# Towards Mining Eye-Tracking Datasets for Stress Detection

Mahnaz Behroozi
CSC Department
North Carolina State University
Raleigh, NC
Email: mbehroo@ncsu.edu

***Abstract*—Problem-solving on a whiteboard is a popular technical interview technique used in industry. However, several critics have raised concerns that whiteboard interviews can cause excessive stress and cognitive load on candidates, ultimately reinforcing bias in hiring practices. Due to the increased availability of eye tracking devices, it is becoming increasingly feasible for researchers in the software engineering field to collect eye gaze data. Depending on the type of eye tracker, a large amount of data with many different attributes are generated in a short amount of time. We posit that such a dataset (after cleansing) is a rich source for additional information about candidate's stress and cognitive load. In this research proposal, we seek for data mining algorithms which can effectively discriminate the sequences of gaze data taken from candidates in a stressful interview setting and in a more relaxed one.**

***Keywords*—*Data mining, software engineering, eye-tracking, technical interviews***

## I. Introduction

A technical interview is a stage of a job interview, which for software developers, often includes a programming component performed on a whiteboard. A common goal of a technical interview is to obtain visibility into and verifiability of the cognitive processes used by a candidate [14]. Although technical interviews are the most common assessment technique, they have several limitations:

- **Medium and affordances.** Whiteboards are often selected for high visibility of the problem-solving work by interviewers; however, whiteboards lack affordances, such as syntax highlighting, which can cause higher cognitive load [34].
- **Stress.** Public performance, time pressure, and self-efficiency can influence cognitive state [18].
- **Interruption.** Technical interviews require that a candidate perform talk-aloud while problem-solving, which imposes high cognitive load [16, 20] and eliminates opportunity for reflection [11].

When software processes and tools are not well-aligned with the cognitive processing styles of certain populations [21], discrimination can occur. For example, research into gender differences has established that many software tools are designed to be more supportive of problem-solving processes used by males than by females [4]. Critics of whiteboard interviews argue that the technique systematically biases hiring certain candidates [32, 1], due to lack of time to practice for the interview setting, and familiarity with problems. Finally, research in criteria used to evaluate candidates during technical interviews finds that factors [14], such as confidence matter as much as problem-solving ability, meaning that when selecting between two candidates producing the same quality answer, the least visibly stressed candidate is more likely to get the job. Thus, traditional technical interviews can overlook more skillful candidates who do not perform well in these settings.

## II. Criteria

This section briefly expands some of data mining model criteria and explains why each is useful/useless or easy/hard.

### A. *Model Readability*

Model readability is an important criteria from different aspects. In the first place, when the model is readable, it is more easily to be described to the business user. Convincing a business user is one of the most important parameters that makes a model acceptable and usable. Secondly, when a model is readable, it is easier to be interpreted and easier to be maintained. In general, more readable models are also easier to implement. However, achieving and keeping model readability is not an easy task. For example, decision trees are readable when the problem size is small but they will become complex and large as the problem size increases.

### B. *Actionable Conclusions*

Data mining can drive actionable information from large amount of data. Based on the actionable information extracted, a learner can take actionable conclusions. In other words, actionable conclusion deals with the interpretation of the various features of the data and how they help us in modeling the training data to draw decisions on unseen data. For example, a car renting agency can predict the high demanding areas of the town based on the requests made by the costumers over the past two years and can make a decision to increase the number of their cars in those areas. Or for example, a grocery store can increase the number of a particular goods in a particular branch based on mining the shopping history of the costumers. Making actionable conclusions is not always an

easy and accurate task. The key point in a successful actionable conclusion is how successful our model is in generalizing. So, it is not always an easy task to make actionable conclusions and in critical domains the cost of a wrong conclusion is devastating.

