Classification of graphical user interfaces through gaze-based features

Rishabh Sharma and Hubert Cecotti

Department of Computer Science
College of Science and Mathematics
Fresno State, CA, USA
hcecotti@csufresno.edu

Abstract. Graphical User Interfaces (GUIs) represent a key part for building interactive applications, and they act as a direct interface between users and software. The complexity of the GUIs increases in relation to the number of users interacting with software and performing multiple complex tasks, resulting in software that are mazes of functionalities. The impact of such a GUI system is adverse on the software quality and may result in non-robust software. To measure software quality, one has to measure attributes such as usability, efficiency, and learnability. There is no general approach for measuring software validity subjectively in a non-invasive way, i.e. without directly asking the users to evaluate directly the software through predefined use cases or through questionnaires. In this paper, we investigate this issue by detecting the presence of potential errors in GUI by using the users gaze information collected during user interactions. Users gaze is analyzed to generate thirteen unique features based on statistical moment based on time spent on each control, gaze path, and transitions between controls on the different GUI components present on the screen. The performance is evaluated with five state of the art supervised learning classifiers. The results support the conclusion that incongruent GUI states can be reliably detected at the single trial level through gaze-based features with an accuracy of 76% (one-versus-rest classification) and 73.66 (one-versus-one classification) using a Gaussian Naı̈ve Bayes classifier. These results suggest the possibility of co-development approaches between early stage users and developers through eyetracking.

Keywords: eye-tracking, graphical user interface, classification

1 Introduction

Software usability is one of the most important factor for the design and development of modern GUI (Graphical User Interface) based interactive software, websites, and mobile applications. Unusable user interfaces are probably the single largest reason why encompassing interactive systems computers and users, fail in actual use, or are the source of potential errors. With the rapid development of technology and the presence of easy frameworks for creating GUIs, developers have started designing software with more complex GUI implementations, allowing users to execute multiple tasks with a single click making software more robust and efficient. However, this advancement in the technology allures both users and developers from different knowledge backgrounds, thus maintaining the continuous usability level for all the users remains a challenging task. The majority of software testing approaches assume the availability of an oracle, which is a mechanism against which testers can verify the correctness of the outcomes of test case executions [11]. However, in some cases, an oracle is unavailable or its availability is too expensive to be considered: it is the oracle problem that is a key issue for software testing [20].

Previous research studies have shown that software usability and graphical user interface validation is critical for the development of robust GUI applications that can be used by a large number of users [5, 3, 17, 2]. In Software Industries (SI), Quality Assurance (QA) tester works closely with both programmers and the designers in order to make software more efficient and user friendly. Quality assurance testers make efforts to ensure the correct implementation of the GUI application, with the former following the design specification, which was drawn by the latter. Without such an effort, there is no definite promise that a desired usability will be achieved after the complete implementation of the chosen design [5]. However, the task of the QA tester is tedious as it requires manual effort and a constant attention to be performed correctly and despite such an effort the usability of the software is not guaranteed for all the users. To the majority of users, usability can be summarized in three distinct aspects: effectiveness, efficiency, and satisfaction [6].

Software effectiveness and efficiency are affected by its implementation and can be improved by using software testing techniques or appropriate software development life cycle model. Satisfaction is more an abstract quantity to measure, as it is fathomed by the user's interaction with the GUI of the underlying software and can be measured relatively by analyzing the user behavior during interactions. In the academic research, two main methods are present for analyzing the user behavior for predicting the software usability. First, in the survey method, the numbers of users are provided with a survey form to complete after using the software for a certain amount of time [19, 3, 17]. Second, assessing software usability using computer vision technology, where QA tester are supposed to write a script to automate the GUI interaction and analyze if the automation works according to the provided requirement document [5]. Limitation in the above-mentioned approaches are that the survey-based method is time consuming and based on the subjective assessment of the subjects testing the software [9], whereas the computer vision-based method increases the task of QA tester and is dependent on the visual properties of the display. In addition, computer-vision approaches require labeled training data which is time consuming if the test phase will be limited in time.

