Link to data:
https://atreus.informatik.uni-tuebingen.de/seafile/d/8e2ab8c3fdd444e1a135/?p=%2F&mode=list

Histogram of oriented velocities for eye movement detection

Wolfgang Fuhl
University of Tübingen, Perception Engineering
Tübingen, Germany
wolfgang.fuhl@uni-tuebingen.de

Nora Castner
University of Tübingen, Perception Engineering
Tübingen, Germany
castnern@informatik.uni-tuebingen.de

Enkelejda Kasneci
University of Tübingen, Perception Engineering
Tübingen, Germany
enkelejda.kasneci@uni-tuebingen.de

ABSTRACT

Research in various fields including psychology, cognition, and medical science deal with eye tracking data to extract information about the intention and cognitive state of a subject. For the extraction of this information, the detection of eye movement types is an important task. Modern eye tracking data is noisy and most of the state-of-the-art algorithms are not developed for all types of eye movements since they are still under research. We propose a novel feature for eye movement detection, which is called histogram of oriented velocities. The construction of the feature is similar to the well known histogram of oriented gradients from computer vision. Since the detector is trained using machine learning, it can always be extended to new eye movement types. We evaluate our feature against the state-of-the-art on publicly available data. The evaluation includes different machine learning approaches such as support vector machines, regression trees, and k nearest neighbors. We evaluate our feature together with the machine learning approaches for different parameter sets. We provide a matlab script for the computation and evaluation as well as an integration in EyeTrace which can be downloaded at *http://www.ti.uni-tuebingen.de/Eyetrace.1751.0.html.*

CCS CONCEPTS

• **Computing methodologies → Supervised learning**; **Classification and regression trees**; **Support vector machines**; *Feature selection*; *Cross-validation*;

KEYWORDS

Eye tracking, eye movements, machine learning, saccade, fixation, smooth pursuit, post saccadic movements, support vector machine, regression trees, k nearest neighbours

ACM Reference Format:
Wolfgang Fuhl, Nora Castner, and Enkelejda Kasneci. 2018. Histogram of oriented velocities for eye movement detection. In *Workshop on Modeling Cognitive Processes from Multimodal Data (MCPMD'18), October 16, 2018, Boulder, CO, USA,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.1145/3279810.3279843

MCPMD'18 , October 16, 2018, Boulder, CO, USA

ACM ISBN 978-1-4503-6072-2/18/10…$15.00
https://doi.org/10.1145/3279810.3279843

1 INTRODUCTION

Modern research, as well as industry, is more concerned with eye tracking; for both, the extraction of information about the subject is interesting. Specifically, the cognitive state, which also provides information about attention [3, 17]. It helps to assess whether a person is currently able to perform a task [21] and particularly important for autonomous driving [12], as the driver should always be prepared to take over the vehicle. In science, new ways of human-computer interaction and computer-controlled analyses for the detection of diseases employ eye movement information. A user is not aware of the midas touch problem. However, eye tracking as input for the computer become a overload of constant signals, making meaningful interaction impossible. In the field of medicine, eye movements are used to measure and reliably detect defects in movement and perception. Some examples are crossed eyes (strabismus) [20], autism [2], visual field defects (glaucoma) [4, 14], etc.

Initially in eye tracking, extraction is the detection of eye movements based on the raw data. The types are generally determined on the basis of speed and duration [6, 11]. If the subject focuses an image area without moving his eyes, this is called fixation [6, 11]. Therefore, a low speed and the minimum duration the subject needs to perceive the information would be relevant for detection. If the subject moves his eye from one interesting place of the stimulus to another, this is called saccade [6, 11]. It is a fast eye movement where the subject has no visual perception. Meaning, the target of the eye movement can be an overshot, making a compensating movement necessary: Known as the post-saccadic movement [5, 22]. Aside from saccades and fixations, there are also other eye movement types. For instance, smooth pursuits [11], where the subject follows the moving stimulus with his eyes.

In [1] the authors criticize that the current state-of-the-art does not work satisfactorily for real data. They write that the detection of fixations and saccades works reliably only after laborious parameter adjustments. For novel eye movement types, such as smooth pursuits or post saccadic movement, all approaches fail in real data [1]. A recent study by [30] analyzed the applicability of machine learning to the eye movements detection. They found that it outperforms the state-of-the-art for a pointwise classification. Time series where each sample had to be classified without seeing data from the same subject was not evaluated. The feature vector used was based on velocity and statistics of a small window surrounding the inspected position in the data.

