Bayesian Online Clustering of Eye Movement Data

Enkelejda Tafaj*
Computer Engineering
University of Tübingen
Germany

Gjergji Kasneci†
Hasso-Plattner-Institute
Germany

Wolfgang Rosenstiel‡
Computer Engineering
University of Tübingen
Germany

Martin Bogdan§
Computer Engineering
University of Leipzig
Germany

Abstract

The task of automatically tracking the visual attention in dynamic visual scenes is highly challenging. To approach it, we propose a Bayesian online learning algorithm. As the visual scene changes and new objects appear, based on a mixture model, the algorithm can identify and tell visual saccades (transitions) from visual fixation clusters (regions of interest). The approach is evaluated on real-world data, collected from eye-tracking experiments in driving sessions.

CR Categories: I.5.3 [Computing Methodologies]: Pattern Recognition—Clustering

Keywords: online clustering, eye movement data, Bayesian model, fixation clusters

1 Introduction

During visual perception (although we are mostly unaware of it) our eyes are constantly moving. Eye movements enable the fovea – the retinal part of sharpest vision – to fixate different parts of the scene, thus preventing sensory adaptation in our visual path by refreshing our retinal images. These foveal fixations are known as human Regions-Of-Interest (ROIs) [Privitera and Stark 2005] and are explained by the *scanpath theory* by Noton and Stark [Noton and Stark 1971]. This theory states that a top-down internal cognitive model of what we "see" not only controls our vision, but also efficiently drives the sequences of rapid eye movements and fixations over a scene [Privitera and Stark 2005].

Research on visual perception has largely benefited from the development of eye-tracking devices and accurate methods for quantifying eye movements; e.g., state-of-the-art eye trackers allow the recording of eye movements at high sampling rates, up to 500Hz. The efficient detection of visual fixations or ROIs in eye tracker data is essential for scanpath analysis and can be modeled as the problem of identifying clusters in a set of data points. While the annotation of such clusters is trivial for us humans, automated clustering of eye-movement data is still challenging; even more so, when the visual scene changes in an online fashion, e.g., measuring the visual scanning behavior of people while driving, watching advertisements, shopping, etc.

A simple group of clustering algorithms is based on a distance-threshold; two points are considered to be in the same cluster if they are closer to each other than a predefined distance threshold [Privitera and Stark 2000], [Turano et al. 2003]. Although quite intuitive, these algorithms fail to identify dense but separate regions of interest, e.g., when two of the fixation cluster points do not meet the threshold criteria. Other approaches partition the visual scene into a regular grid and record the time spent inside each square [Salvucci and Goldberg 2000]. While these algorithms are well suited to certain applications, such as reading, they do not generalize to scenarios with no a priori information about the visual scene. Similarly restricted are techniques that provide visualizations of ROIs by adapting learned visual models, e.g. [Wooding 2002]. Yet other algorithms are data-driven and typically based on the mean shift procedure [Santella and DeCarlo 2004]. Such algorithms are not applicable to dynamic visual scenes, as they require a clustering parameter as input. Over the last years many of the above proposals have been implemented in academic and commercial tools and many eye tracker manufacturers provide software for analyzing recorded eye movements. Despite several useful features provided by these tools, their main drawback is that they come as black-box solutions and cannot be integrated in self-designed applications. Besides, much of the commercial software in this realm is typically geared towards offline analysis of eye movements. Academic tools such as the recently published MATLAB-toolbox GazeAlyze [Berger et al. 2011] based on ILab [Gitelman 2002] or ASTEF [Camilli et al. 2008] can be easily integrated in self-designed applications but, unfortunately, only for offline analysis.

Imagine a driving assistance system that records the driver's eye movements and analyzes them to warn the driver about entities (e.g., traffic participants) she might have overlooked. An essential requirement for such a system is the online analysis of the driver's scanpath with respect to the entities that appear on the visual scene. As a consequence, any algorithm used to cluster fixation points has to be unparameterized (as new entities may appear on the scene). Note that the system has to know the driver's ROIs at any point in time. Furthermore, as the viewing behavior differs from person to person, an adaptive algorithm is needed.