### C. *Learnability and Repeatability of the Results*

When researchers do research it is important that the results they achieve be learnable and repeatable because other researchers should be able to replicate the results in the case they want to improve the study or to build a new research on the basis of the previous study. An example of small memory model which produces learnable and repeatable results is Naive Bayes. Although simple models are easy to implement and they are light weight, sometimes they suffer from being accurate and in some areas of research they are not quite robust. So, it is important when to use which model.

### D. *Multi-goal Reasoning*

In real world problems, usually there are more than one goals to satisfy. In Computer Science, Goal models have been widely used to represent software requirements, business objectives, and design qualities [22]. However, Existing goal modelling techniques have shown limitations of expressiveness and/or tractability in coping with complex real-world problems. One strategy to make multi-goal reasoning less complex is to cast the problem into single-goal reasoning such as using domination score of the multiple goals[13].

### E. *Anomaly Detection*

In data mining, anomaly detection or outlier detection refers to the identification of items, events or observations which do not match an expected pattern or data distribution in a dataset [8]. Typically the anomalous items will translate to some kind of problem such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies are also referred to as outliers, novelties, noise, deviations and exceptions [17]. Anomaly detection faces many challenges such as tackling with dynamically changing applications such as intrusion detection. In streaming application, the learner cannot hold all the data and the model should be updated on the fly. Also, the data distribution and pattern is changing over the time. So, the learner should be capable of remembering the safe past patterns. Otherwise it will produce many false alarms. The choice of the appropriate anomaly detection method is also critical. For example, in intrusion detection, the target objects are not rare objects but a unexpected burst of activity. Which means that unsupervised anomaly detection methods will fail on detecting this anomalies.

### F. *Incremental*

Incremental learning is a method updating the existing model knowledge. This method is used when the data size is larger to be hold in system memory or when the data will be gradually feed into the model. Incremental learning represents a dynamic technique of supervised learning and unsupervised learning. Incremental learning adapts new data without forgetting its existing knowledge without retraining the model. Fuzzy ART[7] and TopoART[33] are two examples for stable incremental machine learning algorithms which learn representations of the training data that are not even partially forgotten over time Incremental algorithms are frequently applied to data streams or big data, addressing issues in data availability and resource scarcity respectively. Stock trend prediction and user profiling are some examples of data streams where new data becomes continuously available. Applying incremental learning to big data aims to produce faster classification or forecasting times.

### G. *Sharable*

Data privacy is one of the main concerns of Data Science society. Although, results repeatability is another important factor which helps researchers to replicate each others research results, when it comes to data sharing things will become a little complected. One solution is data obfuscation and encapsulation. For the sake of privacy, there are some methods which masks the details of the data but still it can be used for the sake of learnability and repeatability of the results. As an example, KDD also uses these data masking methods to share the data with the participants but still keep it private.

### H. *Context Aware*

Many of today's state of the art data learning models visualize the data in aggregated for and pay little or no attention to portions of the data which has been obtained under different contexts. The data is analyzed as an homogeneous body of information without disaggregating them to analyze in different situations or groups. Sometimes the whole data analysis will result in missing out important trends appearing in portions of data. As a result, context awareness is important in data analysis.

### I. *Self-tuning*

Self-tuning or auto-tuning model refers to a model that is capable of optimizing its own internal running parameters in order to fulfill an objective function; typically the maximization of efficiency or error minimization.

Self-tuning models are typically composed of four components: expectations, measurement, analysis, and actions. Measurements gather data about the conditions and behaviour. Analysis helps determine whether the expectations are being met and which subsequent actions should be performed. Common actions are gathering more data and performing dynamic reconfiguration of the model. Self-tuning (self-adapting) models of automatic control are models whereby adaptation to randomly changing conditions is performed by means of automatically changing parameters or via automatically determining their optimum configuration [31].

## III. KEY CRITERIA

In eye-tracking analysis, the most important criteria are Context awareness and multi-goal reasoning. Context awareness is

important in the field of eye-tracking since every parameters extracted from the eye-tracker should not be considered as effective parameters. In different settings, there might be more important parameters which should be analyzed along with other relevant parameters or as an stand alone parameter. For example, in application of eye-tracking in operation a motor vehicle, analyzing number of gaze regressions seems an off topic analysis while in code comprehension or study of cognitive load during a technical interview it can play an important role.

