In the pursuit of (ground) truth: A hand-labelling tool for eye movements recorded during dynamic scene viewing

Ioannis Agtzidis* Mikhail Startsev† Michael Dorr‡

Technical University Munich

ABSTRACT

We here present parts of our ongoing work to facilitate the large-scale analysis of smooth pursuit eye movements made while viewing dynamic natural scenes. Classification of smooth pursuit episodes can be difficult in the presence of eye-tracking noise, and we thus recently proposed an algorithm that clusters gaze recordings from several observers in order to improve classification robustness. We now implemented a publicly available tool that allows for generation of a ground truth benchmark by assisted hand-labelling of video gaze data. Based on the labelling produced with the tool we present preliminary evaluation results for our smooth pursuit classification approach in comparison to state-of-the-art algorithms. Overall, human observers spend more than 12% of their viewing time performing smooth pursuit, which emphasizes the importance of investigating smooth pursuit behaviour in naturalistic contexts.

Index Terms: H.5.2 [Information Interfaces And Presentation]: User Interface—Evaluation; H.5.2 [Information Interfaces And Presentation]: User Interface—Interaction styles; I.5.3 [Pattern Recognition]: Clustering—Algorithms; I.5.5 [Pattern Recognition]: Implementation—Interactive systems

1 INTRODUCTION

Because of the space-variant resolution of the human retina, eye movements are a fundamental component of visual information processing. For many extensively studied visual tasks such as reading or inspecting an image, the eyes alternate between fixations, mostly stationary phases in which visual information is being processed, and saccades, rapid ballistic movements that bring the fovea to new informative regions of the stimulus. When dynamic stimuli are being presented, however, the eyes may also perform smooth pursuit movements that track moving targets to keep them foveated. While smooth pursuit eye movement behaviour has been investigated in depth using simple, synthetic stimuli such as moving dots on an otherwise blank background, there is still a need to better understand smooth pursuits in more complex, naturalistic paradigms and tasks such as video-watching or even navigation in the real world. In these contexts, the exact trajectories of potential pursuit targets are often not known a priori, and in the absence of a clear instruction to pursue certain objects, a post-hoc classification of gaze data into fixations, saccades, and episodes of smooth pursuit has to be performed. Because of their potentially low speed, smooth pursuit eye movements are hard to distinguish from fixations, especially in noisy gaze recordings. A recently developed approach to classify smooth pursuits made while viewing naturalistic videos [1] now aims at increasing reliability by using information across multiple observers: After an initial step that labels those fixations and saccades that were detected with high confidence (very low and very high speed, respectively), remaining samples are classified as smooth pursuit only if other observers exhibited a similar gaze pattern at the same spatio-temporal video location. By contrast, noise in the gaze recording and other eye movement types should be independent across observers, and idiosyncratic gaze patterns that are neither (confidently detected) fixation nor saccade are thus labelled as noise. In a pilot evaluation [1], this approach yielded very promising results, outperforming existing algorithms [2, 6] that only look at the gaze traces of individual observers in isolation. Notably, our approach improved both precision and recall, even though the original design goal was to only filter out noise, i.e. increase precision. This evaluation was based on a "ground truth" that was obtained with a very simplified Matlab tool. This tool allowed the labelling of gaze only in relatively coarse time windows of 250 ms each, and without visualizing the video stimulus and the corresponding gaze data at multiple time scales.

For a more thorough evaluation of our novel eye movement classification algorithm and other algorithms that detect smooth pursuit episodes, however, a solid large-scale ground truth would be desirable. Typically, human raters are used to establish such a baseline, even though it should be stressed that in the absence of electrophysiological recordings, the ground truth may only be inferred based on the observed, potentially noisy, gaze signal, which may even suffer from artefacts due to eye-tracking technology [8]. At any rate, manual labelling needs tools to support this arduous task. Existing approaches typically support the coding only of fixation and saccades [7, 11]. For the coding of smooth pursuit episodes, the tool should also provide an appropriate visualization of the stimulus video with its moving objects, i.e. potential pursuit targets.

In the following, we shall present our ongoing work towards such a tool that assists the user in hand-labelling eye movement data that was recorded while viewing dynamic stimuli. After a brief review of automated smooth pursuit detection, we shall describe this tool that simultaneously visualizes gaze data and the moving image content. We shall also review the use of the popular ARFF file format for eye-tracking data.

