INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 6.078
(Volume 7, Issue 1)
Available online at: https://www.ijariit.com

Automated User Experience Monitor (A.U.X.M)

Prabhath Mannapperuma
d.p.mannapperuma@gmail.com
Sri Lanka Institute of Information Technology, Colombo, Sri Lanka

Munasinghe T. S.
munasinghets93@gmail.com
Sri Lanka Institute of Information Technology, Colombo, Sri Lanka

Wijetunga D.
dihara.w94@gmail.com
Sri Lanka Institute of Information Technology, Colombo, Sri Lanka

Perera M. M. I.
alexvista1234@gmail.com
Sri Lanka Institute of Information Technology, Colombo, Sri Lanka

Jagath Wickramarathne
jagath.w@sliit.lk
Sri Lanka Institute of Information Technology, Colombo, Sri Lanka

ABSTRACT

User experience is a crucial element in any kind of software. If the user cannot easily and satisfactorily use a software, they will not be inclined to use that software. Therefore, testing the user experience of a software is important. Conducting usability evaluations can be a tedious matter however, since there are several steps involved in setting up even a single usability test. This relates more to scheduling meetings between the test user and the expert evaluator that will conduct the test. Although there are several commercial software which allows designers to "relive", or observe how users use their software, there are no such products that allow usability testing to be conducted, with the role of an expert evaluator being fulfilled. This paper describes a novel approach, the Automated User Experience Monitor (AUXM) system, where usability testing can be conducted in the absence of an expert evaluator, thus eliminating the issue of setting up meeting between test users, and expert evaluators. It is a system that allows designers/developers to have their user interfaces tested for usability. The designer of the prototype screens is able to feed their designs to the system described in this paper, and then invite any number of users to test the usability of those designs. While the users perform the tasks provided to them by the designer, their emotional feedback, keystrokes, mouse clicks, and verbal utterances are recorded. The recorded data is used by the system to inform the designers what went wrong, and suggest how to fix the issues in the designs. Results show that such a system is indeed possible to be implemented, and that usability analysis could be performed much easier using the system than having expert evaluators and test users meet in person. In conclusion, if the proposed approach is developed further, and refined, usability testing could change significantly in the software production industry.

Keywords: *Automated User Experience Testing, User Experience, Remote User Testing, Usability Issues*

1. INTRODUCTION

User experience evaluation plays an important role in the delivery of usable software to the end users. If a software is not pleasing to use, the user will simply not use it, whether or not the software is of a high quality. To remedy this, software prototyping is used, where the system to be developed is "mocked up" by using dummy interfaces that can simulate the actual functioning system. This proves invaluable when determining what the best option is when it comes to certain design choices.

Unfortunately, conducting usability evaluations can be quite time consuming, and are sometimes even omitted, due to the sheer time cost incurred. Conducting usability testing for most, if not all, of the user interfaces, drastically affects the release schedules of software products. On the other hand, the omission of usability evaluations, leads to confusing, and misleading design choices on the part of the designers.

According to interviews conducted with several industry professionals, particularly front end developers, it was revealed that the most commonly used tool to measure user-interactions with web pages is Google analytics. Google analytics show the path a user has taken in a website, and how much time they have spent on a particular web page. Front end developers use this information to determine which pages are keeping users' attention more than others. Using this data they make changes to their designs.

Further study revealed a few commercial tools available for user interaction monitoring.

"UsabilityTools" allows designers to watch video playbacks of how users use their websites, and help them determine why users leave their website. "MouseStats" allow designers to see all the mouse movements, scrolls and keystrokes done by users. Heat-maps allow designers to see what parts of a page attracts most attention. "Usabilla" allows users to leave 'notes' on the site, to show the designer what they think of certain design choices. "Verify" allows designers to submit their designs, and select a type of test that matches the design, and have users use those designs to get reports on how successful a design was.

There are a multitude of website usability monitoring tools available. When all the features provided by those products are generalized, it's mostly playback of user actions, reports of how many users were successful in a task, or heat-maps generated using mouse pointers.

