Getting (More) Real: Bringing Eye Movement Classification to HMD Experiments with Equirectangular Stimuli

Ioannis Agtzidis
Technical University of Munich
Munich, Germany
ioannis.agtzidis@tum.de

Michael Dorr
Technical University of Munich
Munich, Germany
michael.dorr@tum.de

ABSTRACT

The classification of eye movements is a very important part of eye tracking research and has been studied since its early days. Over recent years, we have experienced an increasing shift towards more immersive experimental scenarios with the use of eye-tracking enabled glasses and head-mounted displays. In these new scenarios, however, most of the existing eye movement classification algorithms cannot be applied robustly anymore because they were developed with monitor-based experiments using regular 2D images and videos in mind. In this paper, we describe two approaches that reduce artifacts of eye movement classification for 360° videos shown in head-mounted displays. For the first approach, we discuss how decision criteria have to change in the space of 360° videos, and use these criteria to modify five popular algorithms from the literature. The modified algorithms are publicly available at https://web.gin.g-node.org/ioannis.agtzidis/360_em_algorithms. For cases where an existing algorithm cannot be modified, e.g. because it is closed-source, we present a second approach that maps the data instead of the algorithm to the 360° space. An empirical evaluation of both approaches shows that they significantly reduce the artifacts of the initial algorithm, especially in the areas further from the horizontal midline.

CCS CONCEPTS

• **Applied computing** → **Psychology**; • **Human-centered computing** → **Virtual reality**;

KEYWORDS

eye movement classification, event detection, 360° content

ACM Reference Format:
Ioannis Agtzidis and Michael Dorr. 2019. Getting (More) Real: Bringing Eye Movement Classification to HMD Experiments with Equirectangular Stimuli. In *2019 Symposium on Eye Tracking Research and Applications (ETRA '19), June 25–28, 2019, Denver , CO, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3314111.3319829

ETRA '19, June 25–28, 2019, Denver , CO, USA

ACM ISBN 978-1-4503-6709-7/19/06…$15.00
https://doi.org/10.1145/3314111.3319829

1 INTRODUCTION

The eye tracking community has been constantly moving towards more immersive environments. Initially it started with static images, it then expanded towards moving dots and naturalistic videos, and in recent years it has begun to utilize wearable eye tracking and head mounted displays (HMDs) with integrated eye tracking due to their accessibility and affordability. For that reason new eye movement data sets for fixation detection [Steil et al. 2018] in wearable eye tracking and for saliency prediction in 360° images [Rai et al. 2017; Sitzmann et al. 2018] and 360° videos [David et al. 2018; Xu et al. 2018] are being published. In this paper, we are going to tackle the problem of eye movement segmentation in the equirectangular projection, which is used in the four saliency data sets and is generally the most common representation of 360° images and videos.

Segmentation of the raw gaze trace into a sequence of semantically meaningful labels describing its constituent eye movements ranks probably among the most fundamental tasks in eye movement research. For this purpose, many automatic algorithms have been developed to detect either a single eye movement [Agtzidis et al. 2016; Dorr et al. 2010; Salvucci and Goldberg 2000] or more than one eye movement at the same time [Komogortsev et al. 2010; Larsson et al. 2015; Startsev et al. 2018]. Some of these algorithms use simple criteria for the classification, other more complex handcrafted criteria, and lately many algorithms use learned parameters for classification [Hoppe and Bulling 2016; Startsev et al. 2018].

The caveat of all the aforementioned algorithms is that they cannot be applied directly to equirectangularly projected spaces because they were developed by assuming monitor-based experiments. This important assumption simplifies the overall algorithm design in several aspects, and we are going to explain some of these simplifications here. The video coordinate system and the monitor coordinate system coincide and they are usually placed at the top-left corner of the monitor, as can be seen in Figure 1. By having a single frame of reference we avoid difficult to understand conversions between frames of reference. The area of application is almost uniform with nearly the same pixels per degree density for the whole monitor even though there is a small deviation between its edges and its center. The last and most important factor that makes understanding and visualization easier is the fact that we work in the 2D Cartesian space.

As we move towards the equirectangular space things become more complicated. In the new space the video coordinate system and the experiment coordinate system differ. Now the participant lives inside a sphere surrounded by the video and as can be seen from Figures 2 and 3, the correspondence between the equirectangular coordinates and the experiment coordinates is not trivial. Also the

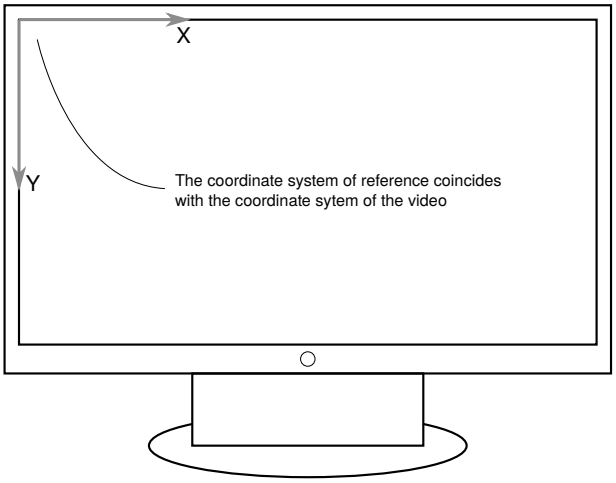

Figure 1: In monitor-based experiments the pixel density can be considered constant for the whole monitor area. The video and monitor coordinate systems coincide at the top left corner of the monitor and every performed calculation has to consider only a single frame of reference.

frame of reference of the experiment can be either fixed or moving together with the participant's head. In the first case the gaze trace visualizes head and eye movements and in the second case visualizes eye motion within the head which is equivalent to the eye's rotation within the eye socket. Moreover the 2D equirectangular plane does not have uniform pixel to sphere surface density as can be seen from Table 1 and Figure 5 where distortions start to appear as we move away from the horizontal midline. If we move to the 3D Cartesian space of Figure 4 the previous limitations are overcome but understanding and visualization of concepts becomes more difficult.