In our study, we proposed comparing two interview settings: On the board and on the paper. Literature in eye-tracking analysis has shown that there are various eye-movement features which are the indicators of cognitive load and high stress levels. But in our proposed settings there are some other factors that might interfere the descriptive features, such as the wideness of the board in comparison to the paper. As a result, we have to find a way to apply the effect of the mentioned difference and to use some sort of domination score in a way that it can explain the effect of the setting difference in our results and interpretations.

## IV. Critique and Review

A few known studies have looked into clustering eye movements. Privitera and Stark [24] compared actual and simulated foveations using k-means clustering. Latimer [19] used a form of k-means based on histograms of foveation durations in an effort to classify partitions more robustly, although he comments that k-means clustering often produces inconsistent results. Santella and DeCarlo [26] quantified visual interest via a mean-shift clustering framework. Though this method is robust to noise and does not require the number of clusters a priori, it does require additional tuning of parameters based on temporal and spatial scales. Success of density estimation via expectation/ maximization[3] of a parametric family distribution in Sawahata et al. [27] has been reported. Sawahata et al. investigated an entropy measure calculated from the log likelihood of a mixture model in order to determine viewer comprehension during dynamic scene viewing. This method when combined with model selection and enough data points can produce consistent results that are robust to noise and does not require the number of clusters a priori (which is the main drawback of k-means clustering). A prior model of the distribution of data is required in order to accurately tune the parameters during expectation/maximization [20].

## V. Planing

Our work investigates the use of head-mounted eye-trackers that can obtain eye movement measures without restriction to movement. We use highly reliable fiducial markers, specifically ArUco markers [], that are robust to occlusion. ArUco markers can be placed in the environment in order to help map gaze coordinates to physical locations(See Figure 1). Our primary research question in this proposal is: ***Do data mining techniques such as clustering or gaze position sequence mining can help us detect differences in stress and cognitive load between the paper and whiteboard technical interview settings?***

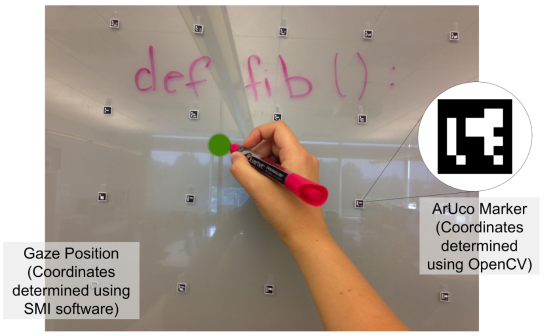


Figure 1. Feasibility study of using ArUco markers to calculate regressions.

Figure 2. A head-mounted eye-tracker from SMI

We hypothesized that the public setting of the whiteboard would increase cognitive load while the private setting of the paper would reduce it. Majority of our participants also confirm our hypothesis after being interview in both of the settings.

Our dataset comes from our previously done pilot study with 11 participants, where participants wore a head-mounted eye-tracker (see Figure 2) and we measured the difference between solving a programming task on paper and on the whiteboard. We were able to recover the ArUco markers and use them to obtain measures, such as regressions.

Using a Latin Square design to rotate experimental conditions, we observed that participants solving problems on the whiteboard appear to experience higher cognitive load (based on higher regressions, higher fixation dispersion, longer blinks, larger saccade velocity, longer saccade duration). Further, participants self-rated the whiteboard setting as being more stressful, we observed several nervous tics displayed by participants, and we obtained several measures consistent with stress (shorter fixation duration, larger saccade velocity, and longer saccade duration).