Finally, we shall present first results for our algorithm and two state-of-the-art smooth pursuit detectors [2, 6] run against the ground truth obtained with this tool on a subset of the GazeCom data set [3].

2 AUTOMATIC SMOOTH PURSUIT DETECTION

2.1 Data set

In this work we use the publicly available GazeCom eye movement data set. More than 50 observers were freely viewing 18 short video clips (20 s each) depicting outdoor scenes in and around a city with a number of potential pursuit targets, such as moving cars and pedestrians.

This data set was chosen partly because of its size: with an appropriate tool it is still possible to hand-label the entire set within a reasonable time interval. Overall, the data set comprises about 5 hours of gaze recordings at 250 Hz; in our ongoing effort to have multiple raters label the entire data set, however, we initially focus on only a subset of about 28 minutes that has been used in a

*e-mail: ioannis.agtzidis@tum.de
†e-mail: mikhail.startsev@tum.de
‡e-mail: michael.dorr@tum.de

previous study [1].

2.2 Proposed smooth pursuit detection algorithm

We here briefly review our smooth pursuit detection algorithm based on clustering the gaze points of multiple observers simultaneously in space and time [1]. The idea behind this is that if we leave out the samples that represent fixations or saccades, what is left should normally occur in the same place at the same time only in the case of smooth pursuits. Other, idiosyncratic gaze motion types and even more so any tracker-induced artefacts should exhibit less spatio-temporal dependence on the presented stimuli.

Consequently one can look at the proposed approach as the following pipeline:

1. Discard gaze samples that were reliably classified as fixation or saccade, respectively
2. Perform clustering in 3-dimensional space (x, y, t coordinates of the gaze sample): detecting "dense" groups of samples and marking them as clusters. To make use of different properties of gaze traces (e.g. velocity direction or speed), higher-dimensional spaces may also be employed.
3. Optionally, perform post-processing to remove some clusters that are unlikely to represent smooth pursuit:
   - too few observers within one cluster
   - duration of the cluster is too short to make a smooth pursuit possible
   - high directional variance across gaze trajectories of different observers within one cluster
4. Optionally, merge some clusters based on their similarity

3 LABELLING TOOL

The designed tool is a hybrid between automatic labelling such as described above and pure hand labelling of eye movements. As a first step, eye movements are detected algorithmically with every algorithm output occupying a column in the data file. Then, we add an extra column for hand labelling and we populate it with the most frequent label for each sample from the algorithmic part. Finally, the user can manipulate the suggested eye movement labels through the provided interface. An example screenshot is shown in Fig. 1.

3.1 UI

The interaction with the user is handled through four panels. The user is able to scroll through the video by dragging the "current position" marker on either of the right-side panels.

3.1.1 Playback panels

On the left half of the window we display the video itself and a dense velocity field of the video on the top and the bottom panels, respectively, if either is available. Here, the velocity field was precomputed with the EpicFlow algorithm [10]; in principle, however, any feature map with the dimensions of the original video may be used.

The video panel also displays the gaze samples from a 200 ms temporal window centred at the current time. In order to visualize temporal order, "history" samples (before the current time position) are represented by red circles, and "future" samples by gray circles. The last sample up to the current time position is indicated by a green circle.

The flow panel additionally displays a rectangle that visualizes the spatial support of all gaze samples that are visible in the video panel. Because the size of the rectangle is proportional to the dispersion of gaze, it is indicative of fixations (low dispersion) and saccades (high dispersion).

The mean of the velocity field inside this rectangle is displayed in the coordinate system on the top left part of the flow panel as a red vector. The mean of the sample-to-sample gaze points velocity for the visible samples is illustrated by a blue vector. These visualizations show the similarity between image motion and corresponding eye motion: a smooth pursuit that follows a moving target with near-perfect gain should yield equal vectors, whereas very different directions are likely indicative of an eye movement other than smooth pursuit.

3.1.2 Interactive panels

On the right half of the application window we display the plots for x and y coordinates of the eye movement data over time t. These panels handle most of the user actions.

The background colour represents the most frequently assigned label to each sample from the algorithmically detected eye movements in order to speed-up hand labelling by providing relevant suggestions.

These background colours also serve the purpose of separating the entire sequence of gaze samples into intervals of different type (i.e. they may have have different labels: fixation, saccade, smooth pursuit, noise, or not yet labelled).