To show the importance of the conversion, imagine a hypothetical experiment where the objective is to measure fixation durations in an immersive scenario. For that purpose an HMD with integrated eye tracking is used for displaying diverse stimuli from drone captured videos. The participants' gaze will likely include positions far below the horizon since all of the interesting things happen in this area. However, the further the gaze is from the vertical center position, the more distorted is the equirectangular projection (see the gray areas in comparison to the black circles in Figure 5), and thus distances between gaze samples will be distorted as well. Now if the fixation duration analysis was performed with the dispersion-based algorithm from [Salvucci and Goldberg 2000], the duration would be overestimated for central fixations and underestimated for fixations at the top and bottom borders of the stimulus because the algorithm uses a single uniform dispersion threshold throughout the input space.

There are algorithms which try to incorporate and compensate head motion in the original 2D space. A recent algorithm for wearable eye-tracking glasses [Steil et al. 2018] defines fixations as foveation on the same target and thus indirectly compensates for head motion. The authors of the algorithm used the scene camera and a CNN [Zagoruyko and Komodakis 2015] to calculate the patch similarity around the gaze position in each frame and if the similarity value was above a threshold a fixation was detected. An earlier algorithm [Kinsman et al. 2012] compensates for head motion before fixation detection by extracting the motion information from the scene camera footage.

In this paper we extend five pre-existing algorithms to handle equirectangular recordings by extending their application space to the 3D Cartesian space and we make them publicly available through an open source license. We also provide a second method for handling equirectangular data by reprojecting the data to a new 2D space where monitor-based algorithms can be applied without any modification. Both the converted algorithms and the reprojection method can handle gaze traces composed of eye and head motion together and gaze traces of the eye within the head. Finally we evaluate our conversion method on the algorithms of [Larsson et al. 2015] and [Dorr et al. 2010] because together they are the most complete and elaborate package and because every other algorithm uses a subset of their criteria.

2 ALGORITHM CONVERSION

In our current analysis we use 360° equirectangular videos, which allows us to account only for rotations and no translations of the head as the participants watch the videos in an HMD. The data available to our algorithms for eye movement detection is the x,y equirectangular coordinates of the gaze and head direction as well as the head tilt. The gaze position is a composition of head and eye motion and the head tilt is equivalent to the roll principal axis. The combination of those 5 channels of information allows us to analyze both eye+head (overall gaze) behavior and eye within head behavior (field-of-view).

2.1 Equirectangular to Cartesian Space

Before moving to the actual algorithm conversion we have to understand the connection between the equirectangular frame of reference and the 3D Cartesian frame of reference. As already mentioned in the introduction, when we display the video in the HMD environment we live inside a sphere surrounded by the video. If we cut the sphere horizontally in the middle and look at it from the top we get Figure 2 with the Y axis pointing towards us. The x_{eq} equirectangular coordinate is the equivalent of angle ϕ in spherical coordinates and varies from 0° to 360°. Consequently, the 3D vector $(1, 0, 0)$ is the point where the left and right sides of the video meet; this is usually behind the participant at start time. The vector $(-1, 0, 0)$ represents the center of the video and the initial viewing position.

Now if we cut the previous figure along the Z - Y plane and look at it from the right we get the view of Figure 3 with the X axis pointing away from us. Here, the y_{eq} equirectangular coordinate is equivalent to the angle θ in spherical coordinates and varies from 0° to 180°; hence, we have a half circle.

With the definition of x_{eq} and y_{eq} coordinates in place we can now define the conversions between equirectangular, spherical, and 3D Cartesian coordinates in the following manner.

Equirectangular to spherical:

$$\begin{aligned} \phi &= x_{eq} * 2\pi / width_{eq} \\ \theta &= y_{eq} * \pi / height_{eq} \end{aligned} \tag{1}$$

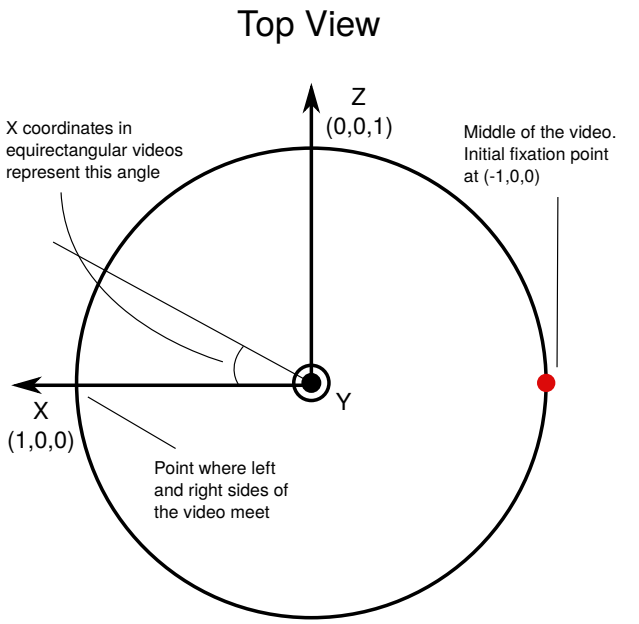

Figure 2: Top view of 3D Cartesian coordinate system of reference for the HMD experiment with the Y axis pointing towards us. The surrounding circle visualizes the intersection of the sphere of the 360-degree video with the X-Z plane. The 3D Cartesian X axis points to the left with the middle of the video represented by the red dot at $(-1, 0, 0)$. The x_{eq} equirectangular coordinate represents the angle of the projected unit vector on the X-Z plane with the X axis.

