A Method for Automatically Estimating Areas of Interest Boundaries for Text Areas

Ezekiel Adriel D. Lagmay
Ateneo de Manila University
Quezon City, Metro Manila, Philippines
ezekiel.lagmay@yahoo.com

Ma. Mercedes T. Rodrigo
Ateneo de Manila University
Quezon City, Metro Manila, Philippines
mrodrigo@ateneo.edu

ABSTRACT

This paper explores the use of Histogram Equalization as a step in estimating Areas of Interest (AOI) boundaries for text areas on desktop screenshots as part of a larger study on automatic AOI discovery and tracking of objects in dynamic eye-tracking stimuli. Results show that Histogram Equalization is effective for text on a light background, but it is not able to discriminate non-text elements and does not even work for text on a dark background.

CCS CONCEPTS

• **Human-centered computing** → **Natural language interfaces**; *Command line interfaces*; *Graphical user interfaces*; • **Computing methodologies** → **Interest point and salient region detections**; *Shape representations*; *Image segmentation*; *Video segmentation*; *Shape inference*; • **Applied computing** → Computer-assisted instruction; Interactive learning environments;

KEYWORDS

Areas of Interests (AOI), Dynamic Stimuli, Eye-Tracking, Histogram Equalization

ACM Reference Format:
Ezekiel Adriel D. Lagmay and Ma. Mercedes T. Rodrigo. 2019. A Method for Automatically Estimating Areas of Interest Boundaries for Text Areas. In *Proceedings of Information and Computing Education Conference (ICE2019).* Davao City, Philippines, 6 pages.

1 INTRODUCTION

1.1 Context of the Study

Object Detection "deals with detecting instances of semantically meaningful objects of certain classes, such as humans, buildings, or cars in digital images and videos" [5]. Object Detection has already been used in surveillance, transportation systems, driver assistance, smart rooms, and visual data summarizing, but the main challenge to achieving accurate object detection is the handling of complex data found in images and videos (such as pose, lighting, and clutter) [5].

Since 2015, the Ateneo Laboratory for the Learning Sciences (ALLS) has been conducting numerous experiments and research on how novices and experts differ when it comes to code reading in relation to debugging. This kind of research has been made possible thanks to the introduction of eye-tracking methodology which enables the computer to capture a user's eye movements as it moves across the screen/computer monitor. So far, the analysis efforts in this area of research involving eye-tracking and the debugging and program comprehension behaviors of novice and expert programmers focused mainly with static stimuli, but not all participants were asked to point out bugs by just looking at static pictures/PDF files depicting the buggy codes. Sometimes, the participants were given a text editor containing those codes and were allowed to interact with it (either by adding or deleting code, thereby changing the code depicted itself). Thus, when analyzing eye-tracking data involving dynamic stimuli, the main hurdle would be to define dynamic Areas of Interests (AOIs), which may change in shape, form, or even location. In addition, for dynamic stimuli, manually-defined AOIs would be very tedious, since a video comprises of frames which are so numerous that it would take a very long time to define AOIs manually.

This paper focuses on one aspect of this problem: Automatically Estimating the AOIs of Text Areas in Eye-Tracking Stimuli. This can be achieved using an image processing technique known as Histogram Equalization, which shows some interesting properties when applied to any stimuli with text on it.

1.2 Research Questions

The research questions are as follows:

(1) How can we automatically track AOIs which is able to adapt to the constant changing of the visible computer environment in the video, yet also flexible enough to suit the needs of potential future researchers in the field of dynamic eye-tracking?
(2) How can text in videos be taken account in the automatic detection and tracking of AOIs? (this is considered a special case).

1.3 Research Objectives

The research objectives are as follows:

- To detect and keep track of AOIs automatically for each group of scenes which takes into consideration the presence of text.