A strong interest in software testing and for its improvement over the time with the addition of functionalities, a modern approach is to directly gather information from the users, without asking them explicitly to perform a testing task. Currently, a large number of software are released in non-final versions to users (e.g. early access software), where users who have already paid for the software participate for the enhancement of the application by helping developers in finding bugs. In such a context, the users are both users and testers of the application. By considering this paradigm of dynamic enhancement and correction of an application, we propose to investigate the extent to which it is possible to detect possible issues in an existing application, without asking directly the user to participate to testing phase, i.e., the testing phase is invisible to the user. In order to test such an approach, we propose an eye tracking based approach for predicting the usability of the software. As eye-tracking is free form the subjective assessment of the user using the software and is independent of the screen resolution, it overcomes the limitation of both the proposed method in the academic research for analyzing the user behavior for predicting the usability of the software. Moreover, the eye-tracking technology has been undergoing rapid development with improvements in accuracy, stability and sampling rate making devices more reliable to use [8, 10]. Eye-tracking has a long history in human-computer interaction with multiple applications for people with severe disabilities [12, 13, 15]. Moreover, it has been observed that eye tracking can help people to do scientific research by recording and analyzing visual behavior. We propose in this study an eye tracking approach to analyze the recorded user gaze for classifying the GUI with low usability. To validate this approach, we consider a GUI where users have to verify a series of pairs, as it is common after the validation of a form where users have to check its validity, e.g., if the entered last name corresponds to the last name field. The key contributions of this paper are the creation of a set of features and its evaluation for estimating the presence of an incongruent element in the GUI, i.e., the presence of an error. Because errors can happen at any control, some control invariant detection should happen. It is not as a traditional heat map where the most salient element is highlighted, in the present case, the relevant control or set of controls can change between trials or between GUIs. The problem becomes a pattern recognition problem where features based on the gaze information acquired across the multiple controls should be extracted to determine if the GUI contains an incongruent element or not. The remainder of this paper is organized as follows: Section 2 discusses feature extraction from the gaze data and the GUI created for the experiment to performed for user's gaze collection. In Section 4, the classification results achieved by using various supervised machine learning classifiers on the extracted features are presented. The impact of the results are then discussed in Section 5. Finally, the paper is concluded with potential prospective evolutions in Section 6.

2 Methods

2.1 Features extraction

Contrary to problems where the location of a salient element is located to a specific location, or where multiple trials are combined to determine the saliency of an object in a scene, it is not possible to rely on the same approaches for the detection of an incongruent state in a GUI because the potential user will not spend a large amount of time on the element, and it will correspond to a brief moment during the use of the software. For these reasons, the feature extraction procedure must be robust and invariant to the position of the control or controls that represent an incongruent state in the GUI, and therefore a potential source of error.

For performing the classification of the different types of GUIs, the collected raw data is not enough as it represents the low-level pixel information, which is too noisy and variable across trials. We extracted high-level features from the raw data by performing statistical computation to obtain potential discriminative features. As the information is relevant in both time and distance, we computed the probability density moment on transition between the components, and the time spent on each component, calculating moments (mean, standard deviation, skewness, and kurtosis), and the maximum viewed components. Furthermore, we have calculated the distance traveled by the gaze using Euclidean distance along with the total distance traveled in either direction i.e., horizontal and vertical. The distance travel-based calculation was performed from the centroid of the component (key and value) in the GUI.

We denote by $\mathbf{x}_i$, $\mathbf{y}_i$, and $\mathbf{t}_i$ the recorded (x,y) coordinates and T timestamps of the user's gaze for the i^{th} trial and the total number of gaze points recorded in ith trial are denoted as Ng_i. The values of Ng_i are different for each trial, as it depends on the time taken by the participant for completing each trial. In the subsequent sections, we consider the calculation for a single trial.

2.2 Distance-based features

For the calculation of the distance based feature, we considered a distance vector $\mathbf{d}$ containing the component number such that there exists a transition between the k_1^{th} component and k_2^{th} component, with $k_1 != k_2$ as shown in Algorithm 1 and a 2D distance Matrix M of size $N_c \times N_c$ for ith trial of the experiment as in Algorithm 2 showing the count of the number of transitions between two components during a trial. The transition is estimated from the component number in a row (source) to the component number in a column (destination). The threshold α corresponds to the minimum number of time points corresponding to the minimum duration (in second) spent by a user gazing to a component (label or image).