The main problems in extracting the eye movements stem from eye tracker device variability, such as different sampling rates and the noise in the eye-tracking data itself. Most algorithms automatically adapt the parameters to the frame rate, but from [1], they do

not perform satisfactorily. Therefore, the researcher has to adapt the parameters, which becomes laborious. Machine learning alone does not improve this situation, since it needs annotated data (generally done manually), but it can automatically adapt to new data if annotations some samples are given. Together with a data generator as proposed in [9], the amount of data necessary to be manually annotated is effectively reduced.

Another issue with state-of-the-art approaches is the discovery of novel eye movement types. All algorithms are designed for specific types and cannot easily be extended to others. Generally, machine learning approaches need examples of the new type for the training in order to automatically adapt to it. Therefore, the usage of such a machine learning based approach is preferable [30].

We propose a novel feature which computes an orientation histogram where the velocities are summed. This feature is derived from the well-known histogram of oriented gradients [31] from computer vision. The feature can be computed based on only previous information which makes it online applicable. For offline analysis, it can also compute using a window surrounding the inspected data point.

2 RELATED WORK

This section covers the state-of-the-art approaches. We categorized them into three main categories. The first is the classical fixed threshold approaches. In the second category algorithms, which adapt their threshold based on the data are presented. Then, the machine learning approaches are explained that were already applied to the detection of eye movement types.

2.1 Threshold based approaches

The classical approach to detect fixations and saccades is based on the different velocities. The algorithm Identification by Velocity Threshold (IVT) [23] uses one fixed threshold, where values below are classified as a fixation and above as a saccade. This algorithm is improved by using the time dimension. The aptly named Identification by Dispersion-Threshold (IDT) [23] algorithm uses the data reduction proposed in [29] and an additional threshold for the minimum fixation duration. Currently, the IVT is still used for high-speed recordings, but generally does not work for noisy data.

A solution for noisy data was published with Identification by Kalman Filter (IKF) algorithm [16]. The algorithm uses the Kalman Filter to predict the next velocity and applies the same thresholds as IDT on the generated signal: Meaning the data is smoothed online. An alternative for the Kalman Filter was presented in [15] where they used the χ^2-test instead.

There exist also two clustering based approaches the F-tests Dispersion Algorithm (FDT) [27] and the Covariance Dispersion Algorithm (CDT) [28]. The idea behind those approaches is to test two samples if they belong to the same type base on their surrounding statistics. The main disadvantage of the F-Test, which is used in the FDT algorithm, is that it is affected by noise. Therefore, CDT uses the covariance matrix, which is robust against noise. Both algorithms need three parameters. The minimum duration of a fixation, the variance, and either the F-Score or the covariance respectively.

2.2 Adaptive threshold-based approaches

The first adaptive approach was based on IVT. It was proposed in [7, 8] to detect microsaccades. The velocity threshold was computed based on the noise level in the data. In [15], the algorithm Identification by a Minimal Spanning Tree (IMST) was proposed. It computes a tree structure over the data samples, where the velocities are at the leaves. Then, each parent of multiple leaves is combined with the goal to find the tree with a minimum of branches.

For post saccadic movement, the first algorithm was proposed in [22]. Similar to [7, 8], the velocity threshold is adapted based on the noise in the data. An approach using the information of both eyes was published with Binocular-Individual Threshold (BIT) [26]. This algorithm also adapts its thresholds automatically, but the novel enhancement was that both eyes have to perform the same eye movement type. Therefore, it uses both eyes to check the detected types.

The first algorithm detecting all eye movement types (e.g. fixations, smooth pursuits, saccades, and post saccadic movement) was published in [19]. It adapts the parameters automatically using the noise of the data. Another algorithm specially designed for high-speed eye trackers capable of detecting all types of eye movements was proposed in [18]. The algorithm uses a combination of automatically adapting thresholds and clustering to segment the data into eye movement types. After the initial segmentation, a refinement iteration is executed that evaluates each segment again.