1.4 Significance of the Study

The technique for estimating Areas of Interests for text areas shown in this paper is part of a potential solution for a larger problem on Eye-tracking analysis involving Dynamic AOI. As a side note, this is ALLS's first attempt at analyzing dynamic stimuli datasets. The techniques that will be presented in this research will be helpful for the ongoing research on debugging behavior of novice and

ICE2019, October 2019, Davao City, Philippines

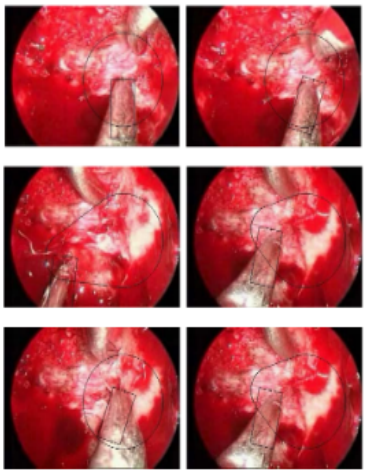

Figure 1: Result of ROI Detection with Updates [6]

expert programmers, and other researchers wishing to delve into eye-tracking studies on video stimuli, most especially the tracking of text.

1.5 Scope and Limitations

This study will use video output data (in .avi format) available from ALLS's eye-tracking experiment archive for the purpose of testing, specifically data obtained from participants coming from University A who used the Java Programming Language, as Java is a common starter programming language for both novice and expert programmers alike (ALLS also collected dynamic eye-tracking data from students of University B, but their video representations were missing). In addition, the study will focus on some of the frames of those video outputs, in .png format.

2 REVIEW OF RELATED LITERATURE

2.1 Automatic AOI Detection for Eye-Tracking

In the paper by Bing Liu, et. al. entitled "Automatic Detection of Region of Interest Based on Object Tracking in Neurosurgical Video", the idea for tracking the user-defined ROI in a Neurosurgical Video (which includes the tip of the instrument and its surrounding tissues) is to use a rectangle to define ROI borders and the edge information of the surgery instrument is used to adjust the tracked position [6]. The overall procedure makes use of Mean Shift Analysis for data cluster and object tracking, wherein the instrument is represented in a featured space by its estimated Probability Density Function [6]. Whenever the instrument changes orientation, the frame's edge information is used to update the tracking position [6]. Lastly, the Tangent Line Problem is used for smoothly "morphing" ROI [6]. The final product of ROI detection with tracking is given in Figure 1 [6].

The main limitation of this method, however, is that ROI detection is done manually at the beginning of the algorithm (only the ROI tracking and update portions are automatic), and hence will not be efficient for more objects [6]. Huang and Li attempted to solve this issue by looking into the interesting properties of Active

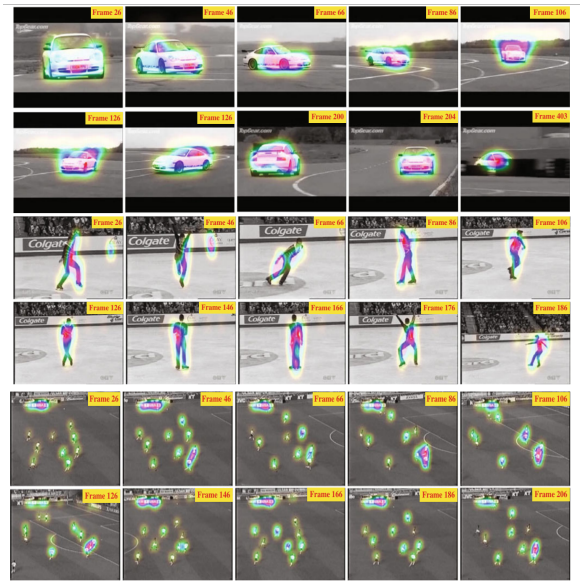

Figure 2: Results of the Proposed Automatic AOI Detection System for Active Video [5]

Video which loosely resembles human eye movements in that it follows the Object(s) of Interest (OOIs) by means of panning, tracking (or dolly), or revolving (or tilting), which is very reminiscent of saccades [5]. These important properties thus help eliminate the need for manually defining the initial AOIs by providing actual cues for video segmentation and enable OOI tracking using the object-centered property manifested by Active Video [5].

The procedure is divided into two parts [5]:

(1) OOI Detection Step (first few frames) - Use Color-Saliency Weighted Probability-of-Boundary (cPoB) confidence map to select OOI Candidate Keypoints (Positive Pool), with the other keypoints in the Negative Pool [5].
(2) Successive Classification and Refinement (all other frames) - Train the strong classifier with AdaBoost using the Positive and Negative Datasets then refine/maximize the similarity between two sets of keypoints using LP Feature Matching (done again after a number of frames) [5]. Also, Positive and Negative Datasets are continuously collected while also discarding older ones to track appearance changes in OOI [5]. Lastly, Global Motion Compensation (GM) is also used for "cleaning remaining keypoints which are moving along the same direction of the global motion vector" [5].