Centroid Based Feature For the calculation of the centroid based features, we calculated a centroid vector CV such that, for the cth component of the GUI, the centroid value is represented by the coordinate $CV_c = (x_c, y_c)$

$$CV = \{CV_0, CV_1, \dots, CV_{N_c}\} \tag{1}$$

In the present system, the elements on the screen represent couples of controls (label, image). CV_1 to CV_{N_p} represent the centroids of the labels (from top to bottom) and $CV_{N_{p+1}}$ to CV_{N_c} represent the centroids of the images (from top

```
1: Ng total number of recorded gaze reading in a trial
2: $\mathbf{x}_i$ recorded gaze x-coordinate
3: $\mathbf{y}_i$ recorded gaze y-coordinate
4: $\mathbf{t}_i$ recorded timestamp
5: $d = []$ {list of the component numbers between which the transition exists}
6: for k to Ng do
7:     final=k+$\alpha$
8:     comp=compID($x_k$,$y_k$)
9:     if (final $\geq$ k) then
10:        final=$Ng$
11:    end if
12:    while (k<final) and (compID($x_k$,$y_k$)==comp do
13:        k=k+1
14:    end while
15:    if (final == k) then
16:        $d = [d; comp]$
17:    end if
18: end for
```

Algorithm 1: Distance Vector Computation.

```
1: $N_c \leftarrow$ Total number of components in a trial
2: for k to d.length-1 do
3:     $M(d[k], d[k+1]) = M(d[k], d[k+1]) + 1$
4: end for
```

Algorithm 2: Transition Matrix.

to bottom). The labels and the images can be replaced by other type of control. The goal is to detect incongruent couples. All the centroid based statistical measurements are calculated using the centroid vector and distance vector CV for **d** for each trial of the experiment, respectively.

Time-based features For the calculation of the time-based feature, we created a vector **h** that represents the total amount of time (in ms) spent by the participant gazing at each component:

$$h = \{h_1, h_2, h_3, \ldots, h_{N_c}\} \tag{2}$$

Here, h_1 to h_{N_p} represent the time spent by the user gazing on the labels (top to bottom), and $h_{N(p+1)}$ to h_{N_c} represent the time spent by the user gazing at the images (top to bottom). **h** represents a discretized version of a heat map.

The mean of the transition matrix represents the average number of times the user gaze performs a transition from one component to another. Similarly, the mean of the **h** represents the average duration (in ms) spent by the user during a trial.

$$\mu_M = \frac{\sum_{j=1}^{N_c} \sum k = 1^{N_c} M_{jk}}{N_c \cdot N_c} \tag{3}$$

$$\mu_h = \frac{\sum_{j=1}^{N_c} h_j}{N_c} \tag{4}$$

For the standard deviation of the transition matrix, we calculated the amount of dispersion in a transition matrix for ith trial i.e., standard deviation in the gaze transition performed from each component (label and image) in the GUI. Similarly, for the standard deviation of the heat map vector we calculated the standard deviation of the time (in millisecond) spend by the participant in the ith trial of the experiment gazing the component (image and label) in the GUI i.e., the variation in the time spend by the user gazing each component in the GUI.

$$\sigma_M = \sqrt{\frac{\sum_{j=1}^{N_c} \sum k = 1^{N_c} (M_{jk} - \mu_M)^2}{(N_c \cdots N_c) - 1}} \tag{5}$$

$$\sigma_h = \sqrt{\frac{\sum_{j=1}^{N_c} (h_j - \mu_h)^2}{N_c - 1}} \tag{6}$$

For the skewness in the transition matrix we calculated the skewness of the total number of transition performed in the ith trial i.e., the symmetry in the number of gaze transitions between each component (label or image) in the GUI.

