spatial resolution is low and noisy. [Ong and Haslwanter, 2010].

Finally there is video-oculography, which has great potential to become the new golden standard for measuring 3D eye movements [Ong and Haslwanter, 2010]. With this technique, video recordings are used to measure the movements of the eye. Horizontal and vertical movements are quite easy to subtract from the recordings due to pupil tracking, but for torsional movements a different approach is needed since torsional movements cannot be subtracted from the pupil directly. More information is thus needed and the iris potentially gives this information. Different approaches were executed before to get the information from the iris. For example, tracking of artificial markers that are tattooed onto the iris, stable iris structures or the whole iris pattern at once. However, artificial markers are invasive and suffer from occlusion fast. Looking at the whole iris favors stable iris structures since the noise is less because there is more information to work with even when the eyelids cover a part of the iris. To eventually measure the amount of torsion cross-correlation, manual tracking or template matching have been used. Template matching is the least labor intensive and most reliable since it suffers from occlusion less than the other approaches. The combination of whole iris tracking and template matching thus seems most favorable. However, before using template matching the iris pattern must be unrolled into a rectangular shape and rectification is needed for eccentric eye positions [Jin et al., 2020][Ong and Haslwanter, 2010]. The amount of torsion can then be determined in a cumulative way, so with respect to the previous video frame or in respect to a template in straight ahead eye position.

Although video-oculography has great potential and many approaches have been developed, there is still not a new accepted standard. This is probably caused by the limitations that still exist, like occlusion by the eyelids, complex algorithms that require manual intervention, low resolution and lack of real-time measurements. Otero-Millan et al. (2015) proposed a technique that overcomes part of these limitations due to the combination of openCV template matching software [OpenCV,] and concepts inspired by the iris recognition field.

Method investigated in this thesis

To see if the method of Otero-Millan et al.(2015) can be used in the research and clinical field A. Barsingerhorn wrote a algortihm to measure torsional eye movements based on the Otero-Milan et al. (2015) article and the opensCV template matching software in combination with a pupil labs eye tracker. It uses the whole iris to perform the template matching and determines the amount of torsion in comparison with a template in straight-ahead position of the eye. This algorithm was then further improved by D. van Veen to overcome some more of the limitations [Van Veen, 2019]. The goal of the algorithm (Appendix A) is eventually to be able to track torsional eye movements adequately and reliably and thus be able to measure 3D eye movements.

The goal of this thesis will be to validate the robustness of the algorithm mentioned above and see if the method is fit to be used in other research. This will be done by doing two different experiments and demonstrating if this method and algorithm generates reliable results. In one of the experiments torsion is measured during vestibular stimulation, in the other the full 3D eye movements are measured during a saccade task.

2 Methods

Experimental set-up

For the experiments the vestibular chair set-up of the Biophysics department at the Radboud university Nijmegen was used. The initial position of the chair is shown in Figure 5 and the chair can rotate around two different axes and a third one can be adjusted before the start of an experiment. The black frame is attached to the ground and thus rotates around a vertical axis. The blue frame is attached to the black one and rotates around a horizontal axis. The red frame can be adjusted by hand to a preferred orientation, in the experiments the frame was set perpendicular to the rotation axis of the blue frame. Since the chair is positioned approximately in the middle of the frames and the height of the chair is adjusted to make sure the head of the participant is in the middle of the set-up translation is minimal. In the chair there are the possibilities to track head movements with a head-tracker and to display light stimuli on the LEDs frame opposite to the participant chair, Figure 6 shows the approximate arrangements of the LEDs.

During the experiments the eye of the test subject was recorded with the pupil labs glasses which are shown in Figure 7. The glasses has an eye camera (or two) to record the eye(s) and a

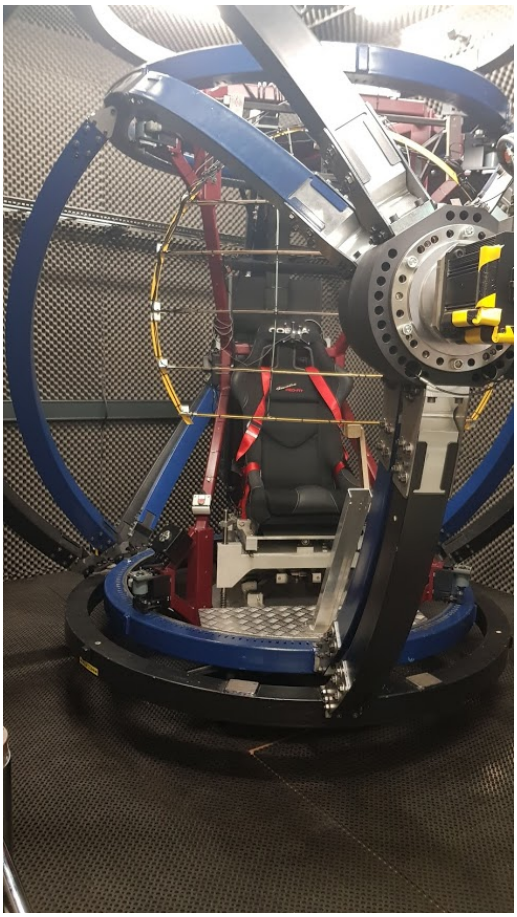

Figure 5: The vestibular chair set-up at the Radboud University

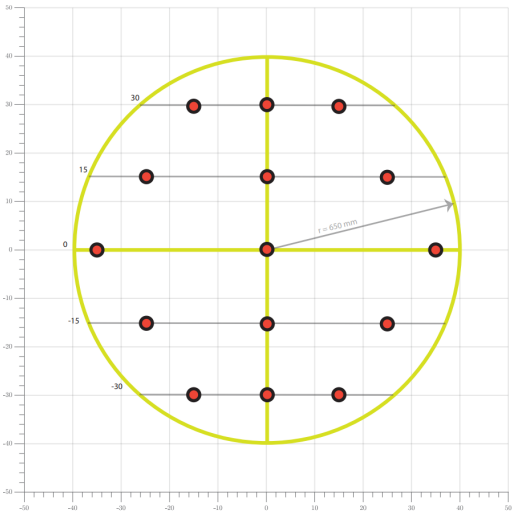

Figure 6: Approximate spacing of the LEDs in the LED frame in the vestibular chair.

world camera that is pointed in the opposite direction. This last camera makes it able to correct for the position of the glasses on the head and visualize were the participant is looking at.

Figure 7: The binocular Pupil labs glasses used, only the recordings of the right eye were analysed in this thesis

Experiments

In this set-up two different experiments were executed with four participants. Out of which one experiment with a moving chair and one with a non-moving, static, chair. The details of the experiments are described below. For both experiments it was important that the participants looked straight-ahead from the moment the session was started. This was done to make sure that in the analysis the template was created in a straight ahead position of the eye. During the experiments all the lights in the rooms were on and the chair was positioned towards the doors to the experimenter room (those were open) to decrease the pupil size and thereby facilitating the analysis. At the start of each session with a participant a manual calibration was executed with the pupil labs software and an external calibration circle that comes with the software. This circle

was shown on a mobile phone that the participant held in front of him or her self and then they were told to turn their head around while keep looking at the middle of this calibration circle with their eyes. After this calibration the following two experiments were executed:

Experiment 1: Ocular counter-roll

The goal of this first experiment is to show that the torsion that is occurring due to the torsional vestibulo-ocular reflex can be measured. The experiment consisted out of different sinusoidal stimuli around the horizontal axis of the chair, with the rotation axis orientated in the gaze direction of the participant due to the placement of the red frame. This was done to to evoke an ocular counter-roll due to the vestibulo-ocular reflex. Two frequencies were measured, 0.1 and 0.2 Hz For 0.1 Hz amplitudes of 5, 15, 30 and 45 degrees were measured and for 0.2 Hz amplitudes of 5, 15 en 30 degrees. This made a total of seven blocks where the participant needed to look at the centre LED during the whole experiment.

Experiment 2: Saccadic movements

The goal of this experiment is to show that Listing's plane can be measured. If this is possible then the technique can be reliably used, for example during locating measurements for light or auditory stimuli. The experiment was executed with a static chair and the participant needed to make a saccadic movement from the central LED to an eccentric LED, which was on for 2000 ms and then return to the central LED which was then shown for 1000 ms. The eccentric LEDs went on in random order and each LED was presented three times during the experiment, creating a total of 42 trials. This was executed twice, the first time the participant was told to keep his head still. The second time the participant could move his head freely.

For the second experiment an additional pilot experiment was performed as well. This was executed behind a computerscreen using the pupil labs software and a normal 5-point calibration. The participant was in bright daylight and made random saccades over the computerscreen.

Data analysis

Template creation

The data analysis was performed using the Matlab script (Appendix **??**) developed by A. Barsinger-horn and D. van Veen. The first step of the analysis was to create a template of the iris of the participant while in a straight ahead orientation (Figure 8, Appendix A.1). This template was made by first transforming the pupil from an ellipse into a circle and then this circle was unfolded into a rectangle. The iris was then extracted out of this rectangle and this image was converted into a black and white image. This was done with a video frame in the beginning of each block where the participant was not blinking. For this the iris band size was set manually to capture as much iris as possible without much eyelid. This depended mostly on the position of the camera and the size of the pupil and iris of the participant during the experiments. To make sure that reflections and eyelids did not influence the results too much, a threshold was set to detect eyelids and reflections and these spots in the black an white image were filled with random noise. This threshold was also variable between subjects and depended on the amount of structure in the iris, the darkness of the iris and the amount of light on the eye lids. The video frame number, the iris band size and this threshold were optimized manually after visual inspection of the images and for each test subject the final parameter settings are listed in Table 1.

Table 1: The parameter settings for each participant

Test Subject	**Iris band size**	**Imbinarization threshold**	**Framenumber**
1	45	98	100
2	35	90	150
3	30	95	150
4	30	95	150 (300 for recordings 032 and 041)

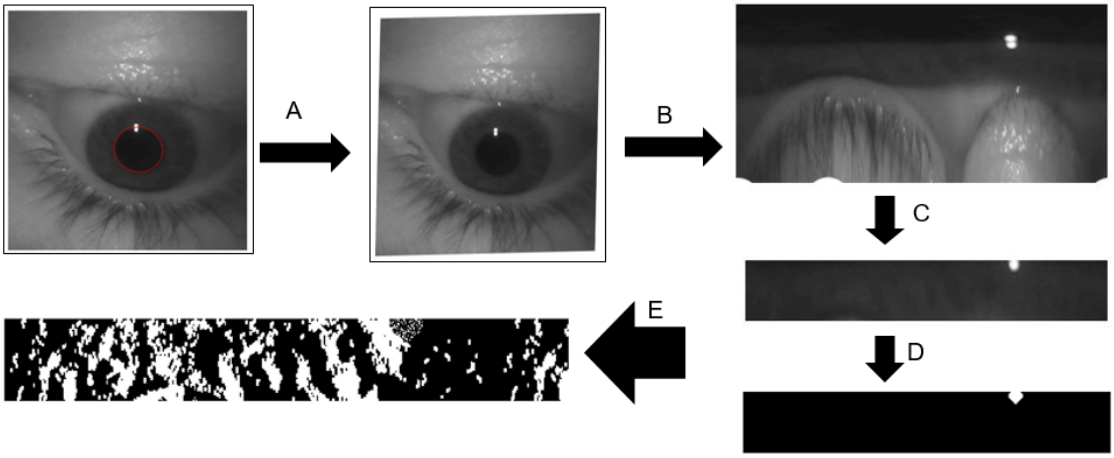

Figure 8: Creating the iris template. A. Transformation from ellipse to circular pupil. B. Unfolding image into rectangle. C. extraction of the iris band D. Detecting of eyelids and reflections. E. Turing image into black and white and adding noise at detected area's

Torsion measurement

After the template was formed the amount of torsion in each video frame was determined by performing the same steps as to create the template and then looking at the shift of the iris pattern in each video frame in comparison to the iris template by template matching (Appendix A.1). The output of this template matching is the amount of torsion of the eye for each video frame in the recordings with respect to eye camera. To smoothen these raw results a median filter with a width of 20 samples was added. To generate more reliable data, the data with a pupil labs confidence of lower than 0.6, meaning in that case the pupil labs software was unable to reliably detect the pupil or reliably fit an ellipse through the pupil boundary data, was removed. The same was done for a cross correlation factor of the template matching output of less than 0.2 (Appendix A.2).