We plan to use sequence mining to find similarities in eye-tracking gaze patterns of the participants to see if we can classify them based on the presence or absence of high cognitive load and stress. One of the challenges here is that the participants wrote their own code and they did not comprehend a unique code. Hence, we have to extract some similarities and then try to cluster them or to classify them

based on some similarity measures. In addition we try to do unsupervised learning approaches such as $k$-means clustering based on the well-known parameters such as fixation duration to investigate if this method can cluster the candidates who experienced higher cognitive load and those who experienced lower stress/cognitive load.

There is also an interesting research which presents an approach to compare saccadic eye movement sequences based on the Needleman–Wunsch algorithm used in bioinformatics to compare DNA sequences [9]. It is worth investigating to see if it works in the case of our research.

The performance of each of the above approaches can be assessed with standard measures such as accuracy, precision, recall, F-measure, and Area Under the Curve (AUC). To further evaluate results, we can use tests such as t-test and two-way analysis of variance (ANOVA) to investigate statistically significant differences between the results. Moreover, we can compare the algorithms and approaches based on their time and space complexity to see which one is more practical to be used.

## VI. Related work

The science of eye-tracking analysis goes back to over a century ago in reading research [23]. Today, eye-tracking as a research is growing in popularity in different disciplines such as usability analysis, sports, psychology, marketing , human computer interaction (HCI), software engineering and many more. Software engineering community started showing interest in eye-tracking studies in the early nineties in investigating the reading strategies in code comprehension [29] The first eye-tracking study in software engineering has been conducted by Crosby et al. [10] on analyzing reading strategies and their impact on procedural code comprehension.

However, before 2006, eye-trackers have not been vastly used in software engineering society due to high price of eye-trackers and the difficulties in using them [2]. As a result, many of the previous studies in software engineering domain used Restricted Focus Viewer approach (RFV) [24]. In this approach, only a small portion of the screen has been displayed to the user and the rest of the screen was blurred. A user controls which part of the screen is of regard with the help of the computer mouse. Since RFV impacts the study results by restricting the subjects and unrealistic conditions, software engineering researchers started using other strategies to conduct their research since 2006.

One method of annotating videos is to instrument applications with a plug-in that records data as a task is completed [28]. The instrumentation approach builds on previous methods in that it provides support for non-static applications, but it fails to capture gazes in non-instrumented areas. Therefore, all possible regions must be determined a priori. For example, prior studies require the participants use Eclipse exclusively. In order to improve this approach, the researcher would need to similarly instrument other applications, such as browsers and text editors. A second method is to defer video annotation until after data collection, but this can be limiting and ineffective.

Unfortunately for researchers, the support available to tag dynamic AOIs in programming environments like Eclipse is limited [25]. The main focus areas of software engineering eye-tracking studies can be grouped in 5 main categories [29] from which we are more interested in the first two categories:

- Model comprehension
- Code comprehension
- Debugging
- Collaborative interactions
- Traceability

Under the category of model comprehension, De Smet et al. in [12] investigated the impact of different design patterns on code comprehension using UML class diagrams. They ran their study on 18 students and 8 faculty members and the variables they took into consideration from EyeLink II eye-tracker were spatial density, transitional matrix, average fixation duration, time and scan-path distance. Another study of the same category has been conducted by Soh et al. [30] where they studied the relation between expertise and professional status for UML diagram comprehension.

In code comprehension studies, the subjects will be asked to read pieces of source code to answer comprehension questions. One of the earliest studies of this category is the research of Crosby and Stelovsky in 1990 [10]. In their study, using Pascal source codes of binary search algorithm, they tried to assess the impact of expertise on the developers' comprehension strategies. Another study conducted by Busjahn et al. [6] investigates the differences between source code reading and natural text reading. They also studies attention distribution on code elements to classify experts' and novices' code reading strategies in [5]. Fritz et al. [15] studied the task difficulty using psycho-physiological measures such as electroencephalography (EEG), eye-gaze, electrodermal activity (EDA), and NASA TLX scores [16].

## References


[1] Chris Alvino. *Technical Interviews an Instrument of Exclusion and Discrimination*. 2014. URL: https://careerconservatory.com/technical-interviews-an-instrument-of-exclusion-and-discrimination/.