Similarly, for the skewness in the heat map vector we calculated the skewness in the time spend (in millisecond) by the participant in the ith trial of the experiment. If the gaze transitions and time distribution belong to a normal distribution, then value of skewness is close to 0. It referred to the third standardized central moment for the probability model.

$$\gamma_M = \frac{1}{(N_c \cdot N_c) - 1} \sum_{j=1}^{N_c} \sum_{k=1}^{N_c} \frac{(M_{jk} - \mu_M)^3}{\sigma_N^3} \tag{7}$$

$$\gamma_h = \frac{1}{N_c - 1} \sum_{j=1}^{N_c} \frac{(h_j - \mu_h)^3}{\sigma_h^3} \tag{8}$$

For the kurtosis of the transition matrix, the measure of the combined weight of the tails relative to the rest of the gaze transition distribution performed from each component (label or image) in the GUI. Similarly, for the kurtosis if the heat map vector we calculated the kurtosis of the time spend by the participant in the ith trial of the experiment i.e., a measure of the combined weight of the tails relative to the rest of the gaze time spend distribution vector (heat map vector) on each component (image and label) in the GUI. If the gaze transitions and time distribution belong to a normal distribution, then the value of kurtosis would be 3.

$$\kappa_M = \frac{1}{(N_c \cdot N_c) - 1} \sum_{j=1}^{N_c} \sum_{k=1}^{N_c} \frac{(M_{jk} - \mu_M)^4}{\sigma_M^4} \tag{9}$$

$$\kappa_h = \frac{1}{N_c - 1} \sum_{k=1}^{N_c} \frac{(h_j - \mu_h)^4}{\sigma_h^4} \tag{10}$$

The maximum number of transitions is defined by:

$$MT = \underset{j}{\operatorname{argmax}} \sum_{k=0}^{N_c} M_{jk} \tag{11}$$

The total distance traveled by the gaze during the ith trial is defined as the Euclidean distance on the centroid vector using distance vector:

$$\begin{aligned} TD = \sum_{j=1}^{N_d - 1} & \sqrt{(CV[\mathbf{d}_j].x - CV[\mathbf{d}_j].x)^2} \\ & +(CV[\mathbf{d}_j].y - CV[\mathbf{d}_j].y)^2 \end{aligned} \tag{12}$$

Total gaze distance - X direction (TDX) an Y direction (TDY) We calculated the linear distance travel in horizontal (or vertical) direction i.e., in x (or y) axis using the x (or y) coordinate of the centroid vector on the distance vector:

$$TDX = \sum_{j=1}^{N_d-1} |CV[\mathbf{d}_j].x - CV[\mathbf{d}_{j+1}].x|$$
$$TDY = \sum_{j=1}^{N_d-1} |CV[\mathbf{d}_j].y - CV[\mathbf{d}_{j+1}].y| \tag{13}$$

The total duration (TD) (in ms) spent by the participant during a trial is obtained by:

$$TD = \sum_{j=1}^{N_c} h_j \tag{14}$$

The total set of features is therefore represented by: the average value of the transition matrix (F1) and the heat map vector (F9), the standard deviation of the transition matrix (F2) and the heat map vector (F10), the skewness of the transition matrix (F3) and heat map vector (F11), the kurtosis of the transition matrix (F4) and heat map vector (F12), the maximum number of transitions (F5), the total distance with the Euclidean distance (F6), the distance only in the x-axis (F7) and the y-axis (F8), and the total duration of a trial (F13).

2.3 Classification

For the experiment in this study, as the recorded data was classified into four different types of GUI, two types of binary classification tasks were performed: one-vs-rest and one-vs-one. Hence, we can classify the given data in four one-vs-rest and six one-vs-one binary comparison for training the classifier. We define the following binary classification tasks: the pairwise classifications $C_p(i, j)$ with $1 \leq i \leq j \leq 4$ corresponding to the classification of Ti vs. Tj and the one vs. all classifications $C_a(i)$ with $1 \leq i \leq 4$ corresponding the classification of Ti vs. the other conditions, where Ti represents GUI type Type i. We used the data that was recorded using the portable eye tracker (Tobii Eye tracker 4C) [1]. We performed the statistical computation on the recorded gaze signal to obtain the above-mentioned features and trained five different classifiers and study the performance of the classifiers for predicting the accurate GUI category. The distribution of the GUI information in the dataset is as follow 50% Type 1 GUI, 15% Type 2 GUI, 15% Type 3 GUI, and 20% Type 4 GUI.