None of them can give accurate information regarding a user's emotional reactions, or determine why users had trouble using a web page.

2. VALIDITY OF USABILITY METRICS USED FOR USER EVALUATION

Before a system that analyses user-experience is considered, the usability metrics used in the system must be verified. One option was to conduct a whole new study in order to settle on usability metrics, but fortunately, such studies and researches have been done before.

Composite methods can be used to measure user experience. For example, using the data collected from the user's verbal utterances and the data collected from a heart rate monitor can determine how the user felt about certain comments. [1]

Private camera conversations can be used to determine user experience [1]. Private camera conversations are tests, where the user goes into a private booth and talks to a camera about given topics.

Usability testing could be done through the internet as well. Online user evaluation methods include the following. prEmo, Emofaces, paired comparison, product personality assignment, sentence completion. [1]

A user's "momentary experiences" and emotions can be observed during task completion. [2]

Qualitative data regarding a user's need fulfillment, momentary experiences and emotions can be collected via observation and open ended questions in interviews. [2]

The user's emotions can be determined through verbal and non-verbal measurements [3]. The most recognized nonverbal measurements recognized in academic research are: PrEmo, Self Assessment Maniki, Facial Action Coding System, and Emocards [3]. This means that the user's facial expressions can be used to determine their experience. Therefore, these findings verify that the usability metrics that we utilize to measure the user's experiences.

2. RELATED WORK

An automated online tool to evaluate web site usability based on its domain [4]. This research managed to create a tool that analyses all the components in a web page and compares it against a set of pre-defined guidelines by professional organizations. The tool also suggests possible enhancements to the web interface.

There's a plugin developed for Microsoft Visual Studio IDE to evaluate ASP.NET websites during development [5]. This tool compares a designer's code against a set of guidelines recommended by a set of international organizations that promote web standards.

Another research has delved deep into tracking a user's interaction with a website, through the use of javascript injected into the page itself before it is delivered to the user [6]. This research has made sure that it does not affect the user's experience in any way while the evaluation is in session.

In 2016, a research was conducted to utilize logging tools to capture user interactions with a website, remotely [7]. These log data were presented in a timeline format to support easier usability expert analysis.

A recent research was conducted in which users were required to provide direct responses as to how their experience was when doing a certain task [8]. This was done by presenting a set of emoticons faces to the user and allowing them to choose how their experience was.

3. METHODOLOGY

The AUXM system is comprised of several components, that communicate with each other. There are four major components of the system.

- Context Analyzer
- GPU Accelerated Facial and Speech Emotion Detector
- Problem Deducer
- Solution Provider