We present an unparameterized, adaptive online algorithm for clustering fixation points in scenarios such as the above.

2 Bayesian Online Mixture Model

Imagine a temporally ordered sequence of T two-dimensional points, $\mathbf{S} = \{s_i \mid 1 \leq i \leq T\}$, recorded by an eye tracker. Assuming that these points reflect the visual scanpath of an observer over time, a dense region of sequential points (i.e., points that are close to each-other in terms of Euclidean distance), might reflect a ROI (or, more specifically, an object that attracts the observer's attention).

Assuming that the recorded points are normally distributed around a ROI, the corresponding distribution could be approximated by exploiting covariances derived from the coordinates of the fixation points in the ROI. However, when the number of observation points is rather moderate, such an approximation typically leads to poor results. Hence our algorithm is based on the intuition that the dis-

*e-mail: tafaj@informatik.uni-tuebingen.de
†e-mail: gjergji.kasneci@hpi.uni-potsdam.de
‡e-mail: rosenstiel@informatik.uni-tuebingen.de
§e-mail: bogdan@informatik.uni-leipzig.de

ETRA 2012, Santa Barbara, CA, March 28 – 30, 2012.
© 2012 ACM 978-1-4503-1225-7/12/0003 $10.00

tances between sequential fixation points in a ROI and the distances between sequential saccade points come from two different Gaussian distributions. The resulting (reduced) dimensionality allows us to efficiently deal with a moderate number of observation points. The parameters of these Gaussians (i.e., means and variances) can be learned through a generative mixture model. The Bayesian network in Figure 1 depicts a mixture model for the two Gaussian distributions.

Let $\mathbf{D} = \{d_i \mid 1 \leq i \leq T-1\}$ be the set of distance variables between points $s_i, s_{i-1} \in \mathbf{S}$. $\Theta = \{\mu_1, \beta_1, \mu_2, \beta_2\, \pi_1, \pi_2\}$ denotes the complete parameter set of the mixture model in Figure 1. The mixture component is denoted by the variable z_i and the observed distance by the observed variable d_i. The simplifying assumption here is that the distances are generated sequentially in an i.i.d. fashion. More specifically, each distance between two sequential points is generated independently by the most likely Gaussian distribution.

The joint probability distribution of the model is given by:

$$\begin{aligned} p(\mathbf{D}, \mathbf{z}|\Theta) &= \prod_{i=1}^{T-1} p(z_i = z|\pi) p(d_i|\mu_z, \beta_z) \\ &= \prod_{i=1}^{T-1} \pi_{z_i} N(d_i; \mu_{z_i}, \beta_{z_i}) \end{aligned}$$

where $\mathbf{z} = \{z_1, ..., z_{T-1}\}$, with $z_i \in \{1, 2\}$ being the index of the mixture component chosen for distance d_i, and $\pi = \{\pi_1, \pi_2\}$ denotes the set of mixture parameters.

We have used Infer.NET[1] to specify the model with the following distributions.

(1) The distribution over the mixture component variables:

$$p(\mathbf{z}|\pi) = \prod_{i-1}^{T-1} \pi_{z_i}$$

(2) The prior distribution over the model parameters:

$$p(\Theta) = p(\pi)p(\mu)p(\beta)$$

(3) The prior over the mixture parts as a symmetric Dirichlet distribution:

$$p(\pi) = Dir(\pi; \lambda)$$

(4) The prior distribution over the means as a product of Gaussians:

$$p(\mu) = N(\mu_1; m, \tau)N(\mu_2; m, \tau)$$

(5) The prior distribution over the precisions as a product of Gammas:

$$p(\beta) = Gam(\beta_1; n, \gamma)Gam(\beta_2; n, \gamma)$$

2.1 The online model

The above Bayesian model comes with the great benefit that it can be easily turned into an online learning model. In general, for given model parameters Θ and observations D, after applying Bayes' rule follows that the probability of the parameters Θ in light of the data D is:

$$p(\Theta|D) = \frac{p(D|\Theta)p(\Theta)}{p(D)}$$

More generally, we can write:

$$p(\Theta|D) \propto p(D|\Theta)p(\Theta)$$

[1] http://research.microsoft.com/en-us/um/cambridge/projects/infernet/

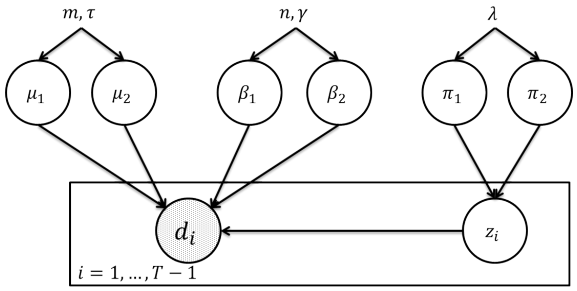

Figure 1: *Bayesian Mixture Model for clustering based on the assumption that the distances d_i between sequential points in a ROI and saccade points come from two different Gaussian distributions.*

The above formula suggests that in an online setting the prior of the parameters $p(\Theta)$ can be iteratively substituted with the posterior $p(\Theta|D)$, while the likelihood on the parameters $p(D|\Theta)$ helps readjust the model, as more and more observations are made (see also [Bishop 2006]).

This insight allows us to readjust the learned parameters μ, β, π as more and more eye movement data is available. To this end, we define Gaussian distributions as priors for the means μ_1, μ_2, Gamma distributions as priors for the precisions β_1, β_2, and Dirichlet distributions as priors for π_1, π_2. All these distributions are learned as new data points are observed, and in each iteration, their priors are updated by their posteriors. The whole model was implemented in C# and Infer.NET. Also, for the probabilistic inference on the model, we have used Variational Message Passing as implemented by Infer.NET. The code can be made available on request.

3 Experimental Results

3.1 Data

To evaluate the proposed Bayesian online mixture model for clustering, we used eye-tracking data, collected from driving sessions with different human subjects. The scope of the study was much broader and involved the investigation of visual scanning behavior (and its impact on the driving performance) of three groups of subjects: (1) subjects suffering from homonymous visual field defects (loss of the field of vision at the same relative position in both eyes), (2) subjects suffering from glaucoma (a disease involving the optic nerve degeneration, thus leading to irreversible loss of vision) and (3) control subjects. In order to record the eye movements, we used a Dikablis[2] mobile infrared eye tracker, at a sampling rate of 25Hz. The scene is captured at an image resolution of 768x576px. After a three-point calibration routine, the gaze and scene information is synchronized online.

For the evaluation of the model we used the raw data as exported by the eye-tracking system. It is important to notice that this did not involve any data cleaning or preprocessing. Gaze points with corrupted position information (e.g., due to unsuccessful pupil detection by the eye-tracking system or missing scene information) are not excluded from the data; nor are saccade points. As the detection of ROIs is performed online, and the gaze points are processed sequentially, the information used by the algorithm consists only of the coordinates of the gaze points in the scene image.

The assessment of the visual scanning differences between the study participants is out of the scope of this paper and will be pre-

[2] http://www.ergoneers.com/de/products/dlab-dikablis/overview.html

sented in a later publication. Here we focus only on the clustering performance of the algorithm.

3.2 Results

Although the algorithm was evaluated on multiple video sequences, for the sake of space, we report experimental results for three typical scenes from the driving sessions of two different subjects. The detected clusters are depicted in Figures 2, 3 and 4. Note that (because of space restrictions) the figures depict representative frames from a longer video sequence. The gaze points from the sequences were used by the algorithm to detect the fixation clusters. The frames were chosen in such a way that they represent as much of the sequence information as possible. In order not to overload these scene images, the detected fixation clusters (in Figure 2 and Figure 3) or saccade points (in Figure 4) are presented in separate images above or below the scene. Different colors stand for different fixation clusters. Dotted lines connect the detected clusters with the corresponding objects or traffic participants that were fixated by the subject in the sequences. The single red points in Figure 4 represent sequential saccade points. The chronological viewing order of the entities in the scene is denoted by time stamps t_i.