Five classifiers were selected from four different categories for evaluating the ten binary classification mentioned, i.e., a density-based classifier - K Nearest Neighbor (KNN) with K = 5, linear classifier - Support Vector Machine (SVM) with radial basis function and penalty parameter 1.0, and Linear Discriminant Analysis (LDA) with singular value decomposition, tree-based classifier - Random Forest classifier (RFT) with the maximum number of allowed tree is set to

10, and Naïve Bayes classifier [18, 16] - Gaussian Naïve Bayes (GNB) by calculating the probability distribution on Gaussian distribution curve. Performance of each classifier was analyzed by calculating the Area Under the ROC (receiver operating characteristic) Curve (AUC) [7].

3 Experimental protocol

3.1 Participants

Twenty-five healthy adult students and faculties from California State University, Fresno, participated to this study (mean age=23 (3.5), 5 females). Participants were verbally informed about the experimental procedure and purpose prior to the experiment. The experimental procedures involving human subjects described in this study were approved by the Institutional Review Board.

3.2 Design

Participants were asked to be the part of experiment which lasted about 5 minutes to complete all the N_t trials of the experiment. For each trial, participants had to gaze through the GUI for a maximum of 4 seconds and predict the type of GUI as congruent (i.e., GUI with high usability and low learning time - Type 1) or incongruent (i.e., GUI with low usability and high learning time - Type 2, 3, and 4). The different GUIs are depicted in Fig. 1. The distribution of the GUI trials in the dataset is as follow 50, 15, 15, and 20 % for Type 1, 2, 3, and 4, respectively. The gaze coordinates of the participants were recorded during the experiment using a portable eyetracker mounted under the screen. The eyetracker was a Tobii Eye tracker 4C (frequency of 90 Hz). The experiment consists of N_t trials with $N_t = 40$.

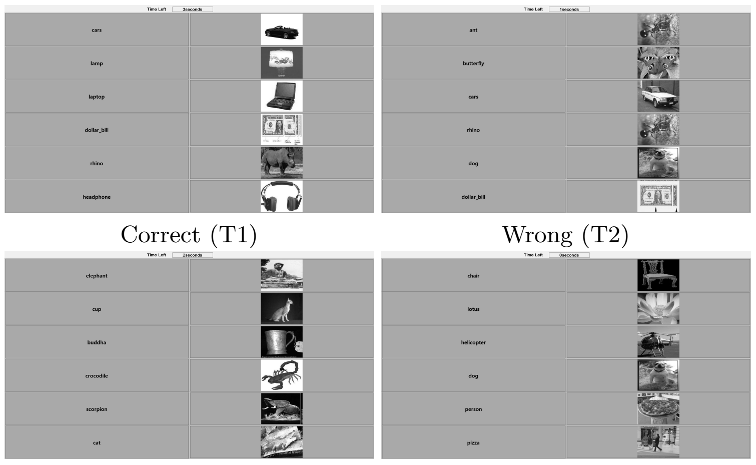
4 Results

4.1 Behavioral analysis

Behavior analysis was performed after calculating the human precision level for classification of the GUIs as congruent or incongruent. The recorded human performance level for the classification across all the participants was 92.5%. Furthermore, we derive the confusion matrix from the prediction value to obtain the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) values in the prediction (see Table 1). The value in each cell corresponds to the mean and standard deviation across participants.

4.2 Gaze detection analysis

The performance of the all the five classifiers was evaluated by calculating the AUC of the ROC curve plotted after calculating the extracted features for each



Correct (T1) Wrong (T2)

Shuffle (T3) Swap (T4)

Fig. 1. The four different types of GUIs.

Table 1. Performance evaluation with the behavioral response.