The results are shown in Figure 2 [5].

Unfortunately, this approach has some disadvantages in that it might not work for desktop environments which are not active videos but rather passive videos (with static backgrounds and multiple objects appearing, disappearing, and moving over it) [5]. Also, the sample videos used are of only one "scene" type [5].

Figure 3: Results of the Proposed K-Means Image Segmentation Algorithm (right column) as compared to the original (left column) and traditional K-Means Image Segmentation (middle column) [3]

2.2 Deep Learning Algorithms for Image and Video Analysis

Related to the topic of automatically extracting and detecting AOIs from images and videos is the concept of Image/Video Segmentation, which involves grouping the pixels of an Image or Video Frame into different clusters according to characteristics such as color and texture. This essentially simplifies the image by showing the relevant objects shown in it and their respective areas and boundaries.

One such algorithm, based on the K-Means Clustering Algorithm, is described by Dhanachandra, et. al. [3]. The algorithm is comprised of the following steps [3]:

(1) Partial Contrast Stretching on the image and the initial number of clusters is initialized [3].
(2) Use Subtractive Clustering Algorithm to get initial K-Means centroids by maximum pixel potential [3].
(3) Euclidean Distance of each centroid from each pixel in the image is calculated and assign pixel to centroid with minimum distance [3].
(4) Get new centers, repeat algorithm until clusters satisfy a tolerance value [3].
(5) Reshape cluster pixels into the image [3].
(6) Median Filter is used to remove potential noises in the segmented image [3].

The results are shown in Figure 3 [3].

An alternative image segmentation algorithm is based on the DP Clustering Algorithm which "...Determine the cluster numbers and cluster centers based on the decision graph, on the representation of the input image in color feature space" [2]. This is based on the notion that "cluster centers are often surrounded by points which has lower density and have a relatively large distance from these points with higher density" [2]. The procedure first involves converting the image into feature spaces using color channel feature, and then DP Algorithm is applied on the resulting representations, with Euclidean Distance used to measure distances between all input data, and Gaussian Kernel as the basic measurement of a point's density [2]. The results are shown in Figure 4 [2].

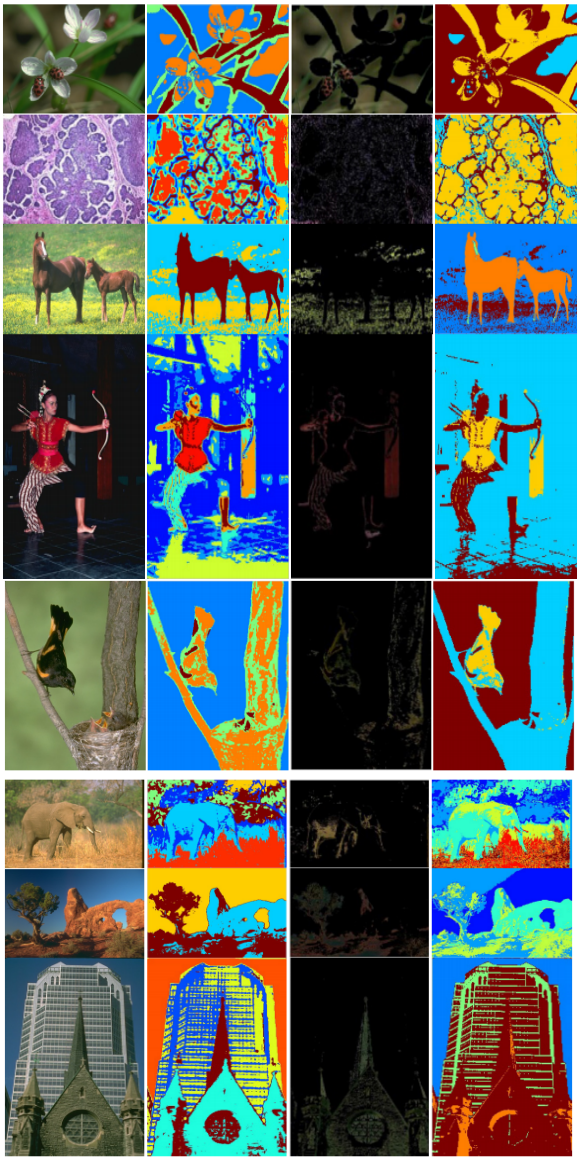

Figure 4: Results of the Density-Distance based Image Segmentation Algorithm (far-right column) as compared to the original (far-left column), K-Means Image Segmentation (center-left column), and FCM (center-right column) [2]