[2] Roman Bednarik and Markku Tukiainen. "Effects of Display Blurring on the Behavior of Novices and Experts During Program Debugging". In: *CHI '05 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '05. Portland, OR, USA: ACM, 2005, pp. 1204–1207. ISBN: 1-59593-002-7. DOI: 10.1145/1056808.1056877. URL: http://doi.acm.org/10.1145/1056808.1056877.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[4] Margaret Burnett et al. “Finding Gender-Inclusiveness Software Issues with GenderMag: A Field Investigation”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. Santa Clara, California, USA: ACM, 2016, pp. 2586–2598. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858274. URL: http://doi.acm.org/10.1145/2858036.2858274.

[5] Teresa Busjahn, Roman Bednarik, and Carsten Schulte. “What Influences Dwell Time During Source Code Reading?: Analysis of Element Type and Frequency As Factors”. In: *Proceedings of the Symposium on Eye Tracking Research and Applications*. ETRA ’14. Safety Harbor, Florida: ACM, 2014, pp. 335–338. ISBN: 978-1-4503-2751-0. DOI: 10.1145/2578153.2578211. URL: http://doi.acm.org/10.1145/2578153.2578211.

[6] Teresa Busjahn, Carsten Schulte, and Andreas Busjahn. “Analysis of Code Reading to Gain More Insight in Program Comprehension”. In: *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*. Koli Calling ’11. Koli, Finland: ACM, 2011, pp. 1–9. ISBN: 978-1-4503-1052-9. DOI: 10.1145/2094131.2094133. URL: http://doi.acm.org/10.1145/2094131.2094133.

[7] Gail A Carpenter, Stephen Grossberg, and David B Rosen. “Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system”. In: *Neural networks* 4.6 (1991), pp. 759–771.

[8] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly detection: A survey”. In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

[9] Filipe Cristino et al. “ScanMatch: A novel method for comparing fixation sequences”. In: *Behavior research methods* 42.3 (2010), pp. 692–700.

[10] Martha E. Crosby and Jan Stelovsky. “How Do We Read Algorithms? A Case Study”. In: *Computer* 23.1 (Jan. 1990), pp. 24–35. ISSN: 0018-9162. URL: http://dl.acm.org/citation.cfm?id=77577.77580.

[11] Jane Dawson. “Reflectivity, creativity, and the space for silence”. In: *Reflective Practice* 4.1 (2003), pp. 33–39.

[12] Benoît De Smet et al. “Taupe: Visualizing and analyzing eye-tracking data”. In: *Science of Computer Programming* 79 (Jan. 2014), pp. 260–278. ISSN: 0167-6423. DOI: 10.1016/j.scico.2012.01.004.

[13] Kalyanmoy Deb. “Multi-objective optimization”. In: *Search methodologies*. Springer, 2014, pp. 403–449.

[14] Denae Ford et al. “The Tech-talk Balance: What Technical Interviewers Expect from Technical Candidates”. In: *Proceedings of the 10th International Workshop on Cooperative and Human Aspects of Software Engineering*. CHASE ’17. Buenos Aires, Argentina: IEEE Press, 2017, pp. 43–48. ISBN: 978-1-5386-4039-5. DOI: 10.1109/CHASE.2017.8. URL: https://doi.org/10.1109/CHASE.2017.8.

[15] Thomas Fritz et al. “Using psycho-physiological measures to assess task difficulty in software development”. In: *Proceedings of the 36th International Conference on Software Engineering*. ACM. 2014, pp. 402–413.

[16] Sandra G Hart and Lowell E Staveland. “Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research”. In: *Advances in psychology* 52 (1988), pp. 139–183.

[17] Victoria Hodge and Jim Austin. “A survey of outlier detection methodologies”. In: *Artificial intelligence review* 22.2 (2004), pp. 85–126.