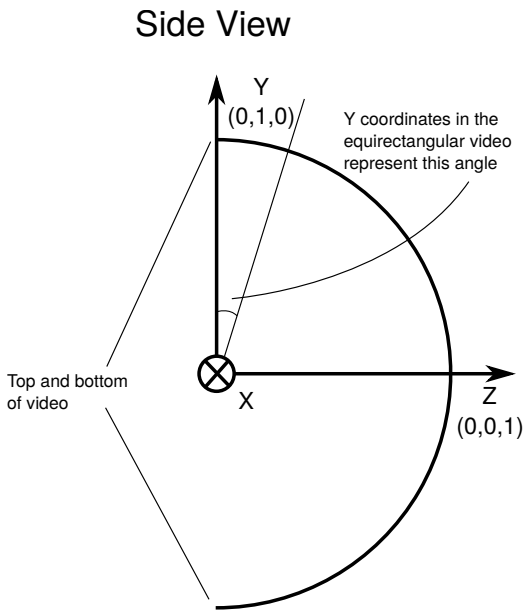

Figure 3: Side view of the 3D Cartesian coordinate system with the X axis pointing away from us. The y_{eq} coordinate represents the angle of a unit vector with the Y axis with the angle ranging from $[0, \pi]$ and is visualized through a half circle. The top and bottom of the equirectangular projection are at the $(0, 1, 0)$ and $(0, -1, 0)$ respectively.

Spherical to Cartesian:

$$\begin{aligned} x &= \sin\theta * cos\phi \\ y &= \cos\theta \\ z &= \sin\theta * \sin\phi \end{aligned} \tag{2}$$

2.2 Application to Existing Algorithms

For the actual algorithm conversion we can now use the 3D Cartesian unit vectors from equation 2 for calculating the angular distance, speed and dispersion which are the basic building blocks for all five algorithms that we want to convert. The angular distance and speed are calculated between two unit vectors in the same fashion as in [Diaz et al. 2013; Duchowski et al. 2002].

$$distance(\vec{v}_1, \vec{v}_2) = arccos(\vec{v}_1 \cdot \vec{v}_2) \tag{3}$$

$$speed(\vec{v}_1, \vec{v}_2) = \frac{distance(\vec{v}_1, \vec{v}_2)}{t_2 - t_1} \tag{4}$$

In the literature the dispersion typically is defined as a function of the size of the bounding box of the sample set defined on the video coordinate system [Larsson et al. 2015; Salvucci and Goldberg 2000]. In the 360° scenario, however, this definition loses its meaning since the two principal axes of the dispersion can be in any position and orientation on the sphere. Therefore we define the dispersion as the angle of the cone that starts from the center of the coordinate system and contains all the data points at its intersection with the sphere. This is visualized in Figure 4 and is defined below. A similar approach to ours is followed by the PUPIL headset in its fixation detector [Barz 2015].

$$\begin{gathered} dispersion(\mathbb{A}) = \{\underset{i,j}{argmax}\{distance(\vec{v}_i, \vec{v}_j)\} \mid i, j \in \mathbb{A}\} \\ \text{where } \mathbb{A} = \{1, 2, ..., n\} \end{gathered} \tag{5}$$

By using the equation 4 we can now directly convert the I-VT algorithm [Komogortsev et al. 2010] which uses a single threshold for velocity (or more specifically, its absolute magnitude, speed) to the 360° equirectangular video domain. The original algorithm labels saccades when the gaze speed is above threshold and fixations when it is below threshold. Because we have free head movement often the eyes will perform compensatory eye movements at intermediate speeds. Therefore in the 360° video domain the algorithm is only reliable for saccade detection and all the samples below the velocity threshold are left unassigned.

The saccade detector described in [Dorr et al. 2010] is more elaborate than I-VT and uses two speed thresholds. A saccade detection is initiated when the speed is above the high threshold and is then expanded forward and backwards in time until speed reaches the lower threshold. With this approach, it avoids erroneous detection of saccadic samples especially in 360° videos where the head can move freely and reach high angular speeds without performing a saccade. To convert this algorithm we again use the definition of speed from equation 4.

The I-DT fixation detector from [Salvucci and Goldberg 2000] calculates the dispersion of gaze samples in windows of fixed length. If the dispersion is below threshold the samples within the window are labelled as part of a fixation otherwise the window moves to the next gaze sample. Here, we use the dispersion definition of equation 5 for 360° videos as a replacement for the original definition.

Our second fixation detector is described in [Dorr et al. 2010] and uses a combination of dispersion and speed thresholds. Initially it starts from a fixed duration window and checks whether the dispersion and mean speed are below their respective thresholds. If both criteria hold true the window duration is extended forward in time and the dispersion threshold increases logarithmically in relation to the new window duration. The expansion phase stops when one of the conditions becomes false. At this point all the samples within the previously valid window are labelled as fixations. The conversion of this algorithm is straightforward and uses equations 4 and 5 for speed and dispersion calculation in the new domain.