To be able to compare the filtered torsional data with the actual movement of the vestibular chair in the first experiment we synchronized all data to the same timescale (Appendix A.3). The gain of the torsional movement was then determined by fitting a sine wave through the filtered data (Appendix A4). To facilitate the fitting, two minima were selected manually. One at the beginning of the signal and one in the end to specify the range of the fit. With this sine wave the gain of the torsional VOR was determined by dividing the amplitude of the fitted sine wave by the amplitude of the vestibular chair.

Defining the torsion in world reference frame

For the second experiment it was important to define the measured torsion in a world reference frame since only then we can show Listing's plane. Multiple steps were executed for this. First the torsion measurements were translated to quaternions that indicates the 3D orientation of the eye in world cam reference frame with the code in figure 9.

```
% Create quaternions for straigh ahead position

% normal of the pupil ellipse fit in eye-cam frame
qEye  = quaternion.rotateutov([0 0 1]',-circlenormal(:,1+ind:end),1,1);

% normal of the pupil ellipse fit in world-cam frame
qEye0  = quaternion.rotateutov([0 0 1]',gaze_normalstotal(:,indexx+ind),1,1);

Eye0Mat=RotationMatrix(qEye0);

% quaternion of torsion estimated based on camera images
qTorsion=quaternion.angleaxis(deg2rad(-torsionhighconf),[0 0 1]);

[qTorsion,n] = normalize(qTorsion); % normalize quaternion

% quaternion of orientation of the eye in world-cam frame
qGaze  = quaternion.rotateutov([0 0 1]',gaze normalstotal(:,1+ind:end),1,1);

lengths=min([length(qTorsion) length(qEye)]);
for j=1:lengths
    qTot(j)=qEye(j)*qTorsion(j);
    qTotGaze(j)=qGaze(j)*qTorsion(j); % combine into one quaternion representing 3D eye
orientation in world-cam reference frame

end
```

Figure 9: Translation from eye camera to world camera reference frame

The next step was the conversion to the world reference frame since due to slightly tilted glasses or the anatomy of the participant this can be different than the world camera reference frame. This last correction was done by manually selecting five markers on the world camera video frame from which we know their positions in comparison to each other and the participant, and determining the position and orientation of the markers in world camera reference frame . With this data a homogeneous matrix was formed (Appendix A.4) and with this the quaternions and thus orientation of the eye was thereby transformed into world reference frame by the steps in Figure 10 .

```
% transform unit vectors of gaze into world ref frame (only orientation).
gaze_normalstotalCorrected=(Transforms(1:3,1:3)*gaze_normalstotal);

% transform in quaternions
qGaze  = quaternion.rotateutov([0 0 1]',gaze_normalstotalCorrected(:,1+ind:end),1,1);

lengths=min([length(qTorsion) length(qEye)]);
for j=1:lengths
    qTotGaze(j)=qGaze(j)*qTorsion(j); % new 3D quaternions in world ref frame

    rvGazeNew(j,:)=vector(qTotGaze(j))/real(qTotGaze(j));%Corresponding rotation
vectors

    matqTotGaze=RotationMatrix(qTotGaze(j)); % corresponding rotation matrices
    normqTotGaze(j,:)=matqTotGaze(1:3,3); % Unit vector of gaze in world ref frame

end
```

Figure 10: Translation from world camera reference frame to world camera reference frame

Finally, a plane fit was performed through the data points to be able to correct for the tilt in Lisings plane originating from the difference in the real primary position of the ocular motor system and the straight ahead position of the LED that was chosen. With the normal vector of this plane the data was corrected to a straight plane around zero to facilitate the interpretation of the results.

3 Results

In this section the results are shown for both experiments that were executed. The first experiment had the goal to show that the torsional vestibular-ocular reflex response could be measured and the second experiment had the goal to display Listing's plane with head still saccadic eye movements.

Experiment 1: Ocular counter-roll

In general, it was possible to measure the torsion of the ocular counter-roll with this method. For most of the vestibular stimuli the output of the analyses showed a sinosoid as was expected. However, the amount of noise and the accuracy of the tracking of the torsional movements by the method differs a lot between test subjects. For the 45 degrees, 0.1 Hz vestibular stimuli the raw data is shown in Figure 11, for the rest of the stimuli this data can be found in Appendix B. Test subject one shows stable tracking and a high signal to noise ratio, whereas for subjects two and three the noise is more abundant but the general movement pattern, the sinosoid, is still clearly visible. For test subject four the case is different, the signal that was measured does not look like a sinosoid and the cross correlation factor is really low for a lot of data. Because of this the data of test subject four was left out of consideration for the rest of the result section.

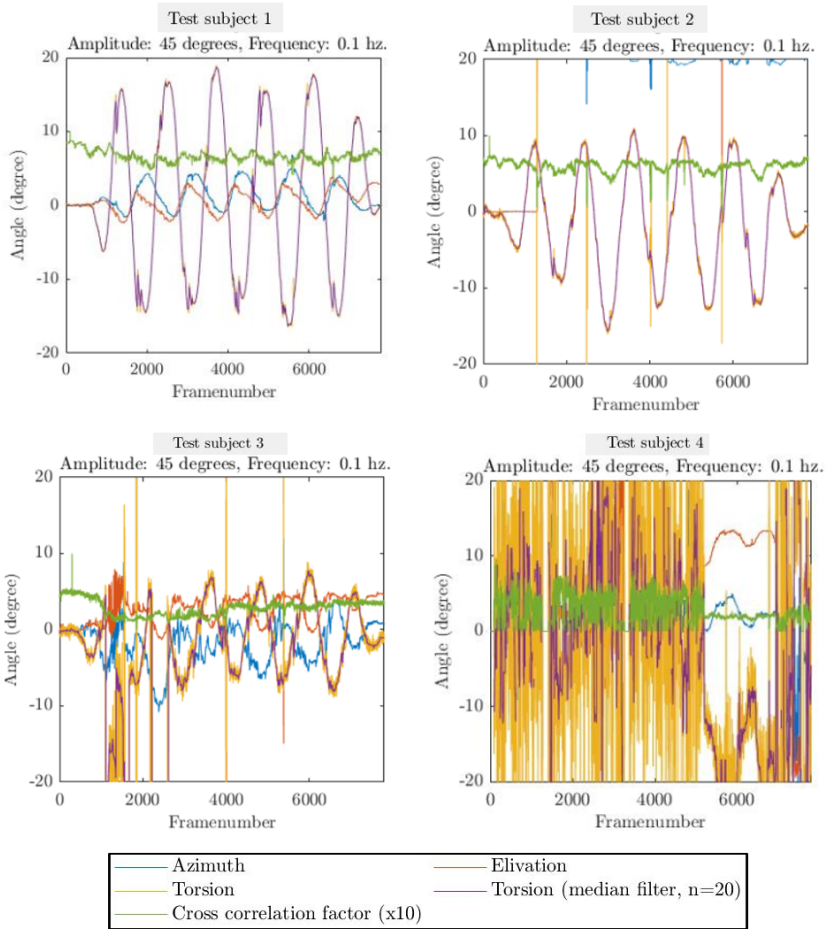

Figure 11: Raw data of each test subject for the 45 degrees and 0.1 Hz sinusoidal vestibular stimuli.

For the first three test subjects the filtered and fitted data is shown in Figure 12 for the stimuli of 45 and 15 degrees at 0.1 Hz and 15 and 5 degrees for 0.2 Hz. As can be seen in the figure, the torsional VOR is well measured for all three participants overall. However, in C we lost the first 30 seconds of data due to the fact that the participant closed his eye partly during this time span causing bad template matching. For Figures K and L the synchronization has failed causing the signals to be in-phase except from anti-phase. In all other cases the synchronization does perform well as can be seen in Appendix B.

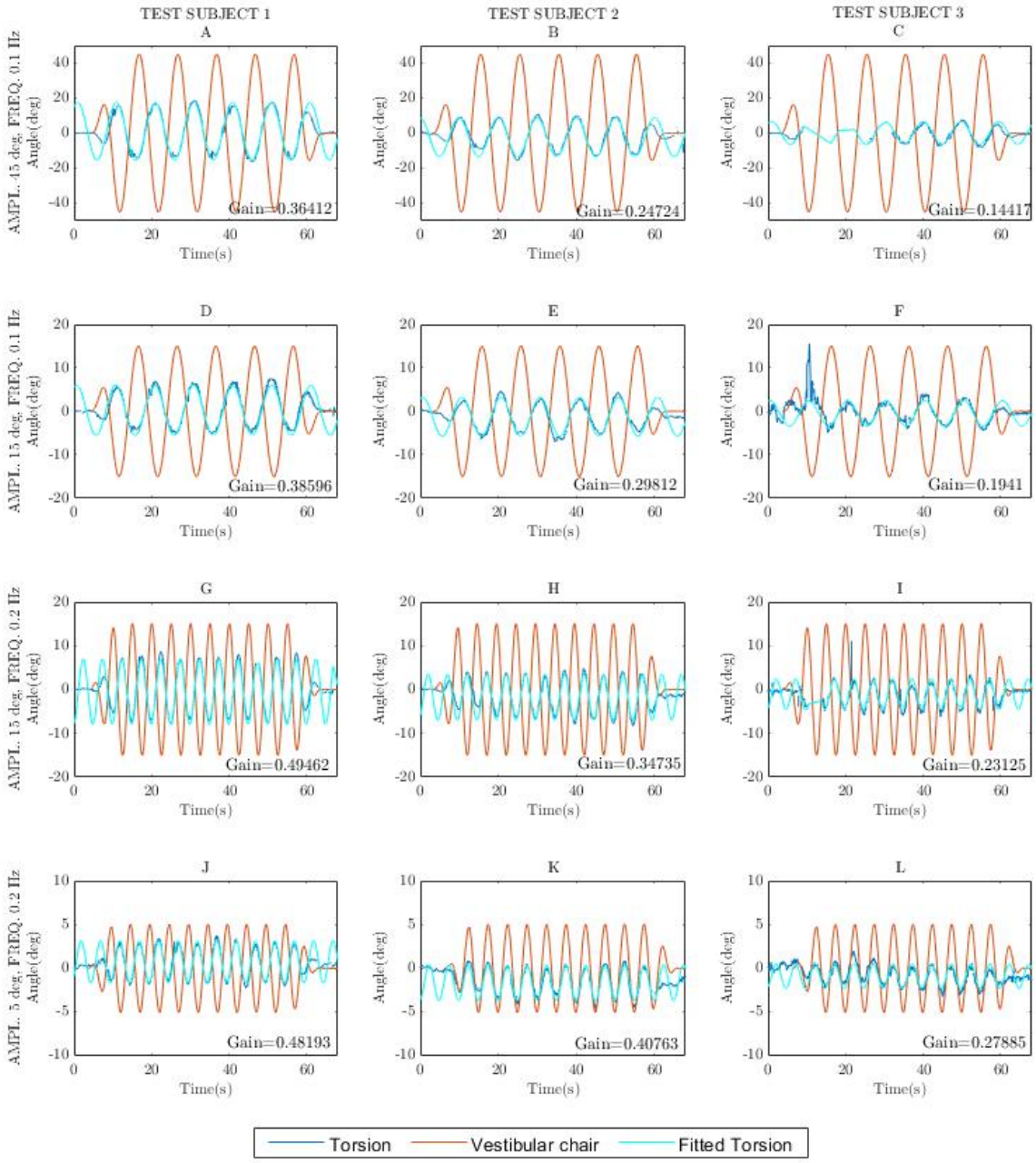

Figure 12: Four different vestibular stimuli and the measured torsional VOR for the three test subjects.