Most of the user interaction activity is dealing with these intervals. The following actions can be performed through these panels (the right side of the window):

- Right-clicking and dragging moves the current position in time (backwards or forwards according to the direction of the mouse movement).
- Scrolling the mouse wheel changes the temporal scale, i.e. increases or decreases (according to the scroll direction) the temporal window represented by the plots on the right-side panels.
- Left-clicking and dragging expands (moves) the borders of the interval, where the left click occured.
- Holding the left-click on an interval and pressing a number on the keyboard changes the label of the interval. The legend provides information on the correspondence between numbers and the assigned labels.
- The sequence of a left-click, a number key press and finally pressing the *Insert* key inserts a new interval of the selected type spanning a temporal window of $\pm$ 40 ms around the current time; this interval can then be adjusted as above.
- The sequence of a left-click and pressing the *Delete* key unassigns the label of the selected interval.

The interaction is completed with some standard keyboard shortcuts:

- Pressing *Space* key starts playing or pauses the video.
- Pressing *Ctrl-Z* reverts the last change.
- Pressing *Ctrl-Shift-Z* acts as "redo" (reapplies the last cancelled change).

The last two shortcuts are platform-dependent. The application menu contains the information about all the used key sequences, including the correct values for the platform-dependent ones.

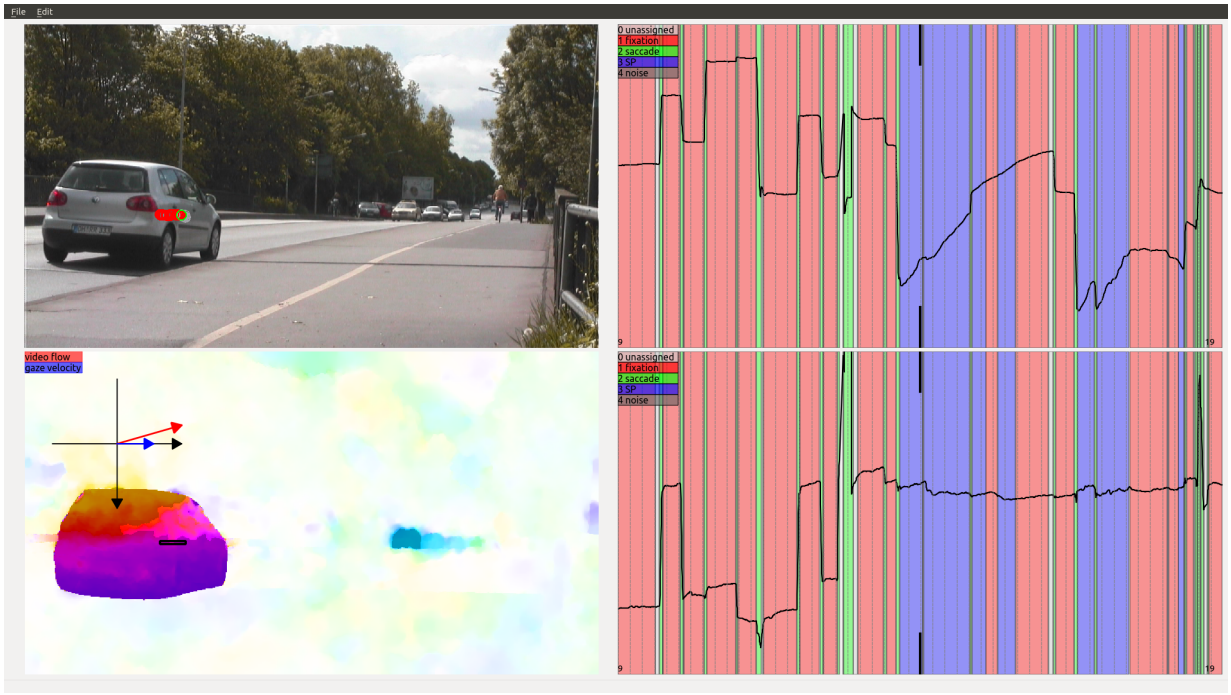

Figure 1: Screenshot of the labelling tool. The video with overlaid gaze traces is playing in the top left, while x- and y-coordinates are plotted as a function of time t on the right. The bottom left panel shows a dense velocity field for the video and the average gaze and image velocity vectors in a spatio-temporal neighbourhood around the point of regard.