3 METHODOLOGY

3.1 Dataset Description and Collection Methods

The dataset used for this study were obtained as part of a larger experiment by ALLS in utilizing eye-tracking to discover differences between novice and expert programmers when it comes to program comprehension and code debugging. In particular, this Thesis will be using the video output component of the Individual Dynamic Data collected from 9 participants from University A. The eye-trackers used in the study were made by Gazepoint, with a

Figure 5: Eye-Tracking Experiment Set-Up Used by ALLS [9]

sample rate of 60Hz and an accuracy of 0.5-1 degree [9]. The screen resolution of the monitors used in the experiment was 1024x768, and, by default, the problems were shown in full screen [9]. There were a total of 12 debugging problems shown to the participants, with the first three having a single bug while all others have three bugs [9]. For each problem, the participants were asked to read the given problem (in PDF form) and walk through its corresponding Java program code (which was shown in the Notepad++ application) [9]. The participants were free to make changes to the original program (adding code, deleting code, etc.) and they have access to a Java compiler via CMD. A typical experiment set-up used is shown in Figure 5 [9].

3.2 Histogram Equalization for Estimating AOIs for Text Areas

Histogram Equalization is a "computer image processing technique used to improve contrast in images by spreading out the most frequent intensity values (or stretching out the intensity range of the image)" [8].

As a prerequisite to the Histogram Equalization procedure, consider an image with r denoting its continuous intensity values such that $0 \leq r \leq L-1$ where 0 means black and $L-1$ means white [1, 4]. Then, the transformation (intensity mappings) of r is given by [1, 4]:

$$s = T(r); 0 \leq r \leq L-1 \tag{1}$$

Where s is the output intensity level for every pixel in the input image with intensity r [1, 4]. The following assumptions are then made [1, 4]:

- (1) $T(r)$ is a monotonically increasing function within the interval $0 \leq r \leq L-1$ (guarantees that output intensity values are never less than corresponding input values) [1, 4].
- (2) $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$ (guarantees that the range of output intensities is the same as the input) [1, 4].

Also, if there exists an inverse function $r = T^{-1}(s); 0 \leq s \leq L-1$, (1) can be rephrased into the following statement [1, 4]:

- (1a) $T(r)$ is a *strictly* monotonically increasing function within the interval $0 \leq r \leq L-1$ (guarantees that the mappings from s back to r is one-to-one, preventing ambiguities) [1, 4].

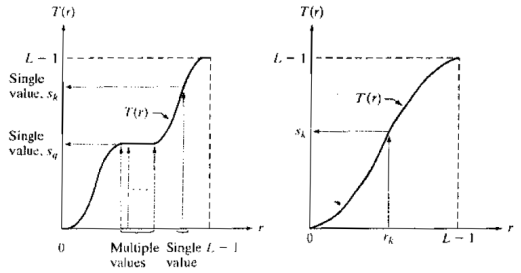

Figure 6: Monotonically Increasing Function (left) vs. Strictly Monotonically Increasing Function (right) [1, 4]

These assumptions can be illustrated with Figure 6 [1, 4]. A monotonically increasing (transformation) function (like the left graph in Figure 6) satisfies (1) and (2) since it performs a one-to-one or many-to-one mapping from r to s, but it is not possible to recover the values of r from the mapped values uniquely [1, 4]. On the other hand, a *strictly* monotonically increasing (transformation) function (like the right graph in Figure 6) will guarantee that the inverse mappings will be single-valued (one-to-one in both directions) [1, 4]. Since the actual Histogram Equalization process involves integer intensity values in practice (hence the need to round all results to their nearest integer values), the requirement of strict monotonicity is important since the problem of nonunique inverse transformation by looking for the closest integer matches might arise when the requirement is not satisfied [1, 4].