2.3 Based on machine learning

The first machine learning based approach that was used to detect eye movement types are Hidden Markov Models (HMM) [13, 15, 24, 25]. The idea behind this approach is to compute at least two probability distributions on the data and select the type based on the probability. It can be seen as two overlapping Gaussian distributions. In [24], the approach was extended to detect smooth pursuits by adding a third probability distribution. Support vector machines, decision trees, and k nearest neighbors were used in [30] to classify different eye movement types. The computed feature vector were statistics surrounding the inspected data sample. In comparison, we propose a novel feature which incorporates the velocity and orientation of movement, which can be classified with the same machine learning approach as proposed in [30]. Our feature can be used in online and offline classification and will be described in more detail in the following section.

3 METHOD

Figure 1 shows the computation of the oriented velocity histograms based on three windows. For the first window (Window 1 in figure 1), the velocity and angle between the inspected point and each preliminary point are computed. The angle is the bin in the histogram, and the absolute value of the velocity is added to this bin. This part of the feature is usable online. In window two, similar computations as window one are performed, but with raw data points sampled after the inspected point. For window three, we compute the angle and velocity between raw data points having the same time distance from the inspected data point. The computation for all three windows is described in algorithm 1. H is the computed histogram for the inspected $Position$. Based on the $WindowType$,

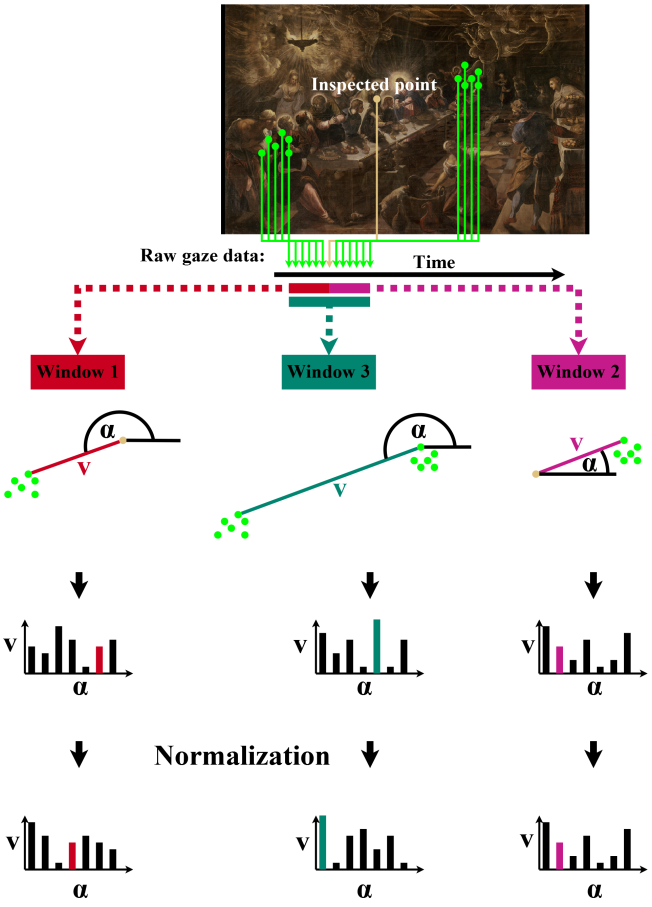

Figure 1: Computation and normalization of oriented velocity histograms based on raw data.

the correct data indexes are computed in the given raw *Data* and the specified *Window* size. After the computation of the histogram, it has to be normalized. The normalization makes it rotation invariant and reduces the numerical impact of large numbers. Therefore, the histogram is rotated so that the highest velocity is at the first bin (zero degree). Afterwards, the histogram is divided by its sum ($H = \frac{H}{\sum H}$). A distribution is formed and each velocity is now a relation to the others in the histogram. The rotation and division by the sum are commutative, which is shown in figure 2. After the computation and normalization of the histogram, the vector for the machine learning approaches is constructed by concatenating all three. This construction gives one vector for each data point, where the window size for the histogram computation is our first parameter (pf). For the classification to be more robust and further incorporate the surrounding information, features from preliminary and successive data points are combined. This is the second window and therefore a second parameter of the feature (pw). The third and last parameter of our proposed feature is the approximation of angles (pa); where the amount of bins in the histogram is reduced. Therefore, each bin represents a range of degrees.