[18] S. Klitzman et al. “Work stress, nonwork stress, and health.” In: *Journal of behavioral medicine* 13.3 (June 1990), pp. 221–243. ISSN: 0160-7715. URL: http://view.ncbi.nlm.nih.gov/pubmed/2213867.

[19] CR Latimer. “Eye-movement data: Cumulative fixation time and cluster analysis”. In: *Behavior Research Methods, Instruments, & Computers* 20.5 (1988), pp. 437–470.

[20] Parag K Mital et al. “Clustering of gaze during dynamic scene viewing is predicted by motion”. In: *Cognitive Computation* 3.1 (2011), pp. 5–24.

[21] Meredith Ringel Morris, Andrew Begel, and Ben Wiedermann. “Understanding the Challenges Faced by Neurodiverse Software Engineering Employees: Towards a More Inclusive and Productive Technical Workforce”. In: *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*. ASSETS ’15. Lisbon, Portugal: ACM, 2015, pp. 173–184. ISBN: 978-1-4503-3400-6. DOI: 10.1145/2700648.2809841. URL: http://doi.acm.org/10.1145/2700648.2809841.

[22] Chi Mai Nguyen et al. “Multi-objective reasoning with constrained goal models”. In: *Requirements Engineering* (), pp. 1–37.

[23] Alex Poole and Linden J Ball. “Eye tracking in HCI and usability research”. In: *Encyclopedia of human computer interaction* 1 (2006), pp. 211–219.

[24] Claudio M. Privitera and Lawrence W. Stark. “Algorithms for defining visual regions-of-interest: Comparison with eye fixations”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.9 (2000), pp. 970–982.

[25] Paige Rodeghero et al. “Improving Automated Source Code Summarization via an Eye-tracking Study of Programmers”. In: *ICSE*. Hyderabad, India, 2014, pp. 390–401.

[26] Anthony Santella and Doug DeCarlo. “Robust clustering of eye movement recordings for quantification of visual interest”. In: *Proceedings of the 2004 symposium on Eye tracking research & applications*. ACM. 2004, pp. 27–34.

[27] Yasuhito Sawahata et al. “Determining comprehension and quality of TV programs using eye-gaze tracking”. In: *Pattern Recognition* 41.5 (2008), pp. 1610–1626.

[28] Timothy R. Shaffer et al. “iTrace: Enabling Eye Tracking on Software Artifacts Within the IDE to Support Software Engineering Tasks”. In: *FSE*. Bergamo, Italy, 2015, pp. 954–957.

[29] Zohreh Sharafi, Zéphyrin Soh, and Yann-Gaël Guéhéneuc. “A systematic literature review on the usage of eye-tracking in software engineering”. In: *Information and Software Technology* 67 (2015), pp. 79–107.

[30] Zéphyrin Soh et al. “Professional status and expertise for UML class diagram comprehension: An empirical study”. In: *Program Comprehension (ICPC), 2012 IEEE 20th International Conference on*. IEEE. 2012, pp. 163–172.

[31] David G Sullivan, Margo I Seltzer, and Avi Pfeffer. *Using probabilistic reasoning to automate software tuning*. Vol. 32. 1. ACM, 2004.

[32] Rachel Thomas. *How to Make Tech Interviews a Little Less Awful*. 2017. URL: https://medium.com/@racheltho/how-to-make-tech-interviews-a-little-less-awful-c29f35431987.

[33] Marko Tscherepanow, Marco Kortkamp, and Marc Kammer. "A hierarchical ART network for the stable incremental learning of topological structures and associations from noisy data". In: *Neural Networks* 24.8 (2011), pp. 906–916.

[34] Erik Wstlund et al. "Effects of VDT and paper presentation on consumption and production of information: Psychological and physiological factors". In: *Computers in Human Behavior* 21.2 (2005), pp. 377 –394. ISSN: 0747-5632. DOI: http://dx.doi.org/10.1016/j.chb.2004.02.007. URL: http://www.sciencedirect.com/science/article/pii/S0747563204000202.