	TN	FP	FN	TP
GUI T1	10.72 ± 2.27	4.40 ± 1.67	3.12 ± 1.24	1.76 ± 1.10
GUI T1	3.36 ± 1.52	1.08 ± 1.12	1.20 ± 1.26	0.36 ± 0.48
GUI T1	3.00 ± 1.01	1.40 ± 0.98	0.84 ± 0.61	0.76 ± 0.90
GUI T1	4.32 ± 1.12	1.72 ± 1.56	0.84 ± 0.61	1.12 ± 0.86

classification C1 to C10 respectively. Furthermore, the AUC was calculated by removing the records from the experiment which shows the failure of the participants to classify the GUI correctly. As all the classification are performed on the binary labels, AUC for each label in ROC curves was calculated. Furthermore, due to the class imbalance in the data, we used the average AUC of both the classes for analyzing the precision level of the classifiers to classify the GUI accurately. Figure 2 shows the average AUC of the classes for each of one versus rest and one versus one classification case (i.e., C1 to C10) respectively for each classifier where AUC represent the AUC of the ROC curve without removing the incorrect prediction of the users and AUC - Correct represents the AUC of the ROC curve after removing the incorrect prediction of the users.

From the obtained results, the classification C3 (Type 3 GUI vs. all the other type of GUI's) is the most difficult to predict accurately compared to all other classification in one versus rest classification (i.e., C1, C2, and C4). Moreover, for the one versus one classification tasks, we can conclude that classification C10 (Type 3 GUI vs Type 4 GUI) is the most difficult to predict accurately as compared to all other classification in one versus one classification (i.e., C5, C6, C7, C8, and C9) and for both the classifications (C3 and C10) Gaussian Naïve Bayes classifier results the maximum average Area Under the Curve 57% and

70% respectively, when calculated without removing the wrongly predicted data item from the dataset. Similarly, Gaussian Naïve Bayes results the maximum average AUC in classifying congruent GUI versus all the incongruent (C1 classification) GUIs with or without removing the wrongly predicted data item from the dataset i.e., 71% and 76% respectively. Moreover, the average AUC score for the classification C5, C6, and C7 (one versus one classification of Type 1 vs. Type 2, Type 1 vs. Type 3, and Type 1 vs. Type 4) with and without the wrongly predicted values was $66 \pm 17.34\%$ and $73.66 \pm 9.81\%$, respectively.

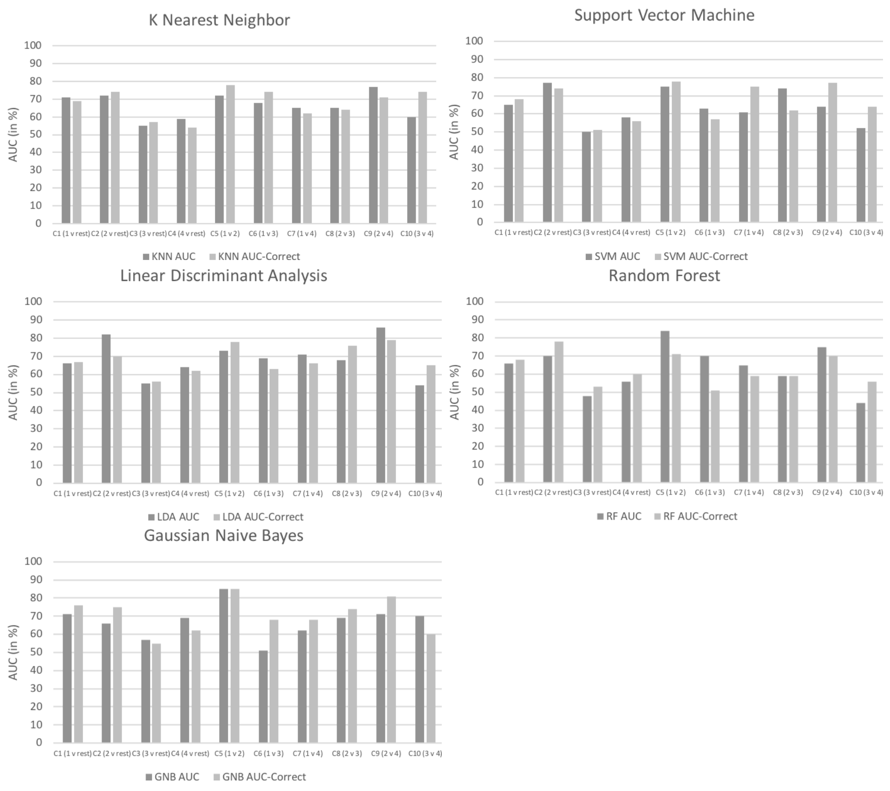


Fig. 2. AUC value of the one versus rest classification (All the results are in %).