3.2 Extendable data file format ARFF

An ARFF (Attribute-Relation File Format) file is a text file that describes a list of instances sharing a set of attributes. This file format is used for example in WEKA [4], a machine learning software suite very popular in the data mining community. The popularity of ARFF files is mirrored by a plethora of tools for manipulating them in a variety of languages such as Python, C++, Java, Matlab, R, etc. The use of a common standard with bindings to many programming languages could be beneficial to the eye tracking tracking community, with its proprietary file formats used by each group separately, and may foster scientific content exchange as well as speed up the evaluation of ideas.

In ARFF, all keywords start with a "@" symbol and the following names are case-insensitive; all lines starting with "%" are considered comments.

Any file comprises two sections for a header and the data. The header starts with "@relation", which defines the relation name. After this, the attributes can be declared through the "@attribute" keyword followed by the name and type of the attribute.

The data section starts with the "@data" keyword. The further lines describe the instances with one instance per line and comma-separated attributes. The attributes should follow the same order used for their declaration in the header section.

In our implementation, we introduce minor deviations from the regular ARFF format, but maintain major conventions and thus keep the compatibility with WEKA. Since an eye-tracking experiment needs information about the video resolution and the dimensions of the monitor, we introduce "special" comments (maintaining ARFF compatibility) with the "%@metadata" notation. So far, the *pixelx*, *pixely*, *dimensionx*, *dimensiony* and *distance* values are handled (describing the vital parameters of an eye-tracking data recording: monitor resolution and physical dimensions as well as the viewing distance from the observer's eyes to the monitor).

An example of such an ARFF file with meta-attributes is presented below. At the beginning we see the name of the relation described in this file followed by five lines of metadata about the experiment. The following three attributes represent data that were returned from the eyetracker. The subsequent attributes in this example hold labels that were returned from algorithms that detect smooth pursuit (SP-DBSCAN, a variant of the algorithm described in section 2.2), saccades (I-VVT – identification with a velocity-velocity threshold), and fixations (I-DT – identification with a dispersion threshold), respectively, and hand-assigned labels.

After all information about the experiment and the attributes is given, the "@data" keyword marks the start of the data instances. In the data section the first instance was marked as saccade by the dual velocity threshold algorithm and the expert agreed. The next sample was left unassigned by all the algorithms but the expert assigned it to the following fixation. Finally the last two gaze samples (instances) were detected as fixations by the dispersion threshold algorithm and the expert labeller agreed.

Listing 1: Sample ARFF file.

```
@RELATION gaze_labels

%@METADATA PIXELX 1280
%@METADATA PIXELY 720
%@METADATA DISTANCE 0.45
%@METADATA DIMENSIONX 0.40
%@METADATA DIMENSIONY 0.23

@ATTRIBUTE time NUMERIC
@ATTRIBUTE x NUMERIC
@ATTRIBUTE y NUMERIC
@ATTRIBUTE sp-dbscan NUMERIC
@ATTRIBUTE i-vvt NUMERIC
@ATTRIBUTE i-dt NUMERIC
@ATTRIBUTE handLabel1 NUMERIC
```

```
% Label types for the 4 attributes above
% NOT_ASSIGNED   0
% FIXATION       1
% SACCADE        2
% SP             3
% NOISE          4

@DATA
1000,416.90,188.00,0,2,0,2
5000,417.00,188.10,0,0,0,1
9000,417.00,189.20,0,0,1,1
13000,417.10,190.30,0,0,1,1
```

3.3 Implementation

The implementation is separated in two parts. The first part is the UI that enables the hand labelling itself. The second part is a library providing manipulation functionality for ARFF files with the constraints of the previous section.

The UI is written in C++ with Qt 5 providing the infrastructure. By using Qt we obtain a strong foundation for a responsive UI and the ability to run on different platforms. The video playback uses the Qt multimedia library; tests at least on the Ubuntu Linux 14.04 platform were performed to ensure that seeking backwards and forwards does not introduce accumulating timing errors.

For the modified, yet standard-compatible ARFF files we implemented a simple C++ library. This library provides reading/writing from file, addition/deletion of attributes and some other useful operations such as the majority vote of labels assigned to each instance.

The tool and its source code are publicly available at http://www.michaeldorr.de/gaze_labelling.

4 Results

We use the ground truth labels obtained with the presented tool to evaluate several smooth pursuit detection algorithms. Classification performance of our algorithm with three different clustering methods (Graph-based, SP-DBSCAN, and SP-LST) as well as two state-of-the-art algorithms is shown in Table 1. These results are qualitatively very similar to those obtained with a previous hand-labelling tool that had only coarse temporal resolution [1]; F1 scores now slightly improved, likely due to the a more fine-grained labelling that results in fewer samples being labelled as smooth pursuit. It is worth noting that to prevent bias the suggested (automatically detected) "smooth pursuit" labels were hidden from the experts. Hand-labelling of 20 s of gaze data with the proposed tool took an expert user about 3 to 5 minutes.