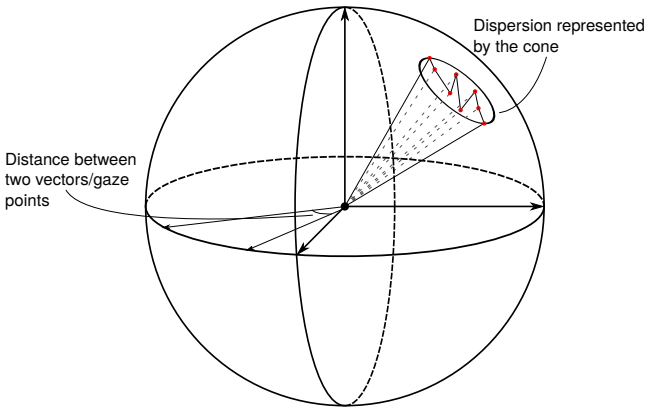

Figure 4: Visualization of the 3D Cartesian coordinate system within the HMD presentation sphere. The angle between two vectors represents the distance in the 3D space. The angle of the cone that contains a gaze segment represent the dispersion in the 3D space. The path length is the sum of distances between consecutive gaze points.

The algorithm described in [Larsson et al. 2015] works on windows in intersaccadic intervals and labels fixations and smooth pursuit (SP). The algorithm uses four criteria, namely dispersion, consistent direction, positional displacement and spatial range for the classification of eye movements. When none of the previous criteria are satisfied the window samples are labelled as fixations. Contrary to fixations, the window samples are labelled as SP when all the criteria are fulfilled. For the remaining cases where between one and three criteria are satisfied the algorithm looks at other labelled segments in the same intersaccadic interval to make its decision.

A publicly available implementation of this algorithm is provided by [Startsev et al. 2016] and includes the saccade detector from [Dorr et al. 2010] that we have already described above. For the conversion of this algorithm apart from using the equations 3 and 5 we have to define the sample-to-sample direction which is used in the preliminary segmentation described in section 2.2 of the original paper. The sample-to-sample direction within a window in the 3D Cartesian space is defined as

$$\vec{d}_i = \vec{v}_{i+1} - \vec{v}_i \quad \text{where } i = \{1, 2, \ldots, N-1\} \tag{6}$$

with the rest of the calculations for the Raleigh test staying the same.

In the evaluation of spatial features described in section 2.3 of the original paper, the parameter p_D represents the dispersion within each segment and is calculated as the ratio between the explained variance of the second and first principal components. In the 3D space we keep it as is. The parameter p_{CD} is a measure for the consistency of movement direction and is calculated as the ratio between d_{ED} and the length of the first principal axis, where d_{ED} is the distance of the first and last samples of the segment. In the 3D Cartesian space d_{ED} is the angular distance of the first and last vectors of an interval as defined in equation 3. In the 3D Cartesian space the length of the first principal axis is equivalent to the maximum dispersion of the segment and is taken from equation 5. The positional displacement parameter p_{PD} is the fraction of d_{ED} to d_{TL} with d_{TL} being the gaze path length. The gaze path length is calculated as the sum of consecutive vector distances within a window by using the equation 3. The parameter p_R is the spatial range which in the original algorithm is the length of the diagonal of the bounding box for all the sample of a window. In the 3D Cartesian space it is equivalent to the dispersion of the segment and is calculated through equation 5.

The last modification to the [Larsson et al. 2015] algorithm comes in section 2.4 for the classification of eye movements when between 1-3 criteria are satisfied and the criterion 3 is true. In this case we use the mean gaze direction between two segments which in the 3D Cartesian space is calculated as the angle, equation 3, between the mean direction vectors of each segment after vector normalization.

All the converted algorithm are available under GPL license in https://web.gin.g-node.org/ioannis.agtzidis/360_em_algorithms.

3 DATA CONVERSION

The approach to convert an algorithm to the 3D Cartesian space as described above may be either deemed too time consuming in comparison to the amount of data that we want to process or impossible due to closed source or very convoluted code. Because of these limitations we describe here a method of how to transform equirectangular gaze recordings to a new space where the original algorithms can be applied without any modification.

3.1 Cartesian to Equirectangular Space

Before presenting the reprojection process in the next section we first have to understand how we can move from the 3D Cartesian space back to equirectangular space. First we convert a normal vector $\vec{v} = (x, y, z)$ to its spherical representation with equation 7 which is the inverse of equation 2.

Table 1: Distortions of equirectangular projection increase exponentially the further away we move from the equator. Distortions are zero at the equator and infinite at the top and bottom of the projection.

Distance from equator	Distortion
0°	0.00
22.5°	0.08
45°	0.41
67.5°	1.61
90°	Inf

$$\begin{aligned} \phi &= arctan2(z, x) \\ \text{if}(\phi < 0): \ \phi &= \phi + 2\pi \\ \theta &= arccos(y) \end{aligned} \tag{7}$$

The two spherical angles are directly connected to the equirectangular coordinates and are calculated from equation 8 which is the inverse of equation 1.

$$\begin{aligned} x_{eq} &= width_{eq} * \phi/2\pi \\ y_{eq} &= height_{eq} * \theta/\pi \end{aligned} \tag{8}$$

3.2 Gaze Data Reprojection

As we have already mentioned the equirectangular projection for video and gaze representation comes with the drawback of higher distortions as we move away from the equatorial line (horizontal line passing through the middle of the video). We visualize these distortions by projecting the intersection of a cone with the sphere (Figure 4) to the equirectangular representation of Figure 5. The angle of the cone is 20° and it is centered at 0°, 22.5°, 45°, 67.5° and 90° away from the equatorial line. The gray area represents the equirectangularly projected cone with the black circle representing a hypothetical undistorted cone.