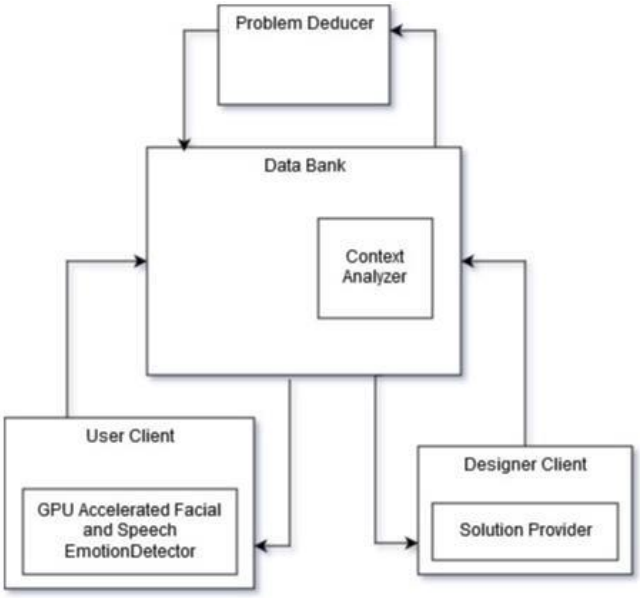

Fig. 1: Components of the AUXM system

Before delving into the details of these components, the functioning of the entire system, as a whole must be described. The process starts with the "Designer Client". The Designer Client is the software created for the developers/UI designers to interact with the AUXM system. The first thing to do with the Designer Client is to feed the user interfaces into the system. The AUXM system only accepts html files. Once these files are fed into the system, the system will then analyze these files, and "describe" each and every interactable element in them. This is termed "Context Analysis", where each element is given a "context". The context analysis occurs just as the files are added to the system. Once the files are added to the system, the designer may link the files, specifying how to reach other files through each file. Once the files are linked, the designer may then provide the "task list" into the system. The task list is the list of tasks a test user would be expected to complete. After that, the designer creates an "action sequence", where they feed the set of actions required to complete each task that's given to the test users to complete. This is done by the developer performing each task, and recording the "correct/expected" set of steps required to complete a certain task. Once all of these steps are completed, the designer may then invite specific users to test the user interfaces fed into the system. The designer may specify which

users should receive the invitation by using their email addresses, or by selecting them from a list of preregistered users. The designer may even publish the interfaces to the "public domain", where any registered user can choose to take part in the testing.

In the case of inviting a user that is not already registered in the system, an email will be sent to them. The email will contain a link to download the necessary software (The "user client"), and a link to the registration page of AUXM. The test user may then click a link provided with the email, to download the "user client", which is a software created to allow testing, and recording the interactions of users. Once the software is installed, the user may then log in, using their respective credentials. After logging in, the user will then see the system to be tested listed down in a menu. The user can select the required project, and then they would be provided with a list of tasks that they are expected to complete. Before the user proceeds, they will be given the options to adjust their inputs to optimal levels (web cam feed, audio, lighting, etc.). The user may then proceed and attempt to complete all the tasks provided. The system will record all of the user's interactions during those tasks, and will encourage the user to implement the think aloud protocol if they get stuck. The system will also provide hints as to how to proceed, if the user gets stuck for too long. Once all the tasks are completed, with, or without issues, all the recorded user interaction data will be uploaded into the system. This concludes the testing, and the tester can choose to keep the software in their machines, if they intend to partake in future tests.

Once a test has been concluded, the client application will generate the transcript for the session using the recorded audio and script deviations and input events. Afterwards the transcript, as well as the recorded audio, video, emotion and input data, will be compressed into a single package in order to reduce file size. The video and audio recordings are not uploaded to the system, only the data extracted from them are stored. The resulting compressed archive will be uploaded to the server. Once uploaded, it'll be analyzed, and the system will attempt to recognize what kind of design fault caused the user to get stuck, or do the tasks incorrectly. After associating a possible cause for a flaw, and pin pointing the violating element, the system will then pass the findings to the "solution provider" component which will use the retrieved data to provide possible solutions to fix the recognized design fault.

The system will then analyze the possible causes for the design faults, and access the system's databases to find user interfaces that have similar elements, but without any usability faults in them. These are termed "good/valid designs". These valid designs should be fed into the system separately via qualified individuals. To find similar elements the "contexts" generated, just after feeding the user interfaces into the system, will be used. Once similar elements are found, the system will provide valid designs that can be applied to the current violating interface, to the designer. This will be done via the "designer client". Once the designer is provided with the list of applicable fixes, they may select one. The system will then change the violating element to resemble the one in the suggested fix, and show the designer how it would look like, when it is fixed. If the designer is satisfied, they may save the changes to the file. If not, the designer could just observe how the fix would affect the element, and choose to manually change the interface.
Once the designer applies the suggested fix to their design, they may resubmit the pages to be tested again, and the process will begin once again. This however would be the "next iteration" of the test. The designers can go through any number of iterations, fixing recognized faults and changing the design however they fit, until they settle with a design that have no design faults, or at least, an acceptable level of usability. The four major components of the AUXM system will be described in detail below.

A. Context Analyzer

The AUXM system gives solutions to fix certain elements that have design faults in them. This is done by finding designs with elements similar to the elements that have design faults in them. There is no conventional way to find "similar" elements across different web pages. Even if two similar web pages were to be compared, the elements within them could be in different places, or added in different ways into the web page. There is no "one correct" way to create a web page, and as such, web developers need not follow a single procedure and structure when creating web pages [9]. Due to this, pin pointing elements of similar nature becomes problematic. Similar elements do need to be recognized in order for the AUXM system to provide solutions to fix the faults in an element that has design flaws in it.