The first 200 gaze points (corresponding to 8 seconds) at the beginning of each driving session were used by the model to learn the individual viewing behavior of the subject. This way the algorithm can adapt to a new subject. All the following gaze points were processed in an online fashion, as described in Section 2.1.

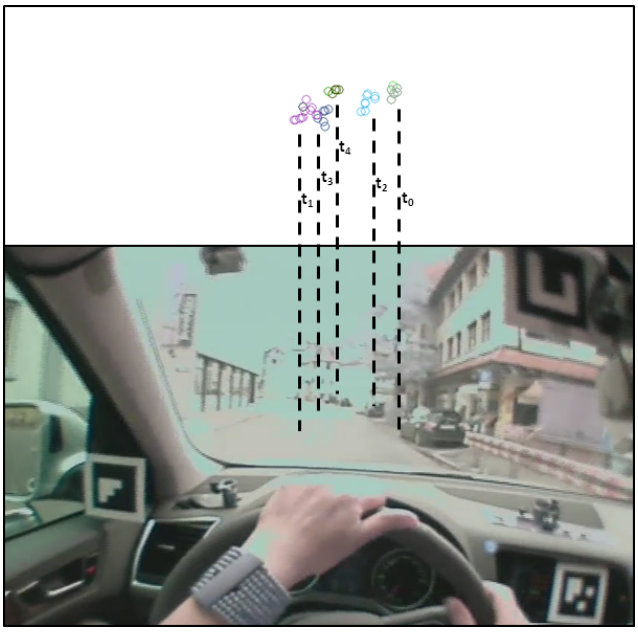

Figure 2: *Regions-of-interest for a driving sequence of 1.76 sec length (subject 1). During this driving sequence (corresponding to 44 frames) the model detects 5 fixation clusters time-stamped according to their chronological order $t_0 \ldots t_4$. At the beginning of this sequence the subject was fixating the first car on the right side of the road (t_0). A few frames later the gaze was directed toward the middle of the road (t_1). The ROI in the actual frame corresponds to the car connected to the t_2-cluster. Yet a few frames later, the subject fixates again the road (t_3) and finally, in the last frames a car (t_4) in the rear of the road is fixated.*

It is important to note that, depending on the driving speed, the scene information can change very quickly. Objects and traffic participants appear within the driver's visual field for a very short time period. Furthermore, for every new entity appearing on the scene, when the entity is fixated by the subject, the algorithm needs to recognize a new cluster in an online fashion. This means that the number of possible clusters is not known beforehand. Consequently, parameterized clustering algorithms are not applicable to these scenarios.