The gains of all three test subjects for each vestibular stimuli are displayed in Figure 13. Already for these three test subjects there is a general trend visible; the gain becomes smaller with greater amplitudes and bigger with higher frequencies. The 5 degree and 0.1 Hz of the second test subject appears to be an outlier in this trend.

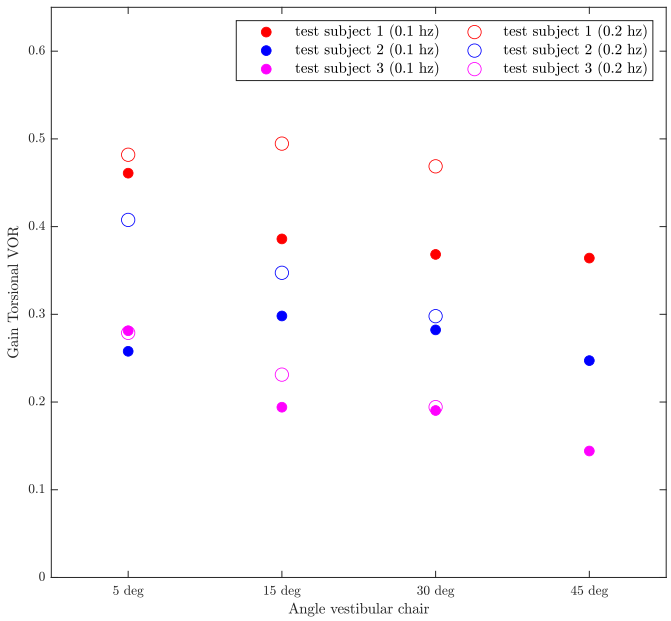

Figure 13: Gain of the torsional VOR for each test subject and each vestibular stimuli.

Experiment 2: Saccadic movements

In this experiment the saccadic eye movement for each participant was measured in head free and head still situations. The rotation vectors of each video frame were detriment and plotted to be able to show listing's plane. In the pilot that was executed in optimal circumstances for the eye tracker it was able to visualize listing's plane with head fixed saccadic eye movemetns as can be seen in Figure 14.

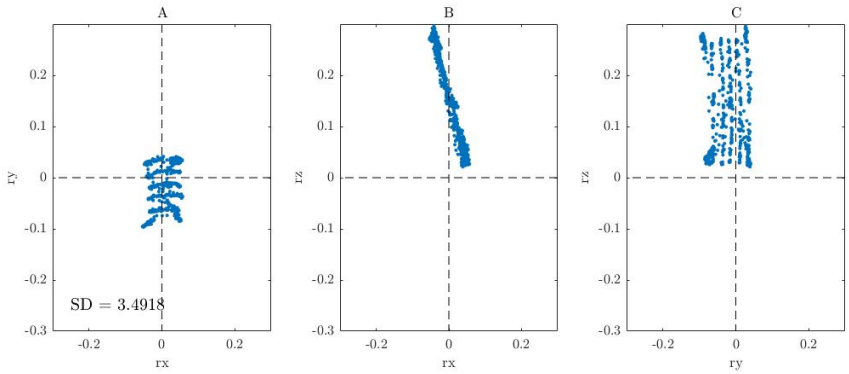

Figure 14: rotation vector plot for the pilot with optimal circumstances for the pupil labs eye tracker.

For the vestibular chair set up the rotation vectors for each video frame are plotted for both the head free (red) as head still (blue) situation in Figure 15. Due to listing's law it is expected that the vectors of the head still situation would form a plane in the z-x plane (mid row in the figure) due to the rx component being close to zero, however this is not immediately clear from this plot.

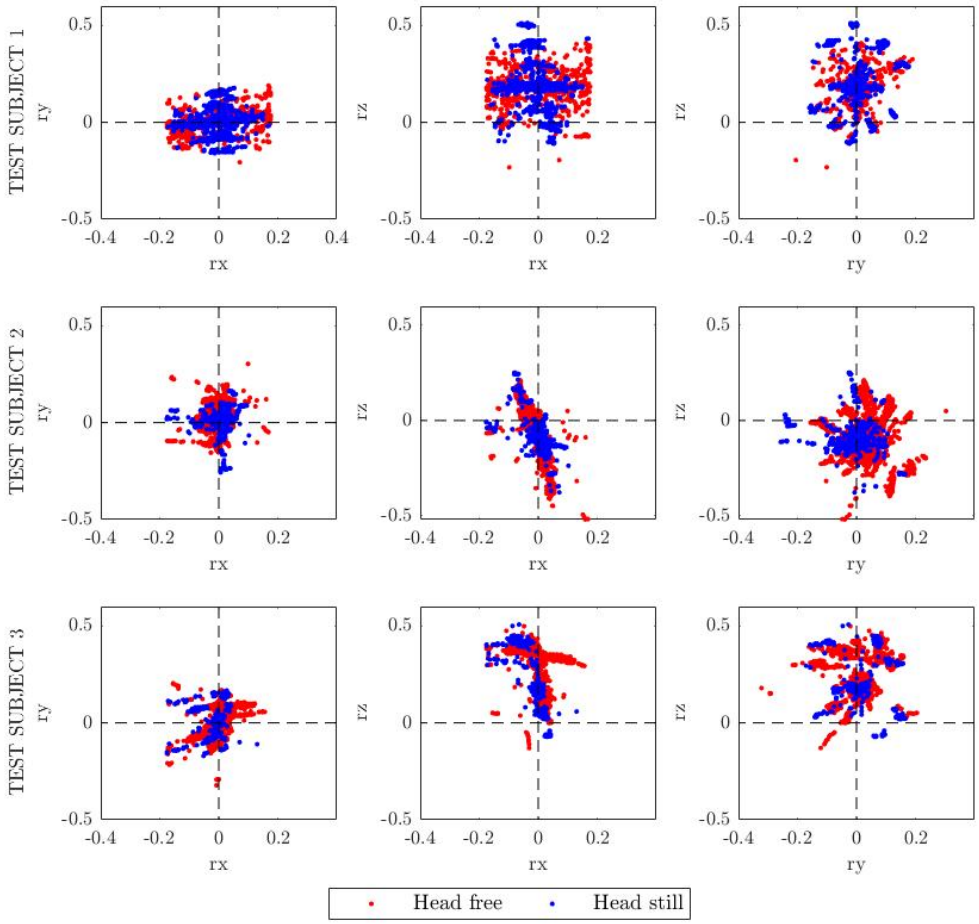

Figure 15: Rotation vector plot for the head free (red) and the head still(blue) situation.

To see however if there is a plane, but for example the plane is twisted or tilted, the plane fit was performed. The data with respect to the normal vector of this plane (2) is shown in Figure 16. After this correction listing's plane becomes more clear especially for test subjects two and three. For test subject one the plane is less clear. For this test subject the set-up of the LEDs is however most clearly visible in the x-y plane, suggesting we have data from all the different eye orientations. For test subjects two and three a lot of the data of the eccentric eye positions have been filtered out due to bad confidence or cross correlation.

Table 2: The normal vector values of each fitted plane

Test Subject	**C(1)**	**C(2)**	**C(3)**
1	0.0183	-0.1617	0.0336
2	-0.0161	-0.2488	-0.0128
3	-0.0235	-0.2057	0.0322

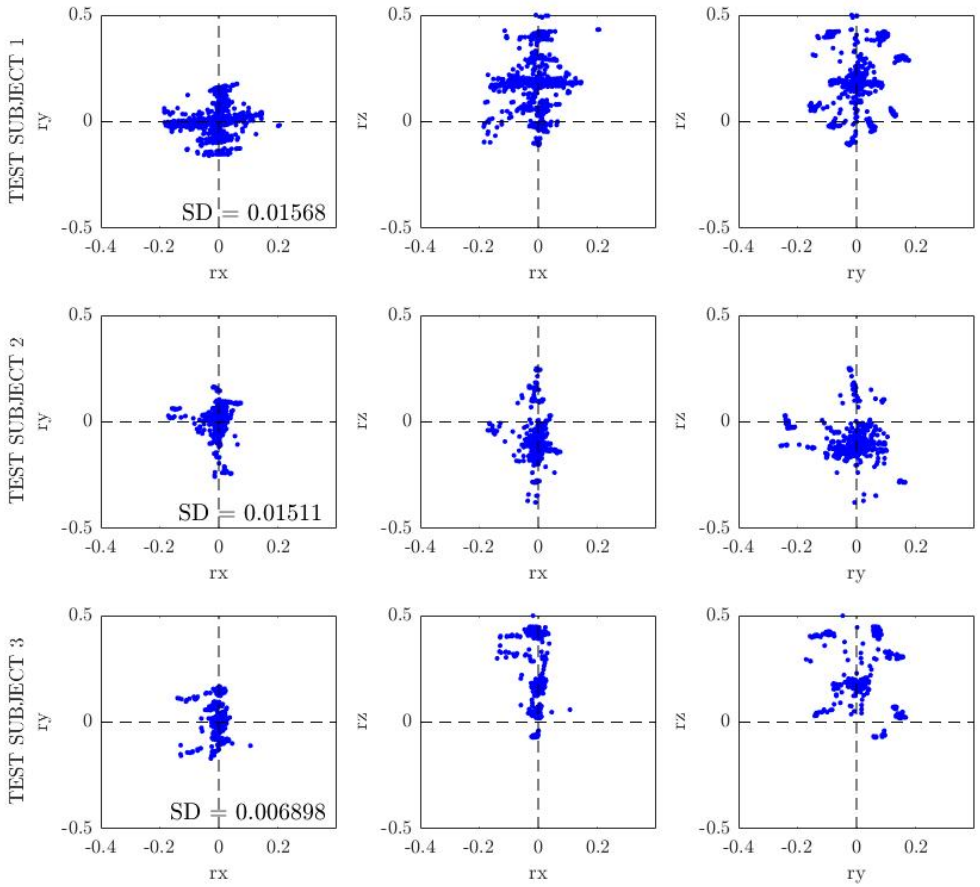

Figure 16: Rotation vector plot for the head still situation corrected for tilted plane.