The intensity levels in an image may be viewed as random variables in the interval $[0, L-1]$ [1, 4]. As such, the Probability Density Function (PDF) can be used as a fundamental descriptor of a random variable [1, 4]. Assuming $p_r(r)$ and $p_s(s)$ are the PDFs of r and s, respectively, and that $p_r(r)$ and $p_s(s)$ are different functions (in general), then, by the basic probability theory, if $p_r(r)$ and $T(r)$ are known, and $T(r)$ is continuous and differentiable over the range of values of interest, then $p_s(s)$ can be obtained using the following formula [1, 4]:

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| \tag{2}$$

In other words, the PDF of the output intensity variable s is determined by the PDF of the input intensities and the transformation function used [1, 4]. One transformation function of importance in image processing is given by [1, 4]:

$$s = T(r) = (L-1)\int_0^r p_r(w)dw \tag{3}$$

Where w is a dummy variable of integration [1, 4]. The right side of the equation is the Cumulative Distribution Function (CDF) of random variable r [1, 4]. Since PDFs are always positive, and that the integral of a function is equivalent to the area under the function, the formula satisfies both (1) (since the area under the function cannot decrease as r increases) and (2) (since the integral evaluates to 1 when the upper limit of the equation is $r = (L-1)$, and so the maximum value of s is $(L-1)$) [1, 4]. Equation 2 is then

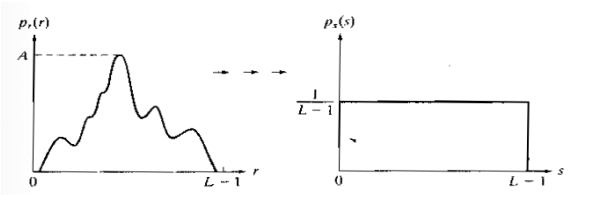

Figure 7: A graph of an arbitrary PDF (left) vs. a uniform PDF (right) [1, 4]

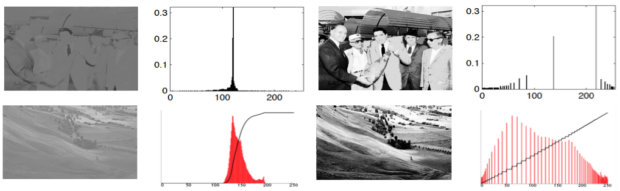

Figure 8: Examples of Images Processed with Histogram Equalization (From left to right column: Original Image, Original Histogram, Transformed Image, and Transformed Histogram) [1, 4, 8]

used to find $p_s(s)$ corresponding to the earlier transformation [1, 4]. By Leibniz's Rule [1, 4]:

$$\begin{aligned}\frac{ds}{dr} &= \frac{dT(r)}{dr} \\ &= (L-1)\frac{d}{dr}\left[\int_0^r p_r(w)dw\right] \\ &= (L-1)p_r(r)\end{aligned} \tag{4}$$

The result is then substituted for $\frac{dr}{ds}$ in Equation 2, then assuming that all probability values are positive, the outcome is [1, 4]:

$$\begin{aligned}p_s(s) &= p_r(r)\left|\frac{dr}{ds}\right| \\ &= p_r(r)\left|\frac{1}{(L-1)p_r(r)}\right| \\ &= \frac{1}{L-1}; 0 \le s \le L-1\end{aligned} \tag{5}$$

$p_s(s) = \frac{1}{L-1}; 0 \le s \le L-1$ is known as a Uniform Probability Density Function, which is proof that performing the intensity transformation in Equation 3 results in the random variable s characterized by a uniform PDF [1, 4]. It could also be noted that although $T(r)$ depends on $p_r(r)$, the resulting $p_s(s)$ is always uniform (as shown in Equation 5), independently of the form of $p_r(r)$, as shown in Figure 7 [1, 4].