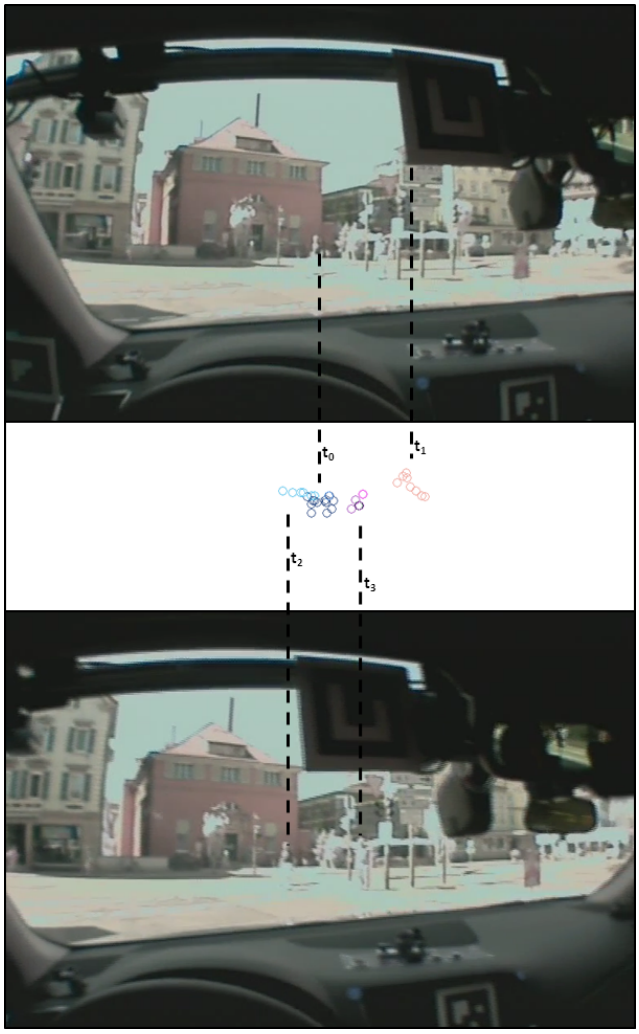

Figure 3: *Regions-of-interest for a road junction scene (subject 2). The four clusters (or regions-of-interest) detected here correspond to the entities fixated during a driving sequence of 1.2 seconds. The frame on the top presents the scene at the beginning of this sequence. First the subject fixates the person crossing the street (time-stamped by t_0). After that, the gaze of the subject is directed towards the traffic sign on the right side (t_1). The lower frame corresponds to a scene image captured about 700 ms later. At this point in time the subject fixates again the person crossing the street (t_2) and moves later his visual attention towards another person crossing the street (t_3).*

As we can see, in the depicted figures, the viewing behavior during driving is characterized by brief fixations. Hence the algorithm has to efficiently make sense of very few new gaze points and decide whether they correspond to a fixation cluster or to saccades. In all experiments the algorithm dealt with the incoming data in real time. We hypothesize that the performance would be unchanged, even if the gaze points were sampled at a much higher rate. Fur-