4 Discussion

Experiment 1: Ocular counter-roll

The amount of torsion could be measured reliably for all tests subjects except for test subject four. This data was thereof left out of consideration for the rest of the results. The measurements were bad, probably due to the fact that this participant was wearing contact lenses which gave bad pupil tracking with the pupil labs software in the first place, which caused the template matching to be performed on an almost random part of the video instead of real matching of the iris. This however does show that tracking and thus determining the amount of torsion is maybe not reliable with test subjects that are wearing lenses en further investigation needs to be done to see if improvements can be made at this point.

With the measurements of 5 degrees at 0.2 hz. the synchronization step failed with two test subjects. This was probably caused by the small amplitude of the stimuli and the torsion not being symmetric around zero but around minus some degrees, making the synchronization harder to perform. For small angles the synchronization thus needs to be optimized as well.

The outlier in the measured gains (Figure 13), the 5 degrees and 0.1 hz. stilmuli for test subject two, could be explained by the fact that it was the first stimuli for the test subject and due to this the head was not kept still. Another explanation can come from the fact that this was the smallest stimuli we have presented the subjects with and the sensibility of the VOR for this test subject is less then for the other test subjects. The trend that the gain decreases at greater amplitudes is probably the consequence of the physical limitation on the amount of torsion the eye can generate. Collewijn, at al. (1985) found this same trend in their measurements as well. Additionally, they also found that the gain gets bigger for the higher frequencies and this trend was visible with these

measurement too. For future research more test subjects can be measured and then statistical analysis can be performed on these trends in gain as well.

Experiment 2: Saccadic movements

The rotation vectors of the pilot experiment showed listing's plane clearly, where the data of the experiments executed in the vestibular chair still have a quite a lot of big torsional components. This could originate from more noise due to the changed circumstances or the adjusted translation to world reference frame for the vestibular chair set-up. Additionally, in the eccentric eye orientations a lot of data is lost by the filtering process. To be able to improve this measurement saccades could be restricted to a smaller amplitude, eyelids could be taped up and down to keep them from obstructing the iris and the pupil size could be increased to visualize more iris. With less data loss, the data will be easier to interpret and all these improvements also facilitate the template matching. This could lead to a reduction of the amount of noise in the measurement. All these improvements would help the visualization of listing's plane; however, if the broadness of the plane is originating from the change in reference frame this will not fix the whole problem and thus further testing of this translation is needed, to be able to exclude this cause.

General discussion

The method does measure actual torsional eye movements, especially during the first experiment the results were as was expected. However, since the method still gives some unexpected results improvements for the method are needed to be able to use it in other research and the clinics. These improvements should focus on the robustness for different test subjects, the prevention of eyelid occlusion and the decrease of the pupil size. The first step could be to let a test subject look straight ahead and then take a small video of this. Then the template can be created out of this data and later the video frames of the experiments can all be compared to this template. This is done to make sure that there is no blink occurring in the image of the template and to determine the iris size. A second step could be to tape the upper eyelid up an the lower down to prevent them from blocking the iris with blinks or in eccentric positions. This can only be done in experiments that do not need a lot of time. To decrease the pupil size, more light can be generated in the experiment room, but then the LEDs in the frame will be more difficult to locate due to decreased visibility, so a small extra light near the eye could be a better solution for this. Additionally, the head of the test subjects could not be fixed in these experiments due to the lack of a head fixed set up in the lab. For future experiments having a real head fixed up set up could help to improve on the algorithm and make results more manageable. As an alternative the head tracker data could be incorporated in the analyses, so that compensation can be made for the movements that are made during the experiments.

Finally if other researchers are going to use this method for their experimental paradigms the analyses needs to be made easier to work with, because a lot of different steps are needed to be taken at this point. The creation of one good template already helps this process, but the ultimate goal would be for the researcher to put its data in and to only set some parameters and then receives output of the torsion measured in a workable format. Whilst all the parts of the algorithm are there the connection still needs to be scripted to achieve this goal.

5 Conclusion

With the method described in this thesis it was possible to measure torsional eye movements during verstibular stimulation. However, the signal to noise ratio depended greatly on the test subject. In eccentric eye positions a lot of data was lost due to bad confidence or cross correlation, due to either obstruction by the eyelids or video angle. Therefore, it was not possible to show listing's plane in our experiments but in optimal situation for the pupil labs tracker our pilot did show listing's plane. Nevertheless, by implementing the improvements and thus having smaller pupils, less obstructions and less manual handwork this technique has potential to be used in other research and in the clinic in the future.

References

[Brown and Marmor, 2006] Brown, M. and Marmor, M. e. a. (2006). ISCEV Standard for Clinical Electro-oculography (EOG). *Doc Ophthalmol*, (113):205–212.

[Collewijn, 1985] Collewijn, H., V. d. S. J. F. L. e. a. (1985). Human ocular counterroll: assessment of static and dynamic properties from electromagnetic scleral coil recordings. *Exp Brain Res*, 59:185–196.

[Fick, 1854] Fick, A. (1854). Die bewegnngen des menschlichen augapfels. *Zeitschrift fiir rationelle Medizin*, 4:109–128.

[Jin et al., 2020] Jin, N., Mavromatis, S., Sequeira, J., and Curcio, S. (2020). A Robust Method of Eye Torsion Measurement for Medical Applications. *Information*, 11(9):408.

[Leigh, 2015] Leigh, R., . Z. D. (2015). *The Neurology of Eye Movements*. Oxford University Press, Oxford, UK.

[Ong and Haslwanter, 2010] Ong, J. K. and Haslwanter, T. (2010). Measuring torsional eye movements by tracking stable iris features. *Journal of neuroscience methods*, 192(2):261–267.

[OpenCV,] OpenCV. Opencv software.

[Otero-Millan et al., 2015] Otero-Millan, J., Roberts, D. C., Lasker, A., Zee, D. S., and Kheradmand, A. (2015). Knowing what the brain is seeing in three dimensions: A novel, noninvasive, sensitive, accurate, and low-noise technique for measuring ocular torsion. *Journal of vision*, 15(14):11.

[Purves and Augustine, 2010] Purves, d. and Augustine, G. J., e. a. (2010). *Neuroscience*. Sinauer Associates Inc, London :.

[van Opstal, 2004] van Opstal, A. (2004). *Psychofysica 2 dictaat: 3D rotations*.

[van Opstal et al., 1991] van Opstal, A., Hepp, K., Hess, B., Straumann, D., and Henn, V. (1991). Two- rather than three-dimensional representation of saccades in monkey superior colliculus. *Science*, 252(5010):1313–1315.

[Van Opstal et al., 1996] Van Opstal, J., Hepp, K., Suzuki, Y., and Henn, V. (1996). Role of monkey nucleus reticularis tegmenti pontis in the stabilization of listing's plane. *Journal of Neuroscience*, 16(22):7284–7296.

[Van Veen, 2019] Van Veen, D. (2019). Improving measurements torsional eye movements. *bachelor thesis*.

[von Helmholtz, 1866] von Helmholtz, H. (1866). *Handbuch der Physiologischen Optik*. Oxford University Press.