The context analyzer uses the neighboring elements of the target element to describe it (to give it a "context"). A similar research was done in 2003 [10], where an entire web page was described using certain html tags in the page. The context analyzer works similarly, but instead of describing the entire page, the procedure is used to describe individual elements of the web page. The context analyzer primarily focuses on the terms appearing around an element. A research was done in 2012 , where a cluster of web pages was described using the terms appearing in the web pages. The context analyzer acts similarly, but uses the terms, not to describe a cluster of web pages, but to describe a single html element.

```
Get html file

While html file has html tags

    if the html tag is a <a>, <input>, <button>
        add html tag to list
    else
        discard html tag
    move onto next html tag


While html tag list has html tags
    While attribute list has attributes
        if html tag has attribute
            extract attribute content
            if attribute content is meaningful
                add attribute content to "context term" list
            else
                discard attribute content
        else
            check next attribute

    While parent tag of current html tag is not <body>
        Get parent of current html tag
        Get first child of parent tag

        While current html tag is not first child
            if current html tag is <p>, <h*>, <label>
                extract innerHtml of current element
                if innerHtml of current element is meaningful
                    add innerHtml of current element to list
                else
                    discard innerHtml of current element
            Get previous sibling of current html element
        Make the parent tag the "current html tag"
    move onto next html tag

Save "context term" list
```

Fig. 2: Pseudo code of context analyzer

When a html file is provided into the system, the context analyzer parses through the entire file, and extracts a set of elements. These elements include <a>, <input>, and <button> tags. Then, each of these extracted tags are analyzed

individually, extracting the attributes of the tags such as the name, id, placeholder, value, and innerHtml. Not all the elements have all of these attributes, but what is present is extracted. These extracted attributes are analyzed to see if the words have meaning. A developer can choose to give a randomly generated Id to an html element as opposed to a meaningful name. For example, "xyz" as opposed to "fname". If the textual information is not meaningful, the attribute extracted is discarded. A key word recognition library was used to check for important words, and from those words, only the meaningful words are considered. This allows the recognition of important words in a text, and to recognize words that really have a meaning, which can be used to describe an html element.

Once the attributes are analyzed, the context analyzer will then move outside of the html element, and seek out elements around the target element that can be used to describe it. The neighboring elements considered are <p>, <h*>, <label>, <font>, and normal text. The textual information retrieved by these are again analyzed in the same way as the attributes, to determine if the data is with or without meaning. The context analyzer will continue to seek out elements till it reaches the <body> tag, or another interactable element (<input>, <a>, or <button>), extracting the textual information of all the elements that it analyzes.

All of this textual information gathered from analyzing neighboring elements, and the target element itself, is then made into a list of terms and phrases. This list of words and phrases is what describes the element, i.e – the context of the element. When finding "similar" elements, during the solution generation of the AUXM system, this list of terms (context of an element), is matched against other lists, to find the closest resemblance. If two contexts (lists of describing words) are similar to a certain extent, the web pages containing those elements will be selected to be offered as possible solutions for the designer of the user interface.

B. GPU Accelerated Facial and Speech Emotion Detector

The emotion detector component of the AUXM system test client consists of a machine learning classifier that has been trained to detect human emotions. Unlike earlier implementations [22] [23] [24], it is entirely comprised of convolutional neural networks (CNN). This classifier considers both facial and speech emotion, thus it consists of two separate classifiers internally. For increased performance of the testing and classification processes, GPU acceleration is utilized.