Table 1 summarizes the distortions at different distances from the equatorial line. The distortions represent how much longer a line segments appear in the equirectangular projection than in the HMD. For example if a horizontal line segment at 45 degrees from the equator covers 10 degrees of visual angle in the headset it would be represented by a projected line segment of 14.1. From the table we see that the equirectangular representation has no distortions in the middle and infinite distortions at the top and bottom. The infinite distortions arise from the fact that a single vector on the Y axis which covers zero degrees in the 3D Cartesian space acquires a whole line of pixels (360°) in the equirectangular representation. From this observation, it becomes obvious that pre-existing algorithms would fail close to the top and bottom areas of Figure 5. Intuitively these algorithms should not experience significant detection quality deterioration in the areas close to the equator.

If we do not want to change the algorithms as we did in the previous section we can take advantage of the central area of 45° vertically which has low distortions that do not exceed 8%. Therefore if all the gaze samples for a given period of time are exclusively

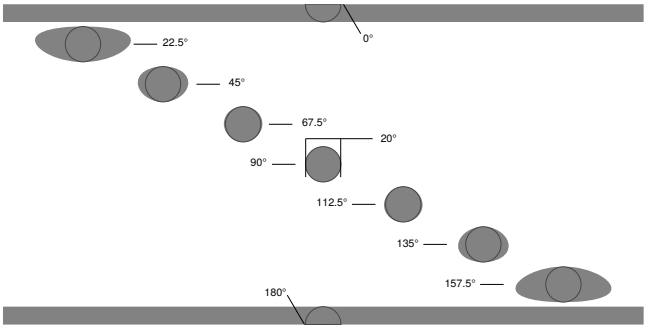

Figure 5: Visualization of distortions at the points of Table 1. The gray areas are the projection of the intersection of a cone with the presentation sphere. The black circles visualize a hypothetical undistorted circle.

within this horizontal stripe we can then use all the pre-existing algorithms in their original forms. We can achieve those preconditions by splitting the gaze input into time periods of no more than 45° vertical spread which can then be moved to the central part.

During the centering of gaze samples we have to be careful not to simply displace vertically the gaze coordinates in the equirectangular representation because we will keep all the distortions, but to perform a rotation in the 3D Cartesian space. Another point of attention should be the limitation of monitor-based algorithms to handle movement along the vertical edges of the equirectangular video which are seamlessly presented during the HMD viewing. These transition cannot be efficiently processed and therefore we choose to move samples not only vertically but also horizontally to the middle of the video. By doing so the possibility of the gaze moving along the vertical edges of the equirectangular frame decreases dramatically.

For the rotation matrix computation we use the extrinsic Y-Z-X Tait-Bryan angles of the mean gaze vector to the coordinate system of reference. The Y rotation is computed with the right-hand rule from the projected vector on the X-Z plane of Figure 2 and is equivalent to ϕ in equation 7. Then the Z rotation is computed in a similar fashion to the angle of Figure 3 but with the X and Z axes exchanged and this is equivalent to θ in equation 7. The rotation around the X axis is provided as head tilt in the recorded data. Then we find the rotation matrix between the middle of the equirectangular video which is represented by the $(-1, 0, 0)$ vector to the coordinate system of reference in the same fashion. The combination of these two rotation matrices gives us the overall rotation from the mean gaze vector to the center of the video. In the last step we apply the already calculated rotation matrix to all of the gaze samples and project them back to the equirectangular space by using equation 7, 8. Now the transformed data is ready to be handled by any algorithm that was developed for monitor-based experiments.

4 EYE WITHIN HEAD GAZE

In the eye within head analysis we want to disentangle head motion from eye motion. We do this by using the x,y equirectangular head coordinates along with the head tilt in order to find the rotation

Table 2: Percentage of samples for the four areas representing different distortion levels.

Range	% Samples
{67.5°, 112.5°}	84.92
{45°, 67.5°} ∪ {112.5°,135°}	10.63
{22.5°, 45°} ∪ {135°, 157.5°}	3.51
{0°, 22.5°} ∪ {157.5°, 180°}	0.95

matrix to a fixed reference vector in the same way as in the previous section. We then apply this rotation to the corresponding gaze vector. The resulting vector corresponds to the eye motion within the head since the rotated head vector always coincides with the fixed reference vector.

After the eye-head disentanglement we can use the converted algorithms directly on the transformed vectors. For the data projection approach we calculate the $\phi_{gaze}, \theta_{gaze}$ and ϕ_{ref}, θ_{ref} from equation 7 for the gaze and the reference constant vector. We can then calculate the field-of-view angle as

$$\phi_{fov} = \phi_{gaze} - \phi_{ref} + fov\_width_{rads}/2$$
$$\theta_{fov} = \theta_{gaze} - \theta_{ref} + fov\_height_{rads}/2$$

with the top left corner of the fov corresponding to the $(0, 0)$ position. With the assumption that the fov is a 2D linear space we can calculate the corresponding x,y coordinates as

$$x_{fov} = fov\_width_{px} * \phi_{fov}/fov\_width_{rads}$$
$$y_{fov} = fov\_height_{px} * \theta_{fov}/fov\_height_{rads} \quad (9)$$

5 EVALUATION

For the evaluation we used the first 8 participants' data from [Agtzidis et al. 2019]. Gaze data were gathered with the FOVE[1] HMD headset while the participants were watching 360° videos. The used headset provides integrated eye tracking and can play back 360° equirectangular videos through the integrated media player of SteamVR[2]. Segments of 14 Youtube videos were used as stimulus, licensed under Creative Commons[3], with a final duration of about one minute each. With an additional synthetic clip comprising of a moving dot, we overall had at our disposal about 120 minutes of gaze data. At this scale, it was not feasible to obtain manual ground truth annotations, and therefore the optimization and thorough evaluation of the converted algorithms was not possible. However, this was not a major limitation since we want to assess how the algorithms fare with different levels of distortion. For that reason, we evaluated the original algorithm, its converted version, and the original algorithm with converted input data in the eye+head scenario (i.e. overall gaze). During testing we split the input space in four pairs of distinct areas, which represent horizontal stripes in the equirectangular video of 22.5° height. These pairs were placed symmetrically on both sides of the equator line, and depending on its distance from this line, each pair represented a different level of distortion.