In the case of discrete values, instead of Probability Density Functions and Integrals, Probabilities (Histogram Values) and Summations are considered instead [1, 4]. The probability of occurrence of intensity level r_k in a digital image is approximated by [1, 4]:

$$p_r(r_k) = \frac{n_k}{MN}; k = 0, 1, ..., L-1 \tag{6}$$

Where MN is the total number of pixels in the image, n_k is the number of pixels having intensity r_k, and L is the number of possible intensity levels in the image (usually 256) [1, 4]. A plot of $p_r(r_k)$ vs. r_k is known as a histogram [1, 4]. The discrete counterpart of the transformation of Equation 3 is given by [1, 4]:

$$\begin{aligned}s_k &= T(r_k) \\ &= (L-1)\sum_{j=0}^{k} p_r(r_j) \\ &= \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j; k = 0, 1, ..., L-1\end{aligned} \tag{7}$$

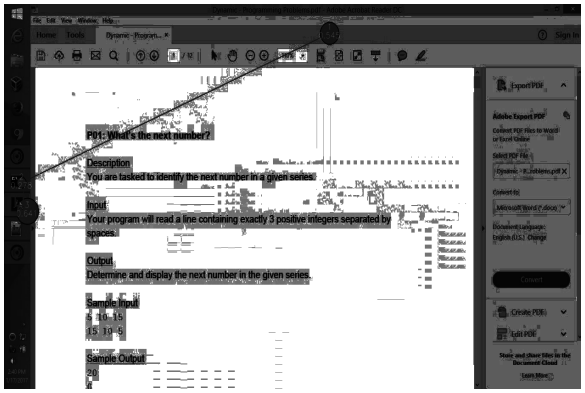

Figure 9: Histogram Equalization for a PDF file.

In other words, each pixel in the input image with intensity r_k is mapped to a corresponding pixel with level s_k in the output image [1, 4]. Here, the transformation (mapping) function $T(r_k)$ is known as Histogram Equalization or Histogram Linearization Transformation, and it satisfies conditions (1) and (2) [1, 4]. Examples of images processed with Histogram Equalization are shown in Figure 8 [1, 4, 8].

In Python, Histogram Equalization is available through Scikit-Image (skimage)'s exposure module via the function equalize_hist() [7]. Interestingly, when it is performed on some frames obtained from the dynamic eye-tracking stimuli used in the ALLS experiment, Histogram Equalization approximates a rectangle area for a line of text. Figures 9 and 10 shows the results of applying Histogram Equalization to a PDF file and a code snippet, respectively. Furthermore, Histogram Equalization is also capable of estimating rectangle areas for text when two desktop windows overlap, as shown in Figure 11.

4 RESULTS AND DISCUSSION

Because of the aforementioned properties that Histogram Equalization has when applied to text areas, it could be used to define the AOIs particularly for text areas. However, using Histogram Equalization for AOI approximation is still insufficient for two reasons. Firstly, it treats adjacent non-text elements (i.e. the GUI) as one big AOI, and does not recognize them as separate components (ex. Taskbars, Titlebars, Menu Items, etc.). A potential eye-tracking

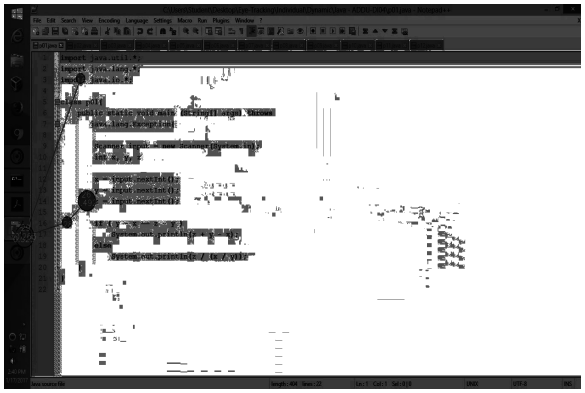

Figure 10: Histogram Equalization for a code snippet.

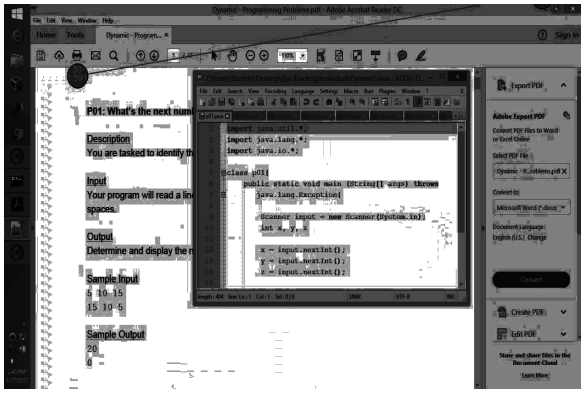

Figure 11: Histogram Equalization on two overlapping windows with text.

researcher who is also interested in user behavior in terms of interacting with the user interface aside from reading text would find this very useless. Secondly, while Histogram Equalization works for text on a light background, it is not able to estimate AOIs for a line of text on a dark background (this would be detrimental most especially if a line of text happens to be on a command-line terminal). An example of this is shown in Figure 12.