Algorithm 1 Calculate histogram H

Require: $Data, Window, Position, WindowType$
  $H(\forall\alpha) = 0$
  for $i = 1 : Window$ **do**
    if $WindowType == 1$ **then**
      $\alpha = atan2(Data(Position) - Data(Position - i))$
      $v = |Data(Position) - Data(Position - i)|$
    end if
    if $WindowType == 2$ **then**
      $\alpha = atan2(Data(Position) - Data(Position + i))$
      $v = |Data(Position) - Data(Position + i)|$
    end if
    if $WindowType == 3$ **then**
      $\alpha = atan2(Data(Position - i) - Data(Position + i))$
      $v = |Data(Position - i) - Data(Position + i)|$
    end if
    if $\alpha < 0$ **then**
      $\alpha += 360°$
    end if
    $H(\alpha) += v$
  end for

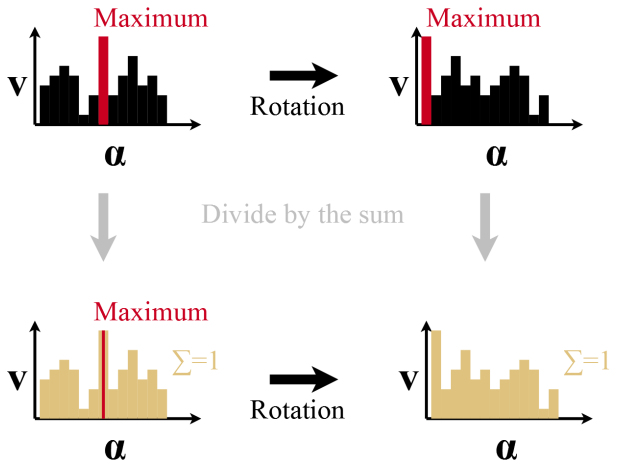

Figure 2: Normalization of a histogram.

4 EVALUATION

For comparison, we used the state-of-the-art algorithms [24] (IBDT), [22] (EV), [10] (I2MC), and [19] (LS). The evaluation was performed on the publicly available data set [24] (IBDT-DATA) without modification. Meaning, each algorithm received the entire data with errors and unannotated data samples. For the statistical evaluation, the unannotated data samples were not taken into account. Our proposed approach was cross-validated, which means that we removed the data from one subject (four files in the data set) for the training data. Afterward, the files from the excluded subject are used for classification and evaluation. This cross-validation was done for each subject was done for each subject. For the proposed feature, we used the support vector machine, k nearest neighbors,

and regression trees with different parameters. Our proposed feature was also evaluated with different parameters to evaluate the robustness.

All state-of-the-art approaches are configured for offline detection. In case of IBDT, this means that the unjitter function was enabled. Therefore, we used our feature for offline detection, which means that we used all three windows for the feature computations. The combination of feature vectors also included vectors from successive data points.

Named	**Configuration**
knn5-20	*k=5,10,15, or 20 (neighbors for comparison).*
tree1	*Maximum splits 50.*
tree2	*Maximum splits 50, Predictor selection with curvature. Exact categorization.*
tree3	*Maximum splits 50, Predictor selection with curvature, Exact categorization, split criterion deviance.*
tree4	*Maximum splits 50, Exact categorization*
tree5	*Maximum splits 50, Exact categorization, split criterion deviance.*
svm-lin	*Linear kernel function.*
svm-pol	*Second order polyniomal as kernel function.*

Table 1: Different configurations of the machine learning approaches used in the evaluation.

In table 1, all used configurations for the machine learning approaches are shown. With knn, we refer to k nearest neighbors, with svm, the support vector machine, and with tree, the regression trees.

Algorithm	*Fixation*	*Saccade*	*Pursuit*	*Noise*
EV	0.24738	0.299	0	1
I2MC	0.92439	0.1034	0	0
LS	0.95498	0	0.069998	0
IBDT	0.97286	0.27932	0.84794	0
knn5	0.97987	0.73688	**0.91866**	0.7007
knn10	**0.98271**	0.70756	0.91818	0.70534
knn15	0.9826	0.69907	0.91552	0.70766
knn20	0.9833	0.68287	0.91407	0.70302
tree1	0.9749	0.92207	0.88462	0.72622
tree2	0.97431	0.91898	0.88631	0.72158
tree3	0.97431	**0.92978**	0.88825	0.73318
tree4	0.9749	0.92207	0.88462	0.72622
tree5	0.97313	0.92824	0.88873	**0.77494**
svm-lin	0.95691	0.85648	0.61357	0.76334
svm-pol	0.82828	0.84414	0.9078	0.72158

Table 2: Recall for each eye movement type for all algorithms. Parameters for the proposed approach are $pf = 8, pw = 6, pa = \frac{360}{25}$.