[1]https://www.getfove.com/
[2]https://steamcommunity.com/games/250820/announcements/detail/3021212865824077306
[3]https://creativecommons.org/licenses/by/3.0/legalcode

We evaluated the algorithm proposed by [Larsson et al. 2015] (with the saccade detector by [Dorr et al. 2010]) and implemented by [Startsev et al. 2016] because it is the most complete package by providing labels for fixations, saccades, SP and noise which includes blinks. Moreover the initial algorithm can label ARFF files directly and because of this we do not have to change it at all. For a full evaluation of the original algorithms, see [Startsev et al. 2018]. As an evaluation metric we use the Cohen's Kappa as defined in equation 10. Each of the four eye movements is evaluated separately in a binary classification problem with positive labels for the detection of the given eye movement. P_o represents the proportion of agreement between the two versions of the algorithm and P_c the proportion of chance agreement. Kappa ranges from $[-1, 1]$ with higher scores representing higher agreement.

$$K = \frac{P_o - P_c}{1 - P_c} \quad (10)$$

For the evaluation of the two versions of the algorithm we kept all the parameters (16 in total) the same but η_{maxFix}. The last parameter represents the dispersion of the interval which is measured as the diagonal of the bounding box containing the interval samples in the original algorithm. In the converted algorithm the dispersion is calculated from equation 5 as the angle of a cone. The maximum difference between these two definitions of dispersion is achieved for a square bounding box where its diagonal is $\sqrt{2}$ times bigger than the radius of the circle that is tangent to all of its sides. Therefore we used an η_{maxFix} of 2.7° for the original version and 1.9° for the converted version.

Table 2 reports the percentage of time spent in each of the four areas, with Table 3 reporting the Kappa values for agreement between three different combinations of algorithms and data for each eye movement type. These combinations use the original version of the algorithm with the original data, the converted version of the algorithm with the original data, and the original algorithm on converted data.

When we compare the two versions of the algorithm applied on the original data, we see that the highest agreement is achieved in the middle part of the video that spans the 67.5° to 112.5° range and holds roughly 85% of all samples. In this area the Kappa scores are 0.82 for fixations, 0.79 for saccades and 0.73 for smooth pursuit. Even though these scores are high, we do not reach 1 which represents perfect agreement. The not *perfect* agreement may be attributed in part to the small distortions that appear at the borders of the middle area. More importantly, though, there is also no perfect agreement between the initial and converted criteria (16 in total) such as the slightly different definitions of dispersion. These two factors become more obvious for the SP detection, which is more sensitive to disturbances because often its characteristics (such as speed and dispersion) overlap with those of fixations and saccades. For now we will not comment on noise and leave its explanation for later.

The second area, which spans from 45° to 67.5° and from 112.5° to 135° from the top of the video, accounts for roughly 11% of the total samples. In this area distortions range from 8% to 45%, which is not enough to influence the overall detection quality for fixations and saccades since their Kappa scores remain static. The flat Kappa scores are probably the result of the clear separation for

Table 3: Cohen's kappa scores for four different areas with varying distortion levels. Evaluating the original algorithm against the converted algorithm against the original algorithm with converted data.

	Cohen's kappa between **original and converted algorithms**			
Range	Fix.	Sacc.	SP	Noise
{67.5°, 112.5°}	0.816	0.794	0.729	0.452
{45°, 67.5°} ∪ {112.5°,135°}	0.804	0.794	0.667	0.551
{22.5°, 45°} ∪ {135°, 157.5°}	0.733	0.759	0.556	0.497
{0°, 22.5°} ∪ {157.5°, 180°}	0.349	0.435	0.091	0.405

	Cohen's kappa between **original and converted data**			
Range	Fix.	Sacc.	SP	Noise
{67.5°, 112.5°}	0.973	0.973	0.952	0.931
{45°, 67.5°} ∪ {112.5°,135°}	0.915	0.936	0.840	0.881
{22.5°, 45°} ∪ {135°, 157.5°}	0.805	0.871	0.646	0.774
{0°, 22.5°} ∪ {157.5°, 180°}	0.345	0.466	0.043	0.462

	Cohen's kappa between **converted algorithm and converted data**			
Range	Fix.	Sacc.	SP	Noise
{67.5°, 112.5°}	0.812	0.777	0.714	0.369
{45°, 67.5°} ∪ {112.5°,135°}	0.810	0.777	0.656	0.448
{22.5°, 45°} ∪ {135°, 157.5°}	0.826	0.792	0.674	0.439
{0°, 22.5°} ∪ {157.5°, 180°}	0.844	0.788	0.693	0.394

the characteristics of fixations and saccades, but we see a substantial drop in the agreement for SP.

In the third area, which spans from 22.5° to 45° and from 135° to 157.5°, we have roughly 4% of the total gaze sample. Here we have high distortions that range from 41% to 161% and therefore the uncorrected artifacts of the original algorithms lead to strongly reduced agreement between algorithms for all three detected eye movement types.

In the last area, which covers the top and the bottom areas of the equirectangular projection, its extreme distortions result in a significant drop in the agreement for fixations and saccades with almost no agreement (at chance level) for SP detection. However, this area holds only a small share of the overall samples (1%).