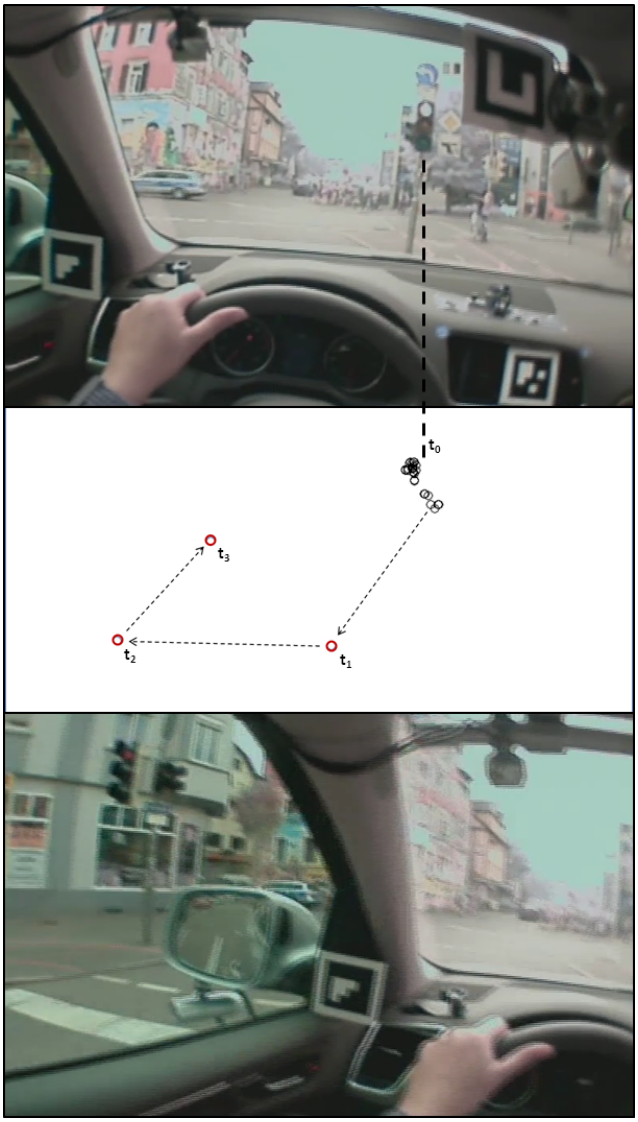

Figure 4: *Fixation cluster and saccade points at a traffic light (subject 1). The single cluster detected in this sequence corresponds to the traffic light as fixated by the subject (see upper-most frame) in the beginning of the sequence (time-stamped by t_0). A few frames later, the subject's gaze is directed towards the traffic light on the left side of the road (see lower frame). This happens over two saccades (t_1, t_2), which were detected correctly by the algorithm. The latter traffic light though is not fixated by the subject, as one frame later the subject shifts the gaze back (t_3). It is interesting to see that for the t_0-cluster the algorithm has decided that all gaze points belong to the same cluster, although there are small distances between the points. And indeed these points correspond to the same fixated object. This indicates that the algorithm has correctly adapted to the subject's viewing behavior.*

thermore, although the raw data is noisy and the viewing behavior is subject-dependent, the proposed algorithm adapts very quickly to the individual viewing characteristics of each subject and performs robustly in detecting regions-of-interest in an online fashion.

Overall, the mentioned strengths allow the application of the algorithm to independent viewing sequences, containing only few gaze points. This flexibility makes it adequate for broad applications in vision or HCI research, where the online analysis of gaze-based interaction (e.g., tracking a subject's attention or detecting overlooked entities) is crucial.

4 Conclusion

We have presented an unparameterized, adaptive online algorithm for clustering eye movement data. The experiments conducted so far have shown that the algorithm performs strongly and in real-time on raw data collected from eye-tracking experiments in driving sessions, with different participants. Further experiments are needed to evaluate the real-time performance of the algorithm at higher sampling rates. As future work, we have planned to evaluate its performance on a broader set of online applications and to integrate it with existing analysis tools, such as Vishnoo [Tafaj et al. 2011]. However, we think that this algorithm already satisfies several crucial criteria that would make it a core ingredient for many online analysis tools of vision research.

References

BERGER, C., WINKELS, M., LISCHKE, A., AND HÖPPNER, J. 2011. GazeAlyze: a MATLAB toolbox for the analysis of eye movement data. *Behavioral Research Methods*, 1–16.

BISHOP, C. M. 2006. *Machine Learning and Pattern Recognition*. Springer-Verlag New York, Inc. Secaucus, NJ, USA.

CAMILLI, M., NACCHIA, R., TERENZI, M., AND NOCERA, F. D. 2008. Astef: A simple tool for examining fixations. *Behavior Research Methods 40*, 373–382.

GITELMAN, D. 2002. Ilab: a program for postexperimental eye movement analysis. *Behavioral Research Methods, Instruments and Computers 34*, 4, 605–612.

NOTON, D., AND STARK, L. W. 1971. Eye movements and visual perception. *Scientific American 224*, 6, 34–43.

PRIVITERA, C. M., AND STARK, L. W. 2000. Algorithms for defining visual regions-of-interest: Comparison with eye fixations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 9, 970–982.

PRIVITERA, C. M., AND STARK, L. W. 2005. Scanpath theory, attention, and image processing algorithms for predicting human eye fixations. In *Neurobiology of Attention*, L. Itti, G. Rees, and J. Tsotsos, Eds., 269–299.

SALVUCCI, D., AND GOLDBERG, J. 2000. Identifying fixations and saccades in eye-tracking protocols. In *In Proceedings of the Eye Tracking Research and Applications*, 71–78.

SANTELLA, A., AND DECARLO, D. 2004. Robust clustering of eye movement recordings for quantification of visual interest. In *Proceedings of the 2004 Symposium on Eye Tracking Research & Applications*, 27–34.

TAFAJ, E., KÜBLER, T., PETER, J., SCHIEFER, U., BOGDAN, M., AND ROSENSTIEL, W. 2011. Vishnoo - an open-source software for vision research. In *Proceedings of the 24^{th} IEEE International Symposium on Computer-Based Medical Systems*.

TURANO, K., GERUSCHAT, D., AND BAKER, F. 2003. Oculomotor strategies for the direction of gaze tested with a real-world activity. *Vision Research 43*, 333–346.

WOODING, D. 2002. Fixation maps: quantifying eye-movement traces. In *In Proceedings of the Eye Tracking Research and Applications*, 31–36.