Overall, about 12.7% of all gaze samples in the GazeCom subset studied here were manually labelled as smooth pursuit; for the algorithms, this rate ranged from 2% to 7%.

Table 1: Precision, recall, F1 and false positive rate (FPR) throughout all the ground truth data.

	Precision	Recall	F1	FPR
Graph-based	0.843	**0.462**	**0.597**	0.012
SP-DBSCAN	0.859	0.449	0.590	0.010
SP-LST	**0.944**	0.144	0.251	**0.001**
Berg et al.	0.561	0.372	0.447	0.041
Larsson et al.	0.477	0.229	0.310	0.035

5 Discussion

In this paper, we reported on ongoing work to analyse eye movements made while viewing dynamic natural scenes, with an emphasis on smooth pursuit eye movements. To this end, we implemented a tool to quickly and reliably hand-label a "ground truth" of gaze data.

One obvious limitation of the current tool is that visualization and labelling are still limited to the gaze traces of one individual observer at a time, even though our results show that – automatic – classification of eye movements may benefit from the integration of information across several observers. It thus could be helpful to extend the tool with a visualization of multiple observers at a time, similar to the iSeeCube approach [5].

A further limitation concerns the algorithmic approach for smooth pursuit classification. With naturalistic videos presented on a computer screen, all observers see the same stimulus, and spatio-temporal clustering of gaze data is thus straightforward. For truly naturalistic environments, e.g. recordings by head-mounted cameras from subjects freely interacting with the real world, this is not the case anymore, and more sophisticated approaches are required; one possibility would be to attempt to automatically match image regions [9].

Finally, our results show that in naturalistic videos, observers spend a substantial amount of time following moving targets with smooth pursuit eye movements, much more than detected by automated eye movement analysis, where the best algorithm still had less than 50% recall. Further study of both eye movement classification and smooth pursuit in naturalistic contexts therefore seems desirable.

Acknowledgements

This research was supported by the Elite Network Bavaria, funded by the Bavarian State Ministry for Research and Education.

References

[1] I. Agtzidis, M. Startsev, and M. Dorr. Smooth pursuit detection based on multiple observers. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, ETRA '16, pages 303–306, New York, NY, USA, 2016. ACM.

[2] D. J. Berg, S. E. Boehnke, R. A. Marino, D. P. Munoz, and L. Itti. Free viewing of dynamic stimuli by humans and monkeys. *Journal of Vision*, 9(5):1–15, 5 2009.

[3] M. Dorr, T. Martinetz, K. Gegenfurtner, and E. Barth. Variability of eye movements when viewing dynamic natural scenes. *Journal of Vision*, 10(10):1–17, 2010.

[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[5] K. Kurzhals, F. Heimerl, and D. Weiskopf. ISeeCube: Visual analysis of gaze data for video. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '14, pages 43–50, New York, NY, USA, 2014. ACM.

[6] L. Larsson, M. Nyström, R. Andersson, and M. Stridh. Detection of fixations and smooth pursuit movements in high-speed eye-tracking data. *Biomedical Signal Processing and Control*, 18:145–152, 2015.

[7] S. M. Munn, L. Stefano, and J. B. Pelz. Fixation-identification in dynamic scenes: Comparing an automated algorithm to manual coding. In *Proceedings of the 5th symposium on Applied perception in graphics and visualization*, pages 33–42. ACM, 2008.

[8] M. Nyström, I. Hooge, and K. Holmqvist. Post-saccadic oscillations in eye movement data recorded with pupil-based eye trackers reflect motion of the pupil inside the iris. *Vision research*, 92:59–66, 2013.

[9] D. F. Pontillo, T. B. Kinsman, and J. B. Pelz. SemantiCode: Using content similarity and database-driven matching to code wearable eyetracker gaze data. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA '10, pages 267–270, New York, NY, USA, 2010. ACM.

[10] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Computer Vision and Pattern Recognition*, 2015.

[11] I. R. Saez de Urabain, M. H. Johnson, and T. J. Smith. Grafix: A semi-automatic approach for parsing low- and high-quality eye-tracking data. *Behavior Research Methods*, 47(1):53–72, 2015.