Now when we look the results of applying the original algorithm on the original and converted data we see that the agreement is almost perfect for the middle area. Still the small deviation can be attributed to the small distortions at the border of this area and the temporal border effects that are introduced by the segment-wise data conversion. As expected, the higher distortions for the off-center areas in the original data lead to roughly the same disagreement scores at the top and bottom, meaning that the approach of converting the data also reduces artifacts.

When we compare the two proposed distortion-aware methods (bottom panel of Table 3) we see that the agreement between them stays roughly the same throughout the equirectangular projection. As before, the Kappa scores do not reach perfect agreement due to subtle changes in algorithm criteria, but they are very close to the results of the middle area (which has the least distortions) between the original algorithm with original data and the converted algorithm. The constant Kappa scores further prove the correspondence between our two proposed conversion methods.

Noise Kappa scores are more erratic between the three different comparison cases. For the results between the original and converted data the noise scores fall in line with the other eye movements and monotonically decrease as we move away from the middle line. For the other two cases, the noise Kappa score hovers around 0.4 for all four areas. The almost flat noise scores may be attributed to the blink detector that is included in the implementation of [Startsev et al. 2016]. The blink detection algorithm first detects saccades by using the algorithm of [Dorr et al. 2010] (explained in Section 2.2). It then searches on both sides of an area with eye tracking loss and if it detects a saccade close in time it labels all the samples (including the saccade) as noise. Also before and after a blink the video-oculography based eye trackers tend to report a shift in gaze. In the case of the initial algorithm the probability of finding a saccade is higher because it does not compensate for the high distortions close to the top and bottom that result in erroneously detecting higher speeds and therefore more saccades.

To summarize the results, we have (i) compared the initial and converted versions of the saccades detection algorithm from [Dorr et al. 2010] and the fixation/SP detection algorithm from [Larsson et al. 2015]; and (ii) compared the initial version of the algorithms using the original and converted data. We have shown that the highest agreement between the original and converted versions occurs in the middle (least distorted) part of the equirectangular projection with high Kappa scores. As we move further away from the middle region the agreement for all three eye movements drops with SP being the most influenced. Especially in the areas with the highest distortions at the top and bottom of the equirectangular projection the eye movement classification becomes completely unreliable with the initial algorithm. As expected the performance between the converted algorithms and converted data is not influenced by the distortions and stays constant throughout the whole area of the equirectangular projection. To conclude, the high Kappa scores for the middle area confirm experimentally the plausibility of both the algorithm conversion and the gaze reprojection methods.

6 CONCLUSION

In this paper we presented two methods of how to segment gaze recordings into their constituent eye movements when the experiment uses HMDs with equirectangular videos as input. In the first method we converted 5 popular eye movement classification algorithms in order to work in the 3D Cartesian space. These algorithms are not influenced by the distortions, but the initial conversion incurs some complexity. In the second method we transformed the gaze data in order to map the gaze to areas without distortions. We were then able to apply the original algorithms unchanged but at the cost of segmenting the input gaze trace into many smaller parts, especially when the gaze frequently moves along the vertical axis. Our methods were empirically verified by the evaluation results presented in Table 3: taking into account the equirectangular

distortions changed the results of the eye movement classification, depending on vertical gaze position even dramatically. This reduction in artifacts shows that standard eye movement classification algorithms should not be used for 360 degree stimulus material without modifications.

Both the algorithm conversion and data reprojection can be used to analyze the overall gaze, which comprises of eye together with head motion as well as the eye motion only which is the motion of the eye within the eye socket. This distinction is important because we can get statistics independently for each view, but most importantly we can combine the two views in order to detect more types of eye movements. For example, if we detect a fixation in the overall gaze and an SP in the eye only view, we can say that the overall eye movement is a fixation with vestibulo-ocular reflex compensating for head motion. This two-view functionality is important because most of the wearable eye trackers report eye only gaze traces and most of the remote eye trackers report eye together with head gaze traces. Therefore, depending on the kind of the reported gaze we have to use the correct version of the algorithm with tweaked thresholds.

Even though our current analysis was limited to HMD experiments where the head pose is always known, the converted algorithms can still be used with wearable eye trackers that include gyroscopes such as the TobiiPro Glasses 2[4]. This approach was implemented to I-VT detector of TobiiPro [Olsen 2012] by [Hossain and Miléus 2016]. The head motion alternatively could be approximated from the motion information of the scene camera of the headset as in [Kinsman et al. 2012]. Neither method is perfect because it can accumulate errors through measurement drift, abrupt or fast head motion, and noisy recordings. Even with these imperfect head estimation methods we can have an overall gaze trace which will point to the wrong direction but its speed, dispersion and all other kinds of local characteristics will be valid for eye movement classification.

Finally, the segmentation of a gaze trace into eye movements in virtual reality (VR) can prove important in many more scenarios such as VR therapy [Lutz et al. 2017], gaze-based wheelchair navigation [Eid et al. 2016; Ktena et al. 2015], and as input modality for human-machine interaction in immersive environments [Choe et al. 2018; Sidorakis et al. 2015]. In some of these cases the work presented in this paper has to be expanded in order to account for head translations and/or ego motion and the subsequent parallax effect.

ACKNOWLEDGMENTS

This research was supported by the Elite Network Bavaria, funded by the Bavarian State Ministry of Science and the Arts.

REFERENCES

Ioannis Agtzidis, Mikhail Startsev, and Michael Dorr. 2016. Smooth pursuit detection based on multiple observers. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications (ETRA '16)*. ACM, New York, NY, USA, 303–306.