Appendix

A Matlab scripts

A.1 Creating iris template and determining torsion

```
addpath(genpath('Y:\Researcher\AnnemiekBarsingerhorn\functionsA'));
addpath(genpath('Y:\Researcher\AnnemiekBarsingerhorn\Torsion'));
addpath(genpath('X:\Students\Eva\DATA'))
addpath(genpath('X:\Students\Eva\extramap'));
addpath('C:\ProgramData\MATLAB\SupportPackages\R2021a\toolbox\vision\supportpackages\vi
sionopencv\example\TemplateMatching');

%% convert data to usefull format
result=pldata2mat('C:\Users\annem\Downloads\2021_04_13\008\', 'gaze'
,'C:\Users\annem\Downloads\2021_04_13\008\Pupildata.mat');

cdir = pb_getdir('cdir','X:\Students\Eva\DATA');
fn = pb_zipblocks(cdir);
D = pb_convertdata(fn);


%% Load pupil-labs data directory
%addpath(genpath('X:\Students\Eva\DATA'));
cd('X:\Students\Eva\DATA\2021_05_27');
dirpl = dir;


for k=1:7 % to facilitate proccesing multiple recordings in one run
    %% create template

    pupildata=1; % if you want to use pupil data ellipses change to 1
    indexx =100; % frame number at which participant looks straight ahead

    TimestampsEye = readNPY('eye0_timestamps.npy');
    TimestampsPupil = readNPY('gaze_timestamps.npy');

    [val, ind]=min(abs(TimestampsPupil-TimestampsEye(1)));
    ind=ind-1;

    %load the pupil labs data of current block
    data = sprintf('X:\\Students\\Eva\\DATA\\2021_05_27\\%s', dirpl(k).name);
    cd(data);
    load('Pupildata.mat')

    clear gaze_normalstotal circlenormal confidence

    for j=1:length(gaze)
        gaze_normalstotal(:,j)=gaze{1,j}.gaze_normal_3d;
        circlenormal(:,j)=gaze{1,j}.base_data{1, 1}.circle_3d.normal;

        confidence(j)=gaze{1,j}.confidence;
    end

    normv=median(gaze_normalstotal(:,indexx+ind:indexx+ind+40)'); % normv die je had
klopte niet.

% Estimate rotation matrix
    GG = @(A,B) [ dot(A,B) -norm(cross(A,B)) 0;
        norm(cross(A,B)) dot(A,B)  0;
        0              0           1];

    FFi = @(A,B) [ A (B-dot(A,B)*A)/norm(B-dot(A,B)*A) cross(B,A) ];

    UU = @(Fi,G) Fi*G*inv(Fi);
    b=normv'; a=[0 0 1]';
    Rot = UU(FFi(a,b), GG(a,b));

    gaze_normalstotalRotated=gaze_normalstotal'*Rot;
```

```
    normv=median(circlenormal(:,indexx+ind:indexx+ind+40)'); % normv die je had klopte
niet.

% Estimate rotation matrix
    GG = @(A,B) [ dot(A,B) -norm(cross(A,B)) 0;
        norm(cross(A,B)) dot(A,B)  0;
        0              0           1];

    FFi = @(A,B) [ A (B-dot(A,B)*A)/norm(B-dot(A,B)*A) cross(B,A) ];

    UU = @(Fi,G) Fi*G*inv(Fi);
    b=normv'; a=[0 0 1]';
    Rotc = UU(FFi(a,b), GG(a,b));

    circlenormalRotated=circlenormal'*Rotc;
    circlenormalRotated=circlenormalRotated';


    [AzElEye]=-
xyz2azelAnnemiek(gaze_normalstotalRotated(:,1),gaze_normalstotalRotated(:,2),gaze_norma
lstotalRotated(:,3));
    [AzElEyenNonRot]=-
xyz2azelAnnemiek(gaze_normalstotal(1,:),gaze_normalstotal(2,:),gaze_normalstotal(3,:));

    [AzElEyenCircRot]=-
xyz2azelAnnemiek(circlenormalRotated(1,:),circlenormalRotated(2,:),circlenormalRotated(
3,:));
    [AzElEyenCirc]=xyz2azelAnnemiek(-circlenormal(1,:),-circlenormal(2,:),-
circlenormal(3,:));

%% Load video data
    [video, ~] = mmread('eye0.mp4');
% v = VideoReader('eye0.mp4');

%% creating template
    IrisSize= 30; %adjust this value manualy.
    frame=(rgb2gray(video.frames(indexx).cdata)); % raw video frame %convert first
frame to greyscale

    if pupildata % use the ellipses from the pupil-labs software

        EllipsePL=gaze{1,ind+indexx}.base_data{1, 1}.ellipse;

        EllipsePLre=[-deg2rad(EllipsePL.angle) EllipsePL.axes(2)/2 EllipsePL.axes(1)/2
EllipsePL.center(2) EllipsePL.center(1)];

        ell_ptsPL = draw_ellipse(EllipsePLre);

        EllipsePL = fit_ellipse_taubin(ell_ptsPL');
        EllipsePLre=[-EllipsePL(1) EllipsePL(3) EllipsePL(2) EllipsePL(5)
EllipsePL(4)];

    else
        EllipsePLre = detect_pupil('ImgMat', frame,'DebugLevel',2);
        framepol =
pol_transAnnemiek(frame,'DebugLevel',2,'EllPar',EllipsePLre,'IrisSize',IrisSize); %
Polar transformed image

    end
```

```
    framepol =
pol_transAnnemiek(frame,'DebugLevel',2,'EllPar',EllipsePLre,'IrisSize',IrisSize); %
Polar transformed image

%search for the white eyelids and glint if present
    level= graythresh(framepol);
    eyegl= imbinarize(framepol,95); % might need to adjust the threshold manualy

% Make and thicken mask
    eyegl = bwmorph(eyegl,'thicken',5);
    mask = bwmorph(eyegl,'thicken',5);

    cfn = pb_newfig(1);imshow(eyegl);

% Use same strategy as described by paper Otero-Millan
% First gaussian blur, then a combination of gradient of x direction
    g = imgaussfilt(framepol,3,'FilterSize',[1 11]); % parameters might be optimized
further

    [ix, iy] = imgradientxy(g, 'sobel');
    Gx = ix; % only use gradient x direction

% Fill mask with random noise. Might be better to do it before gradient,
% but does not seem to make a big difference.
    Gx(mask==1)=randn(sum(sum(mask==1)),1);


%create template
    template(1:size(Gx,1),1:410)=NaN;
    template(:,26:385) = Gx;
    [row,col] = size(Gx);
    template(:,1:25)=Gx(:,col-25:col-1);
    template(:,386:410)=Gx(:,1:25);

    mask2=abs(template)>4*nanstd(template(:));
    template(mask2==1)=randn(sum(sum(mask2==1)),1);

    cfn = pb_newfig(cfn);
    imshow(template);

    img2=template;
    [px,py] = meshgrid(1:size(img2,2),1:size(img2,1));
    figure(24);
%  colormap copper;
    colormap gray;
    shading interp;
    material default;
    camlight right;
    lighting phong;img2;
    %  lighting none;
    surf(px,py,img2)
    axis off;

%% Main part script
    ElPars.ellipse_pol_rc=[];
    ElPars.prev_fit_valid=false;
    % Pupil detection
    tic
    for j=1:length(video.frames)

        try
            if pupildata % Use ellipses pupil-labs software
                EllipsePL=gaze{1,j+ind}.base_data{1, 1}.ellipse;
```

```
                EllipsePLre=[-deg2rad(EllipsePL.angle) EllipsePL.axes(2)/2
EllipsePL.axes(1)/2 EllipsePL.center(2) EllipsePL.center(1)];

                ell_ptsPL = draw_ellipse(EllipsePLre);

                EllipsePL = fit_ellipse_taubin(ell_ptsPL');

                EllipsePLre=[-EllipsePL(1) EllipsePL(3) EllipsePL(2) EllipsePL(5)
EllipsePL(4)];
                ellipsesPL(j,:)=EllipsePLre;
            else
                frame=(rgb2gray(video.frames(j).cdata)); % raw video frame

                ElPars=detect_pupil('ImgMat',frame,'DebugLevel',0,'SeqFlag',
true,'PupPar',ElPars.ellipse_pol_rc,'PrevFitValid',ElPars.prev_fit_valid,'PixPerMill',
8,'FrameRate', 119);
                EllipsePLre=ElPars.ellipse_pol_rc;
                if ElPars.prev_fit_valid==1
                    ElPars.prev_fit_valid=true;
                else
                    ElPars.prev_fit_valid=false;
                end
                ellipsesfits(j,:)=EllipsePLre;

                ex=gaze{1,ind+1}.base_data{1,1}.projected_sphere.center(:,1);
                ey=gaze{1,ind+1}.base_data{1,1}.projected_sphere.center(:,2);
                er=gaze{1,ind+1}.base_data{1,1}.projected_sphere.axes(:,1);
                cx=EllipsePLre(5);
                cy=EllipsePLre(4);
                er=er/2;

                % recalculate gaze
                % quaternion defining the rotation of the eyeball (ignoring torsion)
just the center of the pupil
                angle = atan2((cy - ey), (cx - ex));
                yC = cy - ey;
                xC = cx - ex;
                ecc = asin(sqrt(yC * yC + xC * xC) / er);
                quat = quaternion(cos(ecc / 2), -sin(angle) * sin(ecc / 2), cos(angle)
* sin(ecc / 2), 0);
                vp=RotateVector(quat,[0 0 1]',1);
                vps(j,:)=vp;
            end

        catch ME
            j
            disp('Problem using ellipse fit:');
            warning(ME.message)
            shift(j)=NaN;
        end

    end
    toc

    if pupildata
        Ellipses=ellipsesPL;
    else
        Ellipses=ellipsesfits;
        [AzElEyefit]=-xyz2azelAnnemiek(vps(:,1),vps(:,2),vps(:,3));

    end
    % Ellipses=medfilt1(ellipses,5);
```

```
    % Ellipses(1,:)=medfilt1(Ellipses(1,:),5);
    % Ellipses(2,:)=medfilt1(Ellipses(2,:),5);
    % Ellipses(3,:)=medfilt1(Ellipses(3,:),5);
    % Ellipses(4,:)=medfilt1(Ellipses(4,:),5);
    % Ellipses(5,:)=medfilt1(Ellipses(5,:),5);

    % clear shift

    cfn = pb_newfig(cfn);
    tic
    %polar transform en template matching
    for j=1:length(video.frames)

        frame=(rgb2gray(video.frames(j).cdata)); % raw video frame
        disp(j);

        try
            framepol =
pol_transAnnemiek(frame,'DebugLevel',0,'EllPar',Ellipses(j,:),'IrisSize',IrisSize); %
Polar transformed image

            eyegl= imbinarize(framepol,95);   % might need to adjust the threshold
manualy
            eyegl = bwmorph(eyegl,'thicken',5);

            mask = bwmorph(eyegl,'thicken',5);


            g = imgaussfilt(framepol,3,'FilterSize',[1 11]); % parameters might be
optimized further

            [ix, iy] = imgradientxy(g, 'sobel');
            Gx = ix;

            mask2=abs(Gx)>4*nanstd(Gx(:));

            Gx(mask==1)=randn(sum(sum(mask==1)),1);

            Gx(mask2==1)=randn(sum(sum(mask2==1)),1);
            result = matchTemplateOCV(uint8(Gx), uint8(template(1:size(Gx,1),:)));

            xint=1:0.02:length(result);
            results=interp1(1:length(result),result,xint,'spline');
            [val, idx] = max(abs(results(:)));
            shift(j)=xint(idx);
            vals(j)=val;


        catch ME
            j
            disp('Problem using polar transform:');
            warning(ME.message)
            shift(j)=NaN;
        end
    end
    toc


    torsion=shift-26;

    cfn = pb_newfig(cfn)
    plot(shift-26)
    hold on
```

```
    plot(AzElEye(1+ind:end,:))
    plot(medfilt1(shift-26,10))
    plot(medfilt1(shift-26,20))
    plot(vals*10)

    %% save data
    save('DataTorsionIrissize30thres95frame150.mat')
    %% clear figures and variables
    close all
    clearvars -except dirpl irissizef tresholdf dirpl2
end
```

```
%% Load data
cd('X:\Students\Eva\DATA\2021_05_27');
dirpl = dir;

data =
sprintf('X:\\Students\\Eva\\DATA\\2021_05_27\\%s\\DataTorsionIris45Threshold98frame100.
mat', dirpl(10).name);
load(data);


%% Plot filtered data
cfnn = 1;
amp = [5 15 30 45 5 15 30];
freqt = ["0.1" "0.1" "0.1" "0.1" "0.2" "0.2" "0.2"];
cfnn = pb_newfig(cfnn);
for j=10:17
    q = j-9;
    data =
sprintf('X:\\Students\\Eva\\DATA\\2021_05_27\\%s\\DataTorsionIris45Threshold98frame100.
mat', dirpl(j).name);
    str =['line1', '\n', 'Line2'];
    %nametitlefilt = sprintf('Filtered data of recording %s: \n Amplitude: %d degrees,
Frequency: %s hz.', dirpl(j).name, amp(q), freqt(q));
    filenamesave =
sprintf('X:\\Students\\Eva\\DATA\\2021_05_27\\%s\\Torsionhighconfdata.mat',
dirpl(j).name);
    load(data)

    torsionfilt=medfilt1(shift-26,20); % Median filter to filter torsion estimation
    torsionfilt(abs(torsionfilt)>20)=NaN; % exclude data if torsion is high, be careful
though because you don't want too exclude actual data
    AzElEye(abs(AzElEye)>10)=NaN;
    for j=1:length(gaze)-ind
        conf(:,j)=gaze{1,j+ind}.confidence; % Pupil-labs confidence of ellipse fit
    end
    torsionhighconf=torsionfilt;
    torsionhighconf(conf<0.6)=NaN; % Exclude data with low confidence (pupil-labs
manufacterer indicates <0.6 is not accurate)
    torsionhighconf(vals(1:9500-ind)<0.2)=NaN; %exclude data with low cross
correlation.

    save(filenamesave, 'torsionhighconf') %saved to be used for the synchronization and
fitting of the data later on.

    subplot(2,4,q)
    plot(vals*10)
    hold on
    plot(conf*10)
    plot(torsionhighconf, 'LineWidth', 0.8,'color', 'k')
    %plot(AzElEye(1+ind:end,:))
    xlim([0 7800])
    ylim([-30 30])
    %title(nametitlefilt)
    xlabel('Framenumber')
    ylabel('Angle (degree)')
    %legend({'Torsion', 'Azimuth', 'Elivation', 'cross correlation factor (x10)',
'confidence (x10)'}, 'Location','northwest','NumColumns',2)
    hold off
    clearvars -except dirpl cfnn amp freq
end
lgd = legend({'Torsion', 'Cross correlation factor (x10)', 'Confidence pupil ellipse
(x10)'}, 'Location','northwest','NumColumns',1);
lgd.FontSize=11;
```