4.3 Feature selection

Feature selection is performed using a modified backward elimination technique algorithm for classifications C1 to C10. The algorithm starts with the whole feature set and the feature with the less influence in the AUC is discarded at each step until there is a single feature left in the set to evaluate. The modified feature selection algorithm returns the feature set containing the type of feature and feature influence ratio in term of AUC value. From the feature selection algorithm, we concluded that distance-based feature was having more influence on the classification than centroid-based feature and time-based feature respectively. Features F1, F2, F4, and F7 are the most influential features, while features F3, F8, F9, and F11 are the least influential features for classifying the GUI's as depicted in Figure 3.

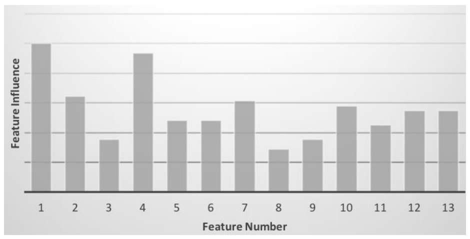


Fig. 3. Feature selection.

5 Discussion

Eyetracking is typically used for control in human-computer interface applications such in virtual keyboards [4, 14], in marketing and medical research, or in vision sciences to better understand cognitive and visual processes. The proposed work aims at going beyond known applications of eyetracking to detect, within a short period, incongruent states on a screen in relation to the relationships between displayed elements. The goal of this approach is to detect potential issues in a graphical user interface that can have an impact on software quality and robustness. By using gaze based features, we have proposed a set of features and their ranking through a feature selection procedure. In this paper, we proposed a set of features for the detection of incongruent states in series of pairs of elements through eyetracking. The increasing importance of software usability with the rise in the number of users and the two approaches that are used in the software industry for developing the user-friendly software. First, a Survey approach and Second, a Computer Vision approach. Moreover, we explained the limitations in those approaches i.e., the first approach is time consuming and highly dependent on the subjective assessment of the user, results in the bias judgment of the GUIs while the second approach for GUI testing increase the workload of the quality assurance tester, furthermore, it requires the big learning curve for writing the efficient testing script and is resolution dependent. Additionally, we proposed an alternative approach inspired from the widely used concept of the early access software in the game industry, i.e., performing the software testing with the help of the users without informing them. We proposed the approach by using an eye tracking technology as it is independent of the user's judgment resulting in bias free observation and is not dependent on the resolution of the system on which the software is used, as well as it does not require any previous knowledge for writing scripts for UI testing. Furthermore, we designed the experiment for classifying the horizontally aligned key value paired GUI into Congruent GUI (GUI with high usability and low learning time) vs incongruent GUI (GUI with low usability and high learning time) by recording the gaze information of the user interacting with the software. Finally, we designed thirteen statistical features and test five supervised learning classifiers for performing the classification. The analysis of the result suggests that Gaussian Naïve Bayes classifier is a suitable classifier for identifying the congruent GUIs based on the proposed feature sets.

6 Conclusion

In this study, we demonstrated the need of better techniques for measuring the software usability and classifying the GUI with high usability to GUI with low usability. Moreover, we exhibited the efficiency of eye tracking technology when it is combined with machine learning techniques for the detection of the anomalies in the GUI with high precision with only information relative to the position of the controls in the layout. By using the proposed machine learning and eye tracking approach, we achieved an average AUC of 76% for classifying congruent vs. incongruent GUI states (C1 classification) using a Gaussian Naïve Bayes classifier under one versus rest classification, after removing the falsely identified values by the users. Furthermore, after removing the falsely identified data items from the dataset, AUC for one versus one classification for identifying congruent GUI states vs. incongruent GUI states, i.e., C5, C6, and C7 using Gaussian Naïve Bayes gives the maximum value with average AUC of $> 73.66(\pm 9.81\%)$ concluding that the Naïve Bayes classifier was more accurate than decision tree based classifier, linear classifier, or density-based classifier for performing eye tracking classification problem for GUI classification. The following three improvements can make the classifier more reliable by increase the precision. First, to gather more training data, which can improve the performance of the classifiers. More data can be acquired for more participants or with longer sessions. Second, to improve feature selection using statistical analysis, more feature analysis can be formed on the recorded gaze so that the outliers in the eye tracker data can be reduced. Third, additional physiological measurements in combination to eye-tracking could be used to enhance the reliability of the decisions, e.g., heart rate, skin conductance. In particular, brain responses associated to visual attention, to the presentation of incongruent elements can be performed and used for training the classifier making it more reliable and accurate in classification. Further work will include other types of interactive controls to extend the ability of the system to detect new GUI states.