In tables 2 and 3, the results are shown. We used recall ($TP/(TP+FN)$) and precision ($TP/(TP+FP)$) as metrics. Recall specifies how effective the method is in detecting the correct type, and precision is how often this type was misclassified. As can be seen, the algorithm

Algorithm	*Fixation*	*Saccade*	*Pursuit*	*Noise*
EV	0.7843	0.31543	0	0.012518
I2MC	0.76897	0.071334	0	0
LS	0.77184	0	0.23741	0
IBDT	0.93263	0.73577	0.76837	0
knn5	0.96352	0.91563	0.91821	0.8274
knn10	0.96151	0.93381	**0.92084**	0.88116
knn15	0.96035	0.9477	0.91818	0.87143
knn20	0.95912	**0.95469**	0.91628	**0.89381**
tree1	0.97095	0.90393	0.89895	0.79241
tree2	0.97	0.91334	0.89824	0.78734
tree3	0.97108	0.91705	0.89342	0.84267
tree4	0.97095	0.90393	0.89895	0.79241
tree5	**0.97219**	0.91553	0.88851	0.84772
svm-lin	0.91414	0.86991	0.76867	0.79661
svm-pol	0.969	0.42651	0.77419	0.2681

Table 3: Precision for each eye movement type for all algorithms. Parameters for the proposed approach are $pf = 8, pw = 6, pa = \frac{360}{25}$.

EV [10] detects all errors in the data (table 2) but also misclassified most of the data as errors (table 3). Our feature, in combination with all machine learning approaches and also with all configurations, outperforms the state-of-the-art. The regression tree performs best in our evaluation, but is not the most reliable (table 3). KNNs seem to have problems with saccades and noise but do not misclassify data, as can be seen in table 3. Therefore, they are the most reliable machine learning approach in our evaluation. The svm performed better than the state-of-the-art, but had the lowest results out of all machine learning approaches. The detection rate (recall) increased using a polyniomal kernel (svm-pol), but this made the classifier also less reliable (table 3).

Table 4 shows the best-performing machine learning approach configurations (tree3, tree5, knn10 and knn20) for different parameter settings of the proposed feature. The highest impact is on the noise detection, where the knn loses approximately 20% and the regression trees about 10%. The detection accuracy for saccades is also decreased by approximately 30% for the knns. In contrast, the regression trees are still capable of detecting the saccades, and are even better for approximately 1% for some parameters. Fixations and smooth pursuits are detected equally to the base configuration: Due to the orientation of our feature. Usually, they are difficult to separate based on the velocity, but they differ in the orientation of the movement. This is why with our feature, which includes the orientation between two gaze points, it is possible to separate them. For the saccades and noise, it is difficult for both machine learning approaches to separate them, since with the low sampling rate of the data set (30 Hz), both are usually one sample long. For higher sampling rates, this changes since saccades and blinks (one cause of noise) are longer and contain therefore relative movement information.