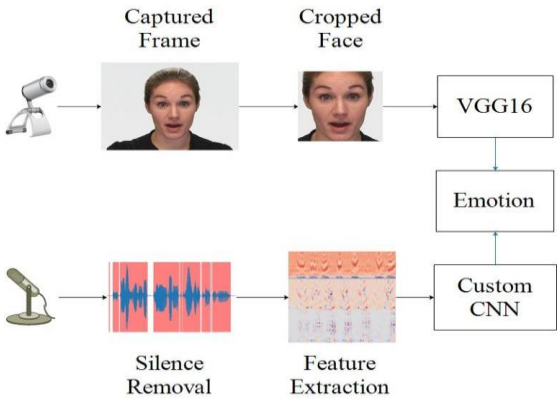

Fig. 3: High-level flow of the emotion detector

When provided with a video and audio stream, this component classifies the input data into one of six emotions (neutral, happiness, anger, fear, surprise, sadness) by extracting the facial and speech emotion and performing a weighted average of the two using the error of each classifier (Figure 3). Both facial and speech emotion subsystems utilize convolutional neural networks as classifiers. A pre-trained VGG16 model [26] was used for facial emotion while a smaller, custom model was used for Speech emotion classification.

The system is trained using multiple audio-visual emotion databases: The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) and the Surrey Audio-Visual Expressed Emotion (SAVEE). Since these databases are videos, frames are extracted at a regular predefined interval for training the facial emotion classifier. Afterwards, all visible faces within the are cropped into individual images using the Haar-cascades classifier provided by OpenCV. These cropped images are then resized into 224x224 pixels. For audio data, the initial step consists of silence removal. Afterwards, the clip is split into frames and for each frame, the first 12 Mel-Frequency Cepstral Coefficients (MFCC) are extracted as it has been used for speech recognition related tasks with high rates of success [27]. The first and second time derivatives of these 12 coefficients are also computed, resulting in 36 features for each frame. Features from 36 of these frames are combined to create a single image of dimensions 36x36 as input to the CNN.

All CNN architectures are implemented using Caffe: a well-known open source deep learning framework developed at UC Berkley [25]. Since the AUXM system is a consumer application, it must run across a wide range of CPU/GPU hardware. Caffe offers GPU acceleration through Nvidia's CUDA framework. Since this is an Nvidia-specific feature, we have resorted to using the experimental branch of Caffe which uses OpenCL as its' GPU compute backend. In case the primary GPU of the system is weak, the entire classification process can be performed on the CPU.

C. Problem Deducer

This component mainly consists of two sub components. They are the transcript analyzer and intelligent problem deducer. Transcript is being generated by the user client application. Usability problem analyzer will extract keywords using natural language processing techniques. Then a sentimental analysis will be conducted. Apart from these data, front end application is also sending emotion data of the user using video and audio streams. When this system identified an unfavorable user action, it will retrieve required data related to that particular time period and will process them as mentioned above. At the end system will return identified/possible causes for that particular unfavorable user action.

D. Solution Provider

Solution Provider is the component that will generate the solutions by looking at the good designs that provided by the context analyzer. In here it will only change the color and sizes of HTML elements because It's not a good method to replace the HTML element in the document, it can break the original design of the HTML document or the context of the element that is taken from the good design can be different to the original element (problematic element).

It uses two methods to generate a solution. In first method it just extracts color and the size CSS attributes of the element by using XPATH and parsing CSS of the good design and then apply them to the original element.

In the second method it will get the background color of the element by using Image processing techniques. Then it will calculate the Delta-E [11] (Distance of the colors) metric using dE2000 [12] equation that is consider the human eye's attributes.

4. EXPERIMENTAL RESULTS AND DISCUSSION

The research produced a usable, functional product, that allows remote usability testing, and evaluations to be conducted, without a human expert evaluator. The outputs and outcomes of each research component is discussed in this section as well.

The Context Analyzer describes html elements in an efficient, and usable manner. It takes the meaningful text content from an element's attributes and from the neighboring elements. Figure 4 shows a sample output for a "First name" field of a web page. When querying the generated contexts, only the web pages with similar, relevant fields are returned. The querying mechanism was created with the ability to take in a desired accuracy as well.