Ioannis Agtzidis, Mikhail Startsev, and Michael Dorr. 2019. A Ground-Truth Data Set and a Classification Algorithm for Eye Movements in 360-degree Videos. *arXiv preprint arXiv:1903.06474* (2019).

Michael Barz. 2015. PUPIL fixation detection. https://github.com/pupil-labs/pupil/blob/master/pupil_src/shared_modules/fixation_detector.py. (2015).

Mungyeong Choe, Yeongcheol Choi, Jaehyun Park, and Hyun K Kim. 2018. Comparison of Gaze Cursor Input Methods for Virtual Reality Devices. *International Journal of Human–Computer Interaction* (2018), 1–10.

Erwan J David, Jesús Gutiérrez, Antoine Coutrot, Matthieu Perreira Da Silva, and Patrick Le Callet. 2018. A dataset of head and eye movements for 360° videos. In *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM, 432–437.

Gabriel Diaz, Joseph Cooper, Dmitry Kit, and Mary Hayhoe. 2013. Real-time recording and classification of eye movements in an immersive virtual environment. *Journal of Vision* 13, 12 (2013), 5–5.

Michael Dorr, Thomas Martinetz, Karl R Gegenfurtner, and Erhardt Barth. 2010. Variability of eye movements when viewing dynamic natural scenes. *Journal of Vision* 10, 10 (2010), 28–28.

Andrew T Duchowski, Eric Medlin, Nathan Cournia, Anand Gramopadhye, Brian Melloy, and Santosh Nair. 2002. 3D eye movement analysis for VR visual inspection training. In *Proceedings of the 2002 symposium on Eye tracking research & applications*. ACM, 103–110.

Mohamad A Eid, Nikolas Giakoumidis, and Abdulmotaleb El-Saddik. 2016. A Novel Eye-Gaze-Controlled Wheelchair System for Navigating Unknown Environments: Case Study With a Person With ALS. *IEEE Access* 4 (2016), 558–573.

Sabrina Hoppe and Andreas Bulling. 2016. End-to-End Eye Movement Detection Using Convolutional Neural Networks. *ArXiv e-prints* (Sept. 2016). arXiv:cs.CV/1609.02452

Akdas Hossain and Emma Miléus. 2016. Eye Movement Event Detection for Wearable Eye Trackers. (2016). Linköpings Universitet thesis LiTH-MAT-EX–2016/02–SE.

Thomas Kinsman, Karen Evans, Glenn Sweeney, Tommy Keane, and Jeff Pelz. 2012. Ego-motion compensation improves fixation detection in wearable eye tracking. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 221–224.

Oleg V Komogortsev, Denise V Gobert, Sampath Jayarathna, Do Hyong Koh, and Sandeep M Gowda. 2010. Standardization of automated analyses of oculomotor fixation and saccadic behaviors. *IEEE Transactions on Biomedical Engineering* 57, 11 (2010), 2635–2645.

S. I. Ktena, W. Abbott, and A. A. Faisal. 2015. A virtual reality platform for safe evaluation and training of natural gaze-based wheelchair driving. In *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*. 236–239. https://doi.org/10.1109/NER.2015.7146603

Linnéa Larsson, Marcus Nyström, Richard Andersson, and Martin Stridh. 2015. Detection of fixations and smooth pursuit movements in high-speed eye-tracking data. *Biomedical signal processing and control* 18 (2015), 145–152. http://dx.doi.org/10.1016/j.bspc.2014.12.008

Otto Hans-Martin Lutz, Charlotte Burmeister, Luara Ferreira dos Santos, Nadine Morkisch, Christian Dohle, and Jörg Krüger. 2017. Application of head-mounted devices with eye-tracking in virtual reality therapy. *Current Directions in Biomedical Engineering* 3, 1 (2017), 53–56.

Anneli Olsen. 2012. The Tobii I-VT fixation filter. *Tobii Technology* (2012).

Yashas Rai, Jesús Gutiérrez, and Patrick Le Callet. 2017. A dataset of head and eye movements for 360 degree images. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 205–210.

Dario D Salvucci and Joseph H Goldberg. 2000. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 symposium on Eye tracking research & applications*. ACM, 71–78.

Nikolaos Sidorakis, George Alex Koulieris, and Katerina Mania. 2015. Binocular eye-tracking for the control of a 3D immersive multimedia user interface. In *Everyday Virtual Reality (WEVR), 2015 IEEE 1st Workshop on*. IEEE, 15–18.

Vincent Sitzmann, Ana Serrano, Amy Pavel, Maneesh Agrawala, Diego Gutierrez, Belen Masia, and Gordon Wetzstein. 2018. Saliency in VR: How do people explore virtual environments? *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (2018), 1633–1642.

Mikhail Startsev, Ioannis Agtzidis, and Michael Dorr. 2016. Smooth Pursuit. http://michaeldorr.de/smoothpursuit/. (2016).

Mikhail Startsev, Ioannis Agtzidis, and Michael Dorr. 2018. 1D CNN with BLSTM for automated classification of fixations, saccades, and smooth pursuits. *Behavior Research Methods* (2018), 1–17.

Julian Steil, Michael Xuelin Huang, and Andreas Bulling. 2018. Fixation detection for head-mounted eye tracking based on visual similarity of gaze targets. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. ACM, 23.

Yanyu Xu, Yanbing Dong, Junru Wu, Zhengzhong Sun, Zhiru Shi, Jingyi Yu, and Shenghua Gao. 2018. Gaze Prediction in Dynamic 360° Immersive Videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5333–5342.

Sergey Zagoruyko and Nikos Komodakis. 2015. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4353–4361.

[4]https://www.tobiipro.com/product-listing/tobii-pro-glasses-2/