A.3 Synchronization and fitting of sine wave

```
%cd('X:\Students\Eva\DATA');
jjj = 3; %adjust to test subject.
load('X:\Students\Eva\DATA\ES-0003-21-05-27\converted_data_ES-0003-21-05-27.mat');

cd('X:\Students\Eva\DATA\2021_05_27');

dd=dir('0*');
cfn=0;
cfn = pb_newfig(cfn);
hold on
%% making the graph
% cfn=0;
% cfn = pb_newfig(cfn);
% hold on

for jj=10:16
    cd([dd(jj+17).folder '\' dd(jj+17).name])

%cd([dd(jj).folder '\' dd(jj).name])

        if jjj==1
    load('DataTorsionIris45Threshold98frame100.mat') % data torsion
    load('Torsionhighconfdata.mat')
        elseif jjj==2
    load('DataTorsionIris35Threshold90frame150.mat')
    load('Torsionhighconfdata.mat')

        elseif jjj==3
            if jj == 13
    load('DataTorsionIris30Threshold95frame300.mat')
    load('Torsionhighconfdata.mat')
            elseif jj == 15
    load('DataTorsionIris30Threshold95frame300.mat')
    load('Torsionhighconfdata.mat')
            else
    load('DataTorsionIris30Threshold95frame150.mat')
    load('Torsionhighconfdata.mat')
            end
        end

    blocknumber=jj;
    [~,inds] = min(abs(D(jj).Pup.Data.timestamp-TimestampsEye(1)));

    FsVC = 10;



    sensehat_posD     = rad2deg(cumsum(D(blocknumber).Sensehat.gyro_x -
D(blocknumber).Sensehat.gyro_x(1)))/-100;
    sensehat_posD     = sensehat_posD - sensehat_posD(1);      % Force sine start at 0
    vestibular_posD     = pb_cleanSP(D(blocknumber).VC.pv.horizontal); % Strip tail
from VC signal
    % vestibular_posD   = vestibular_posD/max(abs(vestibular_posD));
    tsVestibular        = (0:length(vestibular_posD)-1)/FsVC; % Create VC timestamps
(0:0.1:Nx)

    %  Interpolate vestibular data
    tsSense=D(jj).Timestamp.Sense-D(jj).Timestamp.Sense(1);
    vestibular_posDI  = interp1(tsVestibular, vestibular_posD, tsSense, 'pchip');
    tsVestibularI     = tsSense;

    %  Clip extrapolation
```

```
    inds                    = find(tsVestibularI >= max(tsVestibular));  % find index
extrapolated values
    vestibular_posDI(inds)  = [];
    tsVestibularI(inds)     = [];

    %  XCorr synchronization
    fsPup       = length(tsVestibularI)/tsVestibularI(end);
    [r,lag]     = xcorr(vestibular_posDI,sensehat_posD);
    [~,I]       = max(abs(r));
    lagDiff     = lag(I)/fsPup;
    tsSense     = tsSense+lagDiff;                                              %
Correct lagdiff in Sensets

    % Correct timestamps
    [~,inds] = min(abs(D(jj).Pup.Data.timestamp-TimestampsEye(1)));

    tsVestibularI=tsVestibularI+D(jj).Timestamp.Sense(1)-D(jj).Timestamp.Pup(inds)-
lagDiff;
%     tsVestibularIall1 = [

      %tsVestibularIall1.(jj, :) = tsVestibularI(1:4100);
%     vestibular_posDIall1(jj, :) = vestibular_posDI(1:4100);
%     puptimeall1(jj, :) = D(jj).Timestamp.Pup(inds:9000)-D(jj).Timestamp.Pup(inds);
%     torsiontimeall1(jj, :) =
torsionhighconf(1:length(D(jj).Timestamp.Pup(inds:9000)));

    data3(jj-9).tsVestibularI = tsVestibularI;
    data3(jj-9).vestibular_posDI = vestibular_posDI;
    data3(jj-9).puptime =  D(jj).Timestamp.Pup(inds:9000)-D(jj).Timestamp.Pup(inds);
    data3(jj-9).torsiontime =
torsionhighconf(1:length(D(jj).Timestamp.Pup(inds:9000)));

    subplot(2,4,jj-9)
    hold on
    %plot(D(jj).Timestamp.Pup(inds:end)-
D(jj).Timestamp.Pup(inds),shift(1:length(D(jj).Timestamp.Pup(inds:end)))-26)
    plot(D(jj).Timestamp.Pup(inds:9000)-
D(jj).Timestamp.Pup(inds),torsionhighconf(1:length(D(jj).Timestamp.Pup(inds:9000))))
    xlim([0 68])
    ylim([-50 50]) %ylim([min(vestibular_posDI)-5 max(vestibular_posDI)+5])
    xlabel('Time(s)')
    ylabel('Angle(deg)')
    plot(tsVestibularI,vestibular_posDI)



% Adjust for drift in yVOR signal

[xV,~]   = ginput(2); % select 2 peaks, one at the beginning and one at the end of the
signal (either 2 lower peaks, or 2 higher peaks)

[~, ind1]=min(abs(tsVestibularI-xV(1)));
[~, ind2]=min(abs(tsVestibularI-xV(2)));


%  Get fit parameters,zzzsdFzgh
parVor   = sineFit(tsVestibularI(ind1:ind2),vestibular_posDI(ind1:ind2));

ft = fittype('sin(2*pi*freq*x+shift)*yscale','coefficients',{'shift','freq','yscale'});
mdlVEST =
fit(tsVestibularI(ind1:ind2)',vestibular_posDI(ind1:ind2)',ft,'startpoint',[parVor(4),p
arVor(3),parVor(2)]);
```

```
fVEST2=mdlVEST.yscale*sin(2*pi*mdlVEST.freq*tsVestibularI+mdlVEST.shift);

tsTors=D(jj).Timestamp.Pup(inds:end)-D(jj).Timestamp.Pup(inds);
TorsSig= -torsionhighconf(1:length(D(jj).Timestamp.Pup(inds:9000)));
%shift(1:length(D(jj).Timestamp.Pup(inds:end)))-26;
tsTors(isnan(TorsSig))=[];
TorsSig(isnan(TorsSig))=[];


[~, ind1]=min(abs(tsTors-xV(1)));
[~, ind2]=min(abs(tsTors-xV(2)));

ft = fittype('sin(2*pi*freq*x+shift)*yscale','coefficients',{'shift','freq','yscale'});
mdlVOR = fit(tsTors(ind1:ind2)',TorsSig(ind1:ind2)'-
median(TorsSig(ind1:ind2)),ft,'startpoint',[parVor(4),parVor(3),parVor(2)*0.4]);
%adjusted from to - and 0.2 --> 0.4
fVOR2=mdlVOR.yscale*sin(2*pi*mdlVOR.freq*tsTors+mdlVOR.shift)+median(TorsSig(ind1:ind2)
);

data3(jj-9).tsfit = tsTors;
data3(jj-9).fit = -fVOR2;

plot(tsTors,-fVOR2,'c', 'DisplayName', 'Fitted torsion');
%plot(tsVestibularI,fVEST2);

relPhase2 = mdlVOR.shift-mdlVEST.shift-pi;
gain2     = abs(mdlVOR.yscale/mdlVEST.yscale);
Gains(jjj, jj-9)=gain2;


subplot(2,4,jj-9)
title(['Amplitude: ' num2str(D(jj).Block_Info.signal.hor.amplitude) ' deg, Frequency: '
num2str(D(jj).Block_Info.signal.hor.frequency) ' Hz']);
txt = ['Gain=' num2str(gain2)];
text(40, -45, txt, 'FontSize', 9)
% lgd1 = legend('Torsion','Vestibular chair','Fitted torsion');
% lgd1.FontSize = 11;
end


lgd = legend('Torsion','Vestibular chair','Fitted torsion');
lgd.FontSize = 11;
        savefig('TorsRotFigNew12')
        save('datavest', 'data3', 'Gains')
%clearvars -except D cfn dd jjj Gains
%end
for jj = 1:7
    subplot(2,4,jj)
    %A = ['A' 'B' 'C' 'D' 'E' 'F' 'G'];
    %title([A(jj) ': amplitude: ' num2str(D(jj).Block_Info.signal.hor.amplitude) ' deg,
frequency: ' num2str(D(jj).Block_Info.signal.hor.frequency) ' Hz']);
    yticks([ -45 -30 -15 -5 0 5 15 30 45])
end

%% plot gains
cfn = pb_newfig(cfn);
plot(1:4, Gains,'.','MarkerSize',20)
%plot(5:8, Gains,'o','MarkerSize',20)
xticks(1:4, Gains)
xticklabels({' 5 deg','0.1 Hz & 15 deg','0.1 Hz % 30 deg','0.1 Hz &45 deg'})
ylabel ('Gain Torsional VOR')

gain1 = Gains(1, :);
```

```
gain2 = Gains(2, :);
gain3 = Gains(3, :);
%% create gain plot
cfn = pb_newfig(cfn);
plot(gain1(1:4), '.r', 'MarkerSize', 20)
hold on
plot(gain2(1:4), '.b', 'MarkerSize', 20)
plot(gain3(1:4), '.m', 'MarkerSize', 20)
plot(gain1(5:7), 'or', 'MarkerSize', 8)
plot(gain2(5:7), 'ob', 'MarkerSize', 8)
plot(gain3(5:7), 'om', 'MarkerSize', 8)
xlim([0.5 4.5])
ylim([0 0.65])
xticks([1 2 3 4])
xticklabels({ '5 deg','15 deg','30 deg','45 deg'})
ylabel ('Gain Torsional VOR')
xlabel ('Angle vestibular chair')
lgd1 = legend ({'test subject 1 (0.1 hz)','test subject 2 (0.1 hz)', 'test subject 3
(0.1 hz)','test subject 1 (0.2 hz)','test subject 2 (0.2 hz)', 'test subject 3 (0.2
hz)'}, 'NumColumns', 2);
lgd1.FontSize = 9;
```