In the emotion recognition system, the facial expression component offers an average test accuracy of 98.2%. This high accuracy is attributed to the use of transfer learning where a pre-trained model is fine-tuned to work with a different database. The speech emotion classifier displays an average test accuracy of 67%. The drop in accuracy is due to the low number of training data and lack of pre-trained models. The system benefits heavily from the use of GPU acceleration where the average prediction time of the facial expression classifier in CPU mode was brought from 350 milliseconds down to a mere 16 milliseconds on an Nvidia GeForce GTX 1070 GPU. Speech classification however does not benefit from this as the cost of transferring data from between the CPU and GPU is higher than the actual computation time since the model is relatively small.


```
"contextTerms" : [
    {
        "termId" : "1",
        "term" : "input",
        "weight" : 10
    },
    {
        "termId" : "2",
        "term" : "text",
        "weight" : 8
    },
    {
        "termId" : "3",
        "term" : "First Name...",
        "weight" : 8
    },
    {
        "termId" : "4",
        "term" : "First name:",
        "weight" : 6
    }
],
```


Fig. 4: sample context for "first name" field

Once a problematic design sent to the solution provider it will generate solutions using the methods we described above. Figure 5 is an example for a problematic

design. It's an anchor tag which is linked to the home page and the form's submit button's text is not visible to the user, those are the issues with this page. This page will be sent to the solution provider with the relevant XPATH and its cause details. Figure 6 shows the solution's that generated by the solution provider. Even context analyzer didn't send a good design to refer, solution provider still can generate a solution by looking at the background color of the element.

Fig. 5: Problematic Design

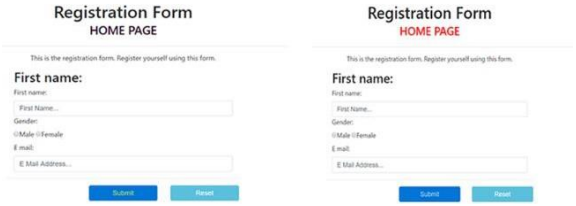

Fig. 6: Results from the Solution Provider

5. CONCLUSION

The AUXM system proves that remote user experience monitoring is possible, and that usability testing can be done without having an expert evaluator meet individual users, which is a time consuming and tedious task for all parties involved. The AUXM system theoretically allows an infinite number of users to test a set of user interfaces, all at the same time. The designer need only submit their designs to the system, it is that simple. Due to the ease at which usability testing can be done, the AUXM system has the potential to be used by software production companies to conduct any amount of user experience testing. This would give rise to other software similar to the AUXM system, which would change how usability testing is conducted in the industry today. Furthermore, because the AUXM system introduces such an easy, convenient means of conducting usability analysis, there will be a smaller chance of software reaching the market that have illogical, or unusable user interfaces.

6. REFERENCES

[1] A. P. O. S. Vermeeren, V. R. Effie Lai-Chong Law, M. Obrist, J. Hoonhout and K. Väänänen-Vainio-Mattila, "User experience evaluation methods: current state and development needs," in NordiCHI '10 Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries, Reykjavik, Iceland, 2010.

[2] K. S. Krömker and Heidi, "A framework to measure user experience of interactive online products," in MB '10 Proceedings of the 7th International Conference on Methods and Techniques in Behavioral Research, Eindhoven, The Netherlands, 2010.

[3] Meyer, A. Agarwal and Andrew, "Beyond usability: evaluating emotional response as an integral part of the user experience," in CHI EA '09 CHI '09 Extended Abstracts on Human Factors in Computing Systems, Boston, MA, USA, 2009.

[4] S. A. U.S.Samaratunge, K. A. H. Madusanka, M. Devanarayana, W. C. M. D. ThimiraSachintha and H. E. Wickramasinghe, "UIX Pro Enhancer Smart UX Enhance Agent for Website Evaluation".

[5] A. M. Sherihan, K. K. Gunewardana, M. N. A. Ahmed, R. J. Pereira, C. J. Wickramarathne and M. A. S. C. Manamendra, "Evaluation Tool for Microsoft ASP.NET Web Applications in Compliance with Usability Guidelines.".

[6] R. Atterer, M. Wnuk and A. Schmidt, "Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction," in WWW '06 Proceedings of the 15th international conference on World Wide Web, Edinburgh, Scotland, 2006.