Parameter	*Algorithm*	*Fixation*	*Saccade*	*Pursuit*	*Noise*
$pf = 8$	knn10	0.98089	0.71682	0.91624	0.7007
$pw = 6$	knn20	0.98143	0.68904	0.91045	0.68213
$pa = \frac{360}{15}$	tree3	0.97474	0.92824	0.89042	0.71694
	tree5	0.97377	0.93056	0.88921	0.76798
$pf = 8$	knn10	0.98185	0.74074	0.92421	0.7007
$pw = 6$	knn20	0.98143	0.71682	0.92059	0.70302
$pa = \frac{360}{35}$	tree3	0.97522	0.9267	0.88849	0.78654
	tree5	0.97532	0.92824	0.88969	0.78654
$pf = 8$	knn10	0.98169	0.73302	0.92469	0.70534
$pw = 6$	knn20	0.98148	0.70833	0.91914	0.70766
$pa = \frac{360}{45}$	tree3	0.97479	0.92824	0.89331	0.7587
	tree5	0.97452	0.93056	0.89042	0.7703
$pf = 8$	knn10	0.98303	0.73534	0.91576	0.7007
$pw = 4$	knn20	0.98469	0.70833	0.90852	0.68445
$pa = \frac{360}{25}$	tree3	0.97324	0.9267	0.88704	0.7587
	tree5	0.97227	0.93133	0.88559	0.77262
$pf = 8$	knn10	0.98218	0.68904	0.92035	0.7123
$pw = 8$	knn20	0.98223	0.65586	0.91673	0.68677
$pa = \frac{360}{25}$	tree3	0.97639	0.91821	0.89331	0.7355
	tree5	0.97752	0.92438	0.89573	0.7587
$pf = 6$	knn10	0.98303	0.73534	0.91673	0.70766
$pw = 6$	knn20	0.98464	0.70293	0.91335	0.69374
$pa = \frac{360}{25}$	tree3	0.97265	0.92438	0.90152	0.74478
	tree5	0.97233	0.92284	0.90176	0.7819
$pf = 6$	knn10	0.98394	0.7608	0.91383	0.6891
$pw = 4$	knn20	0.98565	0.72994	0.90997	0.67749
$pa = \frac{360}{25}$	tree3	0.97179	0.92824	0.88559	0.76334
	tree5	0.97222	0.92593	0.88438	0.78886
$pf = 10$	knn10	0.98078	0.69444	0.91624	0.7123
$pw = 6$	knn20	0.98164	0.66435	0.91166	0.70302
$pa = \frac{360}{25}$	tree3	0.97425	0.91821	0.8909	0.73318
	tree5	0.97425	0.9267	0.89476	0.77262
$pf = 10$	knn10	0.98014	0.66821	0.91624	0.70998
$pw = 8$	knn20	0.981	0.64352	0.90997	0.68677
$pa = \frac{360}{25}$	tree3	0.97179	0.91512	0.8909	0.71694
	tree5	0.97372	0.92824	0.8909	0.76566

Table 4: Recall for knn and regression trees using different parameter settings for feature computation. The first row shows the used parameters for the evaluation.

5 CONCLUSION

We proposed a novel feature that can be used for eye movement detection in combination with machine learning approaches. It outperforms the state-of-the-art and can be used for online and offline detection. We evaluated different machine learning approaches with different configurations. All outperformed the state-of-the-art. In addition, our evaluation included different parameter sets for our feature computation. We will provide a Matlab script for feature computation and an integration into EyeTrace which both can be downloaded at *http://www.ti.uni-tuebingen.de/Eyetrace.1751.0.html.*

REFERENCES

[1] Richard Andersson, Linnea Larsson, Kenneth Holmqvist, Martin Stridh, and Marcus Nyström. 2017. One algorithm to rule them all? An evaluation and discussion of ten eye movement event-detection algorithms. *Behavior Research Methods* 49, 2 (2017), 616–637.

[2] Zillah Boraston and Sarah-Jayne Blakemore. 2007. The application of eye-tracking technology in the study of autism. *The Journal of physiology* 581, 3 (2007), 893–898.

[3] Christian Braunagel, David Geisler, Wolfgang Rosenstiel, and Enkelejda Kasneci. 2017. Online Recognition of Driver-Activity Based on Visual Scanpath Classification. *IEEE Intelligent Transportation Systems Magazine* 9, 4 (2017), 23–36.

[4] David P Crabb, Nicholas D Smith, Franziska G Rauscher, Catharine M Chisholm, John L Barbur, David F Edgar, and David F Garway-Heath. 2010. Exploring eye movements in patients with glaucoma when viewing a driving scene. *PloS one* 5, 3 (2010), e9710.

[5] Heiner Deubel, Bruce Bridgeman, and Werner X Schneider. 1998. Immediate post-saccadic information mediates space constancy. *Vision research* 38, 20 (1998), 3147–3159.

[6] Andrew T Duchowski. 2007. Eye tracking methodology. *Theory and practice* 328 (2007).

[7] Ralf Engbert and Reinhold Kliegl. 2003. Microsaccades uncover the orientation of covert attention. *Vision research* 43, 9 (2003), 1035–1045.

[8] Ralf Engbert and Konstantin Mergenthaler. 2006. Microsaccades are triggered by low retinal image slip. *Proceedings of the National Academy of Sciences* 103, 18 (2006), 7192–7197.

[9] Wolfgang Fuhl, Thiago Santini, Thomas Kuebler, Nora Castner, Wolfgang Rosenstiel, and Enkelejda Kasneci. 2018. Eye movement simulation and detector creation to reduce laborious parameter adjustments. *eprint arXiv:1804.00970* (2018).

[10] Roy S Hessels, Diederick C Niehorster, Chantal Kemner, and Ignace TC Hooge. 2017. Noise-robust fixation detection in eye movement data: Identification by two-means clustering (I2MC). *Behavior research methods* 49, 5 (2017), 1802–1823.