References

1. Tobii developer documentation. https://developer.tobii.com/consumer-eye-trackers/core-sdk/, accessed: 07-11-2018

2. Bangor, A., Kortum, P.T., Miller, J.T.: An empirical evaluation of the system usability scale. Int. Journal of Human-Computer Interaction 24(6), 574–594 (2008)
3. Bevan, N.: Usability. In: Encyclopedia of Database Systems, pp. 3247–3251. Springer US (2009)
4. Cecotti, H.: A multimodal gaze-controlled virtual keyboard. IEEE Trans. Human-Machine Syst. 46(4), 601–606 (Aug 2016)
5. Chang, T.H., Yeh, T., Miller, R.C.: Gui testing using computer vision. In: Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems (2010)
6. Dix, A.: Human-computer interaction. In: Encyclopedia of database systems, pp. 1327–1331. Springer US (2009)
7. Fawcett, T.: An introduction to roc analysis. Pattern recognition letters 27(8), 861–874 (2006)
8. Fu, H., Wei, Y., Camastra, F., Arico, P., Sheng, H.: Computational Intelligence and Neuroscience, Advances in Eye Tracking Technology: Theory, Algorithms, and Applications. Hindawi Publishing Corporation (2016)
9. Fu, J.: Usability evaluation of software store based on eye-tracking technology. In: Proc. of the IEEE Information Technology, Networking, Electronic and Automation Control Conference. pp. 1116–1119 (2016)
10. Hansen, D.W., Ji, Q.: In the eye of the beholder: A survey of models for eyes and gaze. IEEE Trans. Pattern Anal. Mach. Intell. 32(3), 478–500 (Mar 2010)
11. Hierons, R.M.: Oracles for distributed testing. IEEE Trans. on Software Engineering 38(3), 629–641 (May/Jun 2012)
12. Jacob, R.J.K.: The use of eye movements in human-computer interaction techniques: What you look at is what you get. ACM Transactions on Information Systems 9(3), 152–169 (Apr 1991)
13. Lupu, R., Bozomitu, R., Ungureanu, F., Cehan, V.: Eye tracking based communication system for patient with major neuro-locomotor disabilities. In: Proc. IEEE 15th ICSTCC. pp. 1–5 (Oct 2011)
14. Meena, Y.K., Cecotti, H., Wong-Lin, K., Prasad, G.: A novel multimodal gaze-controlled hindi virtual keyboard for disabled users. In: Proc. of the Annual Conf of IEEE Sys., Men, and Cybernetics. pp. 1–6 (2016)
15. Meena, Y.K., Cecotti, H., Wong-Lin, K., Dutta, A., Prasad, G.: Toward optimization of gaze-controlled humancomputer interaction: Application to hindi virtual keyboard for stroke patients. IEEE Trans. Neural Sys. and Rehabilitation Eng. 26(4), 911–922 (Apr 2018)
16. Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems 14, pp. 841–848. MIT Press (2002)
17. Nielsen, J.: Usability Engineering. Morgan Kaufmann (1993)
18. Rish, I.: An empirical study of the naive bayes classifier. Tech. rep. (2001)
19. Seffah, A., Donyaee, M., Kline, R.B., Padda, H.K.: Usability measurement and metrics: A consolidated model. Software Quality Journal 14(2), 159–178 (2006)
20. Zhou, Z.Q., Xiang, S., Chen, T.Y.: Metamorphic testing for software quality assessment: A study of search engines. IEEE Trans. on Software Engineering 42(3), 264–284 (Mar 2016)