5 CONCLUSION AND FURTHER RECOMMENDATIONS

Based on the findings, Histogram Equalization is very useful for estimating AOIs for text areas on eye-tracking stimuli, specifically if it is on a light background. However, it is not very effective if the text is on a dark background, and if there are non-text elements, Histogram Equalization will simply treat them as one big AOI, and is not able to discriminate between separate elements. One possible workaround to this problem is to use traditional image segmentation techniques along with Histogram Equalization to help approximate AOI boundaries most especially for non-text elements. Another possible alternative would be to use Contrast Limited Adaptive Histogram Equalization (CLAHE), which can be considered as well for further research.

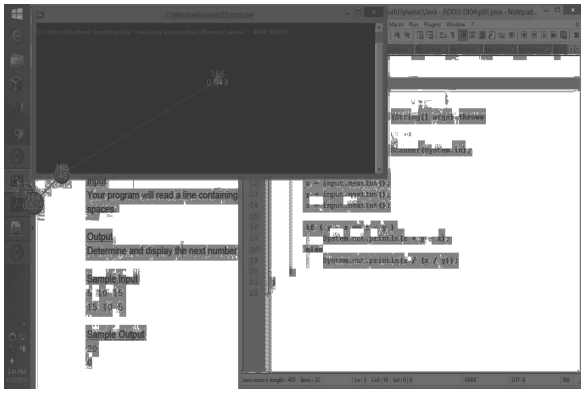

Figure 12: Histogram Equalization fails to estimate AOIs for text areas on a dark background.

ACKNOWLEDGMENTS

Mr. Lagmay would like to thank his Thesis Panelists - Dr. Andrei D. Coronel, Dr. Proceso L. Fernandez, Jr., and Mr. Hadrian Paulo Lim - for their further suggestions and recommendations as to how to improve and move forward with the study as part of his larger thesis on automatic AOI detection and tracking for dynamic eye-tracking stimuli. Mr. Lagmay would also like to thank the Ateneo Laboratory for the Learning Sciences (ALLS) for providing him with the datasets from its experiment on eye-tracking and code debugging. Above all, the authors would like to thank God for the wisdom He has given them as they push through with the research.

REFERENCES

[1] [n. d.]. Histogram Equalization. ([n. d.]). https://www.math.uci.edu/icamp/courses/math77c/demos/hist_eq.pdf
[2] Zhensong Chen, Zhiquan Qi, Fan Meng, Limeng Cui, and Yong Shi. 2015. Image Segmentation via Improving Clustering Algorithms with Density and Distance. *Procedia Computer Science* 55 (2015). Issue 2015.
[3] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. 2015. Image Segmentation using K-Means Clustering Algorithm and Subtractive Clustering Algorithm. *Procedia Computer Science* 54 (2015). Issue 2015.
[4] Rafael C. Gonzalez and Richard E. Woods. [n. d.]. *Digital Image Processing* (3rd ed.). Pearson.
[5] Jiawei Huang and Ze-Nian Li. 2010. Automatic Detection of Object of Interest and Tracking in Active Video. *J Sign Process Syst* 65 (10 2010), 49–62. https://doi.org/10.1007/s11265-010-0540-3
[6] Bing Liu, Mingui Sun, Qiang Liu, Amin Kassam, Ching-Chung Li, and Robert J. Sclabassi. 2005. Automatic Detection of Region of Interest Based on Object Tracking in Neurosurgical Video. In *Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference* (1-4). 6273–6276. https://doi.org/10.1109/IEMBS.2005.1615931
[7] Scikit-Image. [n. d.]. Histogram Equalization. ([n. d.]). https://scikit-image.org/docs/0.9.x/auto_examples/plot_equalize.html
[8] Shreenidhi Sudhakar. 2017. Histogram Equalization. (07 2017). https://towardsdatascience.com/histogram-equalization-5d1013626e64
[9] Christine Lourrine S. Tablatin and Ma. Mercedes T. Rodrigo. 2018. Analysis of Static Code Reading Patterns. *18th Philippine Computing Science Congress* (2018).