[11] Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost Van de Weijer. 2011. *Eye tracking: A comprehensive guide to methods and measures.* OUP Oxford.

[12] Qiang Ji, Zhiwei Zhu, and Peilin Lan. 2004. Real-time nonintrusive monitoring and prediction of driver fatigue. *IEEE transactions on vehicular technology* 53, 4 (2004), 1052–1068.

[13] Enkelejda Kasneci, Gjergji Kasneci, Thomas C Kübler, and Wolfgang Rosenstiel. 2014. The applicability of probabilistic methods to the online recognition of fixations and saccades in dynamic scenes. In *Proceedings of the Symposium on Eye Tracking Research and Applications.* ACM, 323–326.

[14] Enkelejda Kasneci, Katrin Sippel, Kathrin Aehling, Martin Heister, Wolfgang Rosenstiel, Ulrich Schiefer, and Elena Papageorgiou. 2014. Driving with binocular visual field loss? A study on a supervised on-road parcours with simultaneous eye and head tracking. *PloS one* 9, 2 (2014), e87470.

[15] 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.

[16] Oleg V Komogortsev and Javed I Khan. 2009. Eye movement prediction by oculomotor plant Kalman filter with brainstem control. *Journal of Control Theory and Applications* 7, 1 (2009), 14–22.

[17] Thomas C Kübler, Colleen Rothe, Ulrich Schiefer, Wolfgang Rosenstiel, and Enkelejda Kasneci. 2017. SubsMatch 2.0: Scanpath comparison and classification based on subsequence frequencies. *Behavior Research Methods* 49, 3 (2017), 1048–1064.

[18] 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.

[19] Linnéa Larsson, Marcus Nyström, and Martin Stridh. 2013. Detection of saccades and postsaccadic oscillations in the presence of smooth pursuit. *IEEE Transactions on Biomedical Engineering* 60, 9 (2013), 2484–2493.

[20] R John Leigh and David S Zee. 2015. *The neurology of eye movements.* Vol. 90. Oxford University Press, USA.

[21] James G May, Robert S Kennedy, Mary C Williams, William P Dunlap, and Julie R Brannan. 1990. Eye movement indices of mental workload. *Acta psychologica* 75, 1 (1990), 75–89.

[22] Marcus Nyström and Kenneth Holmqvist. 2010. An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. *Behavior research methods* 42, 1 (2010), 188–204.

[23] Dario D Salvucci and Joseph H Goldberg. 2000. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the Symposium on Eye Tracking Research and Applications.* ACM, 71–78.

[24] Thiago Santini, Wolfgang Fuhl, Thomas Kübler, and Enkelejda Kasneci. 2016. Bayesian identification of fixations, saccades, and smooth pursuits. In *Proceedings of the Symposium on Eye Tracking Research and Applications.* ACM, 163–170.

[25] Enkelejda Tafaj, Gjergji Kasneci, Wolfgang Rosenstiel, and Martin Bogdan. 2012. Bayesian online clustering of eye movement data. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 285–288.

[26] Ralf van der Lans, Michel Wedel, and Rik Pieters. 2011. Defining eye-fixation sequences across individuals and tasks: the Binocular-Individual Threshold (BIT) algorithm. *Behavior Research Methods* 43, 1 (2011), 239–257.

[27] Giacomo Veneri, Pietro Piu, Pamela Federighi, Francesca Rosini, Antonio Federico, and Alessandra Rufa. 2010. Eye fixations identification based on statistical analysis-case study. In *Cognitive Information Processing (CIP), 2010 2nd International Workshop on*. IEEE, 446–451.

[28] Giacomo Veneri, Pietro Piu, Francesca Rosini, Pamela Federighi, Antonio Federico, and Alessandra Rufa. 2011. Automatic eye fixations identification based on analysis of variance and covariance. *Pattern Recognition Letters* 32, 13 (2011), 1588–1593.

[29] Heino Widdel. 1984. Operational problems in analysing eye movements. *Advances in psychology* 22 (1984), 21–29.

[30] Raimondas Zemblys. 2016. Eye-movement event detection meets machine learning. *BIOMEDICAL ENGINEERING 2016* 20, 1 (2016).

[31] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and Shai Avidan. 2006. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, Vol. 2. IEEE, 1491–1498.