[7] F. Paternò, A. G. Schiavone and P. Pitardi, "Timelines for Mobile Web Usability Evaluation," in AVI '16 Proceedings of the International Working Conference on Advanced Visual Interfaces, Bari, Italy, 2016.

[8] Mezhoudi and Nesrine, "User interface adaptation based on user feedback and machine learning," in IUI '13 Companion Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion, Santa Monica, California, USA, 2013.

[9] Houqing Lu, Donghui Zhan, Lei Zhou and Dengchao He, "An Improved Focused Crawler: Using Web Page Classification and Link Priority Evaluation," Mathematical Problems in Engineering, vol. 2016, no. Mathematical Problems in Engineering, p. 10, 2016.

[10] E. Uzun, H. V. Agun and T. Yerlikaya, "A hybrid approach for extracting informative content from web pages," Information Processing and Management: an International Journal, vol. 49, no. 4, pp. 928-944, 2013.

[11] CHROMiX ColorNews, "Delta E: The Color Difference," 18 Feb 2005. [Online]. Available:http://www.colorwiki.com/wiki/Delta_E:_The_Color_D ifference.

[12] Yang Yang, Jun Ming and Nenghai Yu, "Color Image Quality Assessment Based on CIEDE2000," 23 April 2012. [Online]. Available: dl.acm.org/citation.cfm?id=2434237.

[13] Arroyo, T. Selker and W. Wei, "Usability tool for analysis of web designs using mouse tracks," in CHI EA '06 CHI '06 Extended Abstracts on Human Factors in Computing Systems, Montréal, Québec, Canada, 2006.

[14] PEMBE and T. GÜNGÖR, "A tree-based learning approach for document structure analysis and its application to web search," Natural Language Engineering, vol. 21, no. 4, pp. 569-605, 2015.

[15] Y. Xue, Y. Hu, G. Xin, R. Song, S. Shi, Y. Cao, C.-Y. Lin and H. Li, "Web page title extraction and its application," Information Processing and Management: an International Journal, vol. 43, no. 5, pp. 1332-1347, 2007.

[16] T. C. Craven, "HTML Tags as Extraction Cues for Web Page Description Construction," InformingSciJ, vol. 6, pp. 1-12, 2003.

[17] E. Glover, D. M. Pennock, S. Lawrence and R. Krovetz, "Inferring hierarchical descriptions," in CIKM '02 Proceedings of the eleventh international conference on Information and knowledge management, McLean, Virginia, USA, 2002.

[18] E. J. Glover, K. Tsioutsiouliklis, S. Lawrence, D. M. Pennock and G. W. Flake., "Using web structure for classifying and describing web pages," in WWW '02 Proceedings of the 11th international conference onWorld Wide Web, Honolulu, Hawaii, USA, 2002.

[19] G. A. Sebastiani and A. G. a. Fabrizio, "Automatic Web Page Categorization by Link and Context Analysis," 1999.

[20] W. Mokrzycki and M. Tatol, "Color difference Delta E - A survey," Machine Graphics and Vision, vol. 20, no. 4, pp. 383-411, 2011.

[21] F. Archetti, G. Arosio, E. Fersini and V. Messina, "Audio-based Emotion Recognition for Advanced Automatic Retrieval in Judicial Domain," 2008.

[22] J. Cheng, Y. Deng, H. Meng and Z. Wang, "A facial expression based continuous emotional state monitoring system with GPU acceleration," in 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, 2013.

[23] D. Duncan, G. Shine and C. English, "Facial Emotion Recognition in Real Time," 2013.

[24] S. Emerich, E. Lupu and A. Apatean, "Emotions recognition by speech and facial expressions analysis," in Signal Processing Conference, Glasgow, UK, 2009.

[25] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," in arXiv:1408.5093, 2014.

[26] O. Parkhi, A. Vedaldi and A. Zisserman, "Deep Face Recognition," in British Machine Vision Conference, 2015.

[27] A. Watile, V. Alagdeve and S. Jain, "Emotion Recognition in Speech by MFCC and SVM," International Journal of Science, Engineering and Technology Research (IJSETR), vol. 6, no. 3, 2017.