A.4 Correction to world reference frame

```
%% Filter data and create quaternions
torsion=medfilt1(shift-26,20); % Median filter to filter torsion estimation
torsion(abs(torsion)>20)=NaN; % exclude data if torsion is high
for j=1:length(gaze)-ind
    conf(:,j)=gaze{1,j+ind}.confidence; % Pupil-labs confidence of ellipse fit
end
torsionhighconf=torsion;
torsionhighconf(conf<0.6)=NaN; % Exclude data with low confidence (pupil-labs
manufacterer indicates <0.6 is not accurate)
torsionhighconf(vals(1:length(gaze)-ind)<0.15)=NaN; % Exclude data with low cross
correlation during template matching.

% Create quaternions for straigh ahead position
qEye  = quaternion.rotateutov([0 0 1]',-circlenormal(:,1+ind:end),1,1); % normal of the
pupil ellipse fit in eye-cam frame
qEye0  = quaternion.rotateutov([0 0 1]',gaze_normalstotal(:,indexx+ind),1,1); % normal
of the pupil ellipse fit in world-cam frame
Eye0Mat=RotationMatrix(qEye0);

qTorsion=quaternion.angleaxis(deg2rad(-torsionhighconf),[0 0 1]); % quaternion of
torsion estimated based on camera images

[qTorsion,n] = normalize(qTorsion); % normalize quaternion
qGaze  = quaternion.rotateutov([0 0 1]',gaze_normalstotal(:,1+ind:end),1,1); %
quaternion of orientation of the eye in world-cam frame


lengths=min([length(qTorsion) length(qEye)]);
for j=1:lengths
    qTot(j)=qEye(j)*qTorsion(j);
    qTotGaze(j)=qGaze(j)*qTorsion(j); % combine into one quaternion representing 3D eye
orientation in world-cam reference frame
    rvGaze(j,:)=vector(qTotGaze(j))/ real(qTotGaze(j));
end

MatrixGaze=RotationMatrix(qGaze);
for j=1:length(qGaze)
MatrixEyeRotTransNew(:,:,:,j)=[MatrixGaze(:,:,:,j) EyeCenterWorld(j,:)'; 0 0 0 1];
end

%% Correct plane
TimestampsEye = readNPY('eye0_timestamps.npy');
TimestampsPupil = readNPY('pupil_timestamps.npy');
TimestampsWorld = readNPY('world_timestamps.npy');

[video, ~] = mmread('world.mp4', 1:400);

index=indexx;
[val, indWorld]=min(abs(TimestampsWorld-TimestampsEye(index))); % synch of cameras

close all
figure
imshow(histeq(video.frames(indWorld).cdata))
[x, y]=ginput(5);


for j=1:length(x)
    PixelPoints(j,:)=[x(j) y(j)];
Points=unprojectpoints([x(j) y(j)]);
unprojectedpointsSpeakersnorm(j,:)=Points;

unprojectedpointsSpeakers(j,:)=unprojectedpointsSpeakersnorm(j,:)*500;
```

```
distspeakerstot(j)=norm(unprojectedpointsSpeakers(j,:)-unprojectedpointsSpeakers(1,:));
distspeakershor(j)=unprojectedpointsSpeakers(j,1)-unprojectedpointsSpeakers(1,1);
distspeakersver(j)=unprojectedpointsSpeakers(j,2)-unprojectedpointsSpeakers(1,2);

end


% Real LEDs positions in vestibular chair frame
xyz=azel2xyz([0 -1 41 -1 -45.5],[0 12.9 -0.4 -14 -0.2]);
LEDs=xyz.*[0.7853 0.8051 0.832 0.7971 0.8313]';

% determining the dinstances between the LEDs
DistancesLEDs(1)=norm(LEDs(1,:)-LEDs(2,:));
DistancesLEDs(2)=norm(LEDs(1,:)-LEDs(3,:));
DistancesLEDs(3)=norm(LEDs(1,:)-LEDs(4,:));
DistancesLEDs(4)=norm(LEDs(1,:)-LEDs(5,:));
DistancesLEDs(5)=norm(LEDs(2,:)-LEDs(3,:));
DistancesLEDs(6)=norm(LEDs(3,:)-LEDs(4,:));
DistancesLEDs(7)=norm(LEDs(4,:)-LEDs(5,:));
DistancesLEDs(8)=norm(LEDs(5,:)-LEDs(1,:));
DistancesLEDs(9)=norm(LEDs(2,:)-LEDs(4,:));
DistancesLEDs(10)=norm(LEDs(3,:)-LEDs(5,:));


b=[1 1 1 1 1]';
fval=100;
cnt=1;
while cnt<10
[b, fval]=FindLocSpeakersNewVC(b',unprojectedpointsSpeakersnorm,DistancesLEDs);
cnt=cnt+1;
end

LocSpeakers=unprojectedpointsSpeakersnorm.*b;
plot3(LocSpeakers(:,1),LocSpeakers(:,2),LocSpeakers(:,3),'o')

xx = LocSpeakers(:,1);
yy = LocSpeakers(:,2);
zz =LocSpeakers(:,3);

N = length(xx);
O = ones(N,1);
C = [xx(~isnan(xx)&~isnan(yy)&~isnan(zz)) yy(~isnan(xx)&~isnan(yy)&~isnan(zz))
O(~isnan(xx)&~isnan(yy)&~isnan(zz))]\zz(~isnan(xx)&~isnan(yy)&~isnan(zz));

x = min(xx):(max(xx)-min(xx))/1000:max(xx);
y =  min(yy):(max(yy)-min(yy))/1000:max(yy);
[xx yy] = meshgrid(x,y);

zzft = C(1)*xx+C(2)*yy + C(3);
hold on;
surf(xx,yy,zzft,'edgecolor','none')

a2=LocSpeakers(2,:)-LocSpeakers(1,:);
a1=LocSpeakers(3,:)-LocSpeakers(1,:);

norms=cross(a1,a2);
norms1=norms/norm(norms);

if norms1(3)<0
    norms1=-norms1;
end
```

```
a3=norms1;
if a1(1)<0
    a1=-a1;
end

if a2(2)<0
    a2=-a2;
end

a1=a1/norm(a1);
a2=a2/norm(a2);


MatRot=[a1' a2' a3'];

PosMatrix=[MatRot LocSpeakers(1,:)'; 0 0 0 1];
GoalMatrix=[eye(3) [0 0 norm(LocSpeakers(1,:))]';0 0 0 1];

Transforms=GoalMatrix/PosMatrix;

LocSpeakersRotTrans=[Transforms*[LocSpeakers'; 1 1 1 1 1]]';

%qRot=quaternion.rotationmatrix(MatRot); %not used


for j=1:length(qGaze)

   MatrixEyeRotTransFormed(:,:,j)=[Transforms*MatrixEyeRotTransNew(:,:,j)];
   normRotTrans(j,:)=MatrixEyeRotTransFormed(1:3,3,j);
      GazeCorrected(j,:)=MatrixEyeRotTransFormed(1:3,3,j);

end

GazeCorrected2=(MatRot*gaze_normalstotalRotated')';
[AzElEyeCorrected]=-
xyz2azelAnnemiek(GazeCorrected(:,1),GazeCorrected(:,2),GazeCorrected(:,3));

AzElEyeCorrected(:,1)=AzElEyeCorrected(:,1)-
median(AzElEyeCorrected(indexx:indexx+40,1)');
AzElEyeCorrected(:,2)=AzElEyeCorrected(:,2)-
median(AzElEyeCorrected(indexx:indexx+40,2)');

[AzElEyeCorrected]=-
xyz2azelAnnemiek(GazeCorrected2(:,1),GazeCorrected2(:,2),GazeCorrected2(:,3));

AzElEyeCorrected(:,1)=AzElEyeCorrected(:,1)-
median(AzElEyeCorrected(indexx:indexx+40,1)');
AzElEyeCorrected(:,2)=AzElEyeCorrected(:,2)-
median(AzElEyeCorrected(indexx:indexx+40,2)');

%% plotting listings plane
range=1000:28000; % select range of data for plots

gaze_normalstotalCorrected=(Transforms(1:3,1:3)*gaze_normalstotal); % transform unit
vectors of gaze into world ref frame (only orientation).

qGaze  = quaternion.rotateutov([0 0 1]',gaze_normalstotalCorrected(:,1+ind:end),1,1); %
transform in quaternions

lengths=min([length(qTorsion) length(qEye)]);
for j=1:lengths
    qTotGaze(j)=qGaze(j)*qTorsion(j); % new 3D quaternions in world ref frame
```

```
    rvGazeNew(j,:)=vector(qTotGaze(j))/real(qTotGaze(j));%Corresponding rotation
vectors

    matqTotGaze=RotationMatrix(qTotGaze(j)); % corresponding rotation matrices
    normqTotGaze(j,:)=matqTotGaze(1:3,3); % Unit vector of gaze in world ref frame

end
```

B Supplementary Results

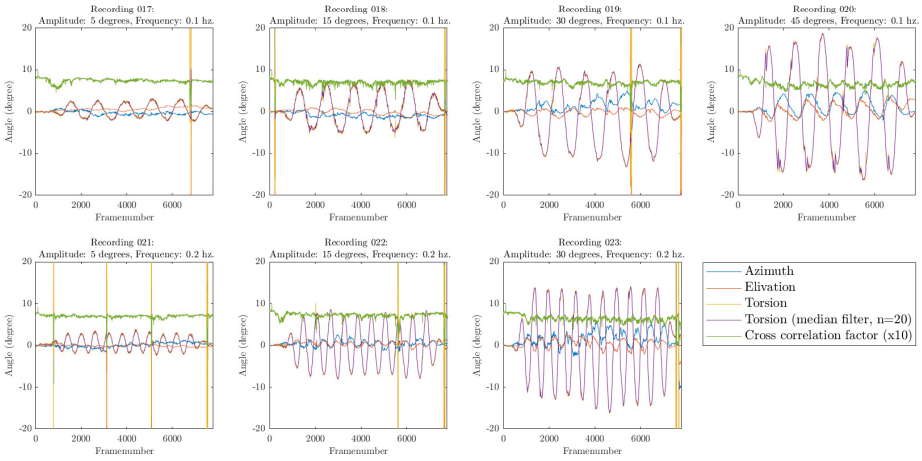

Figure 17: Raw data of test subject 1 during the seven different vestibular stimuli of experiment 1.

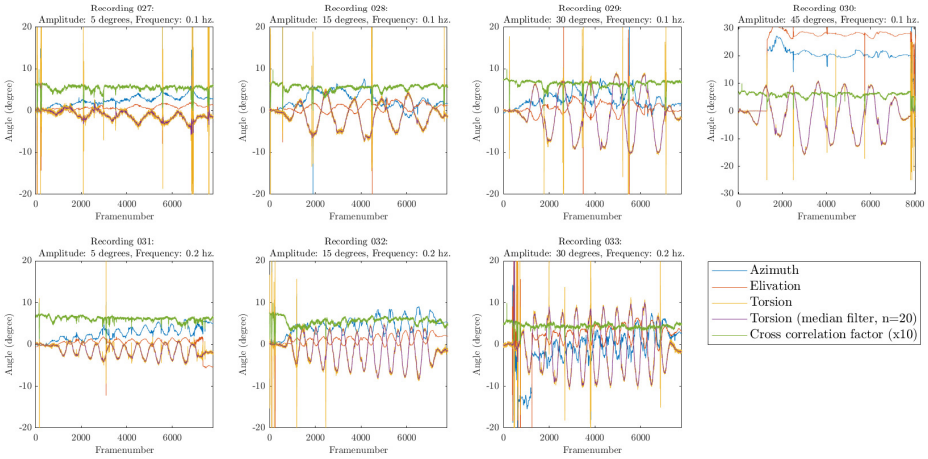

Figure 18: Raw data of test subject 2 during the seven different vestibular stimuli of experiment 1.

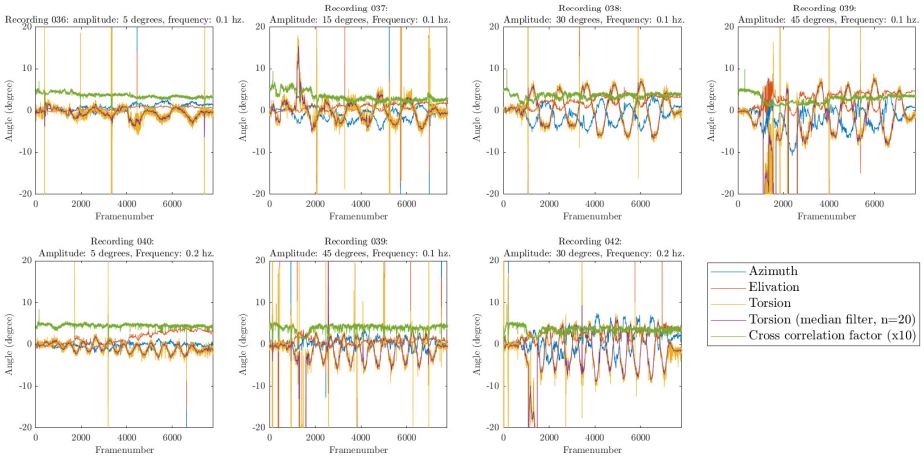

Figure 19: Raw data of test subject 3 during the seven different vestibular stimuli of experiment 1.

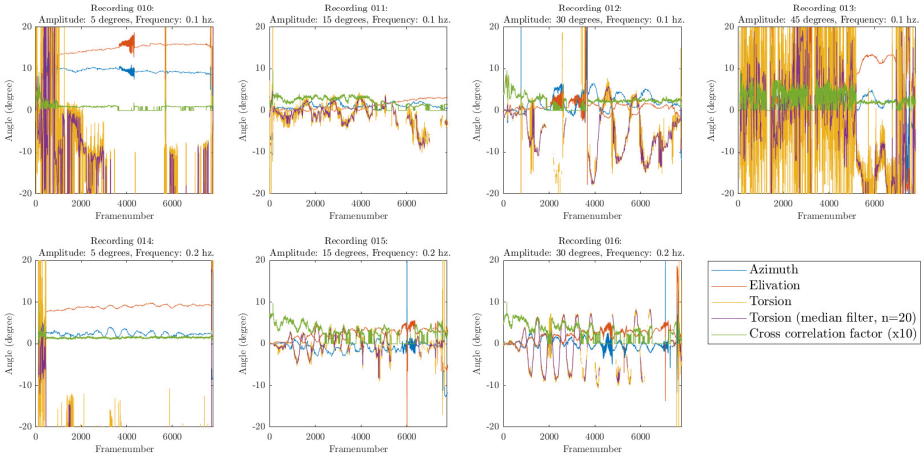

Figure 20: Raw data of test subject 4 during the seven different vestibular stimuli of experiment 1.

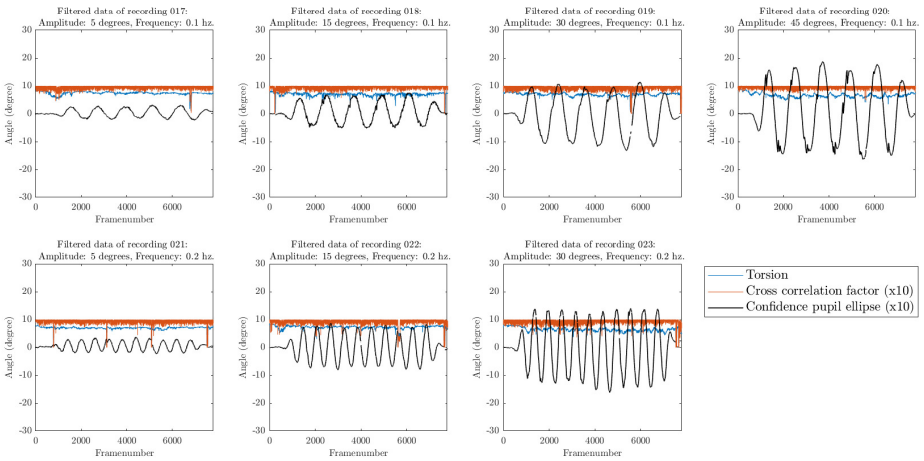

Figure 21: Filtered data of test subject 1

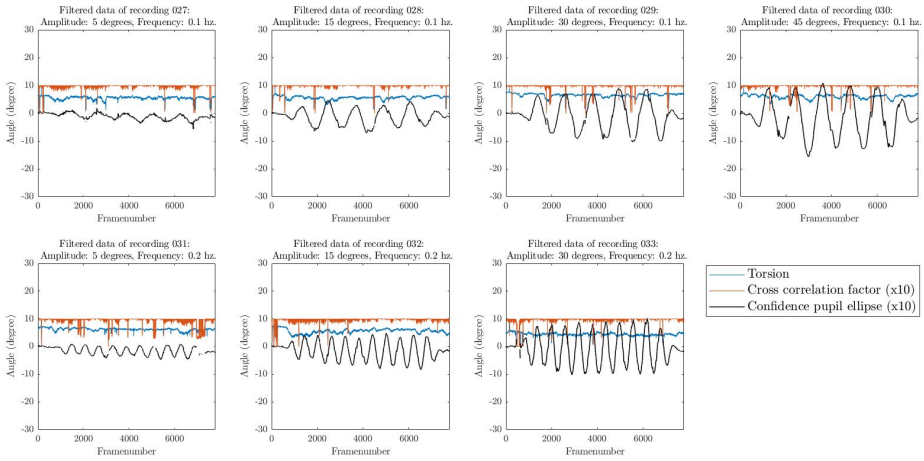

Figure 22: Filtered data of test subject 2

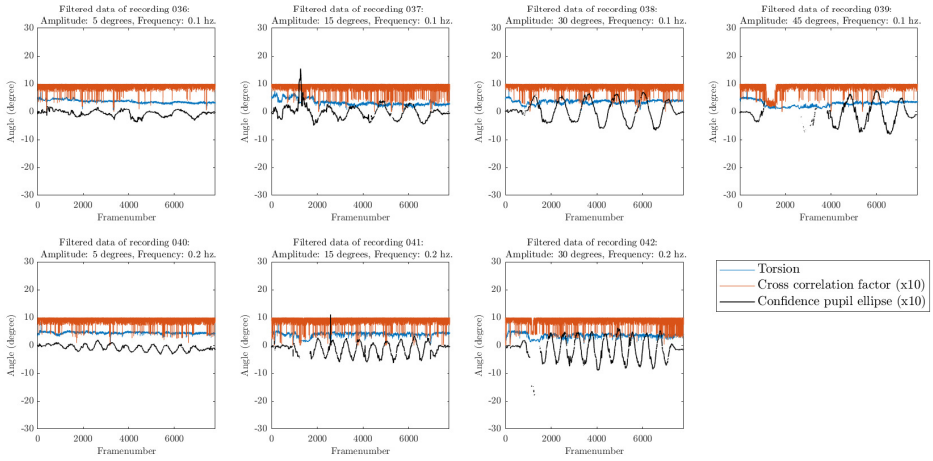

Figure 23: Filtered data of test subject 3

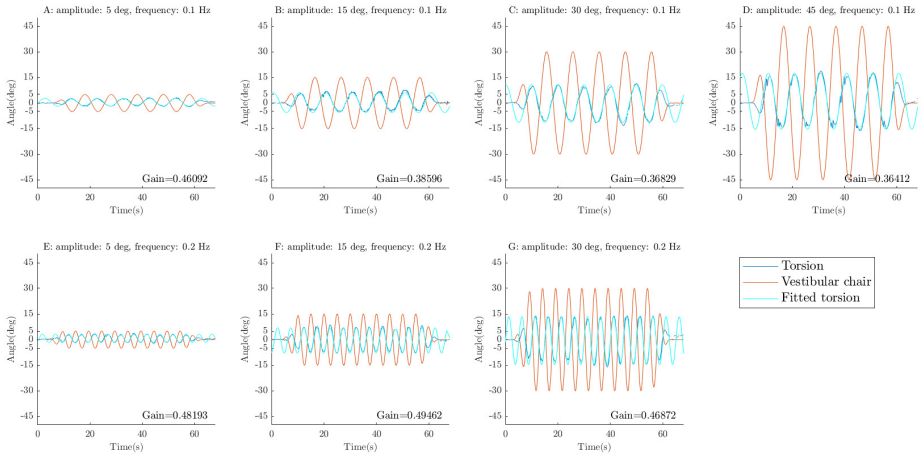

Figure 24: Filtered data of test subject 1

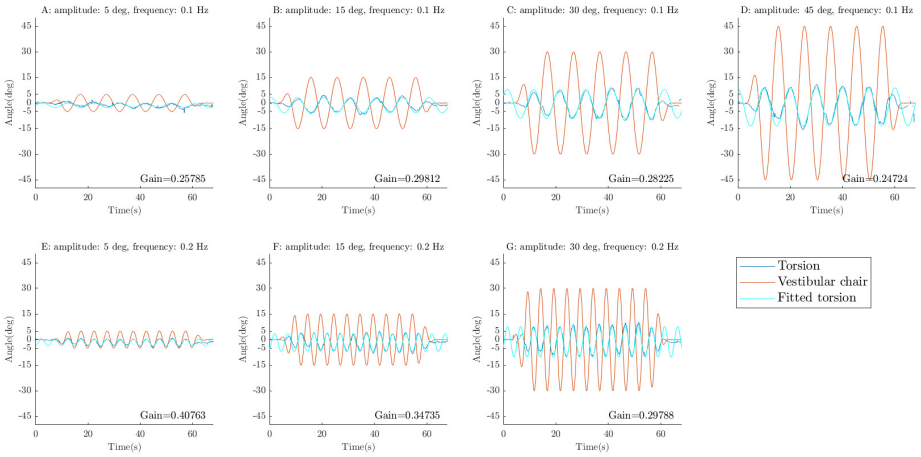

Figure 25: Filtered data of test subject 2

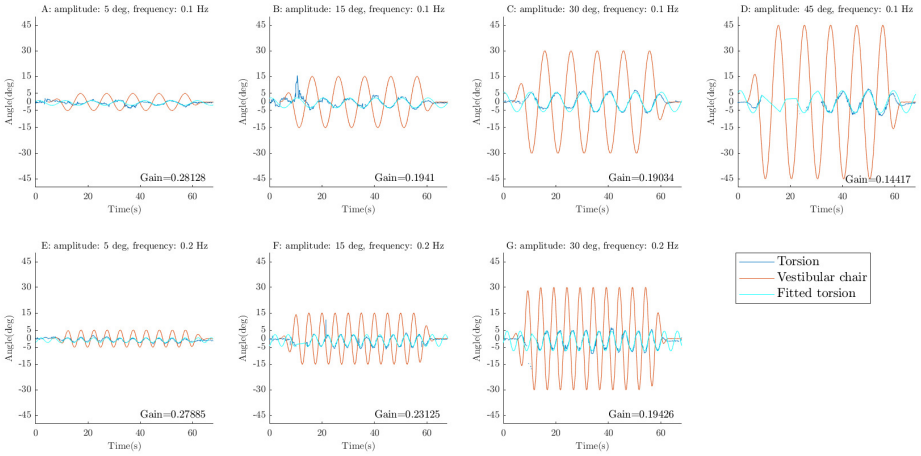

Figure 26: Filtered data of test subject 3