# Application of Parallel Computing in the Analysis of Usability Tests, Under the Mouse Tracking Approach

Gabriel Elías Chanchí G. [1], Laura Sofía Chanchí P. [2], Manuel Alejandro Ospina A.[1]

[1] *Universidad de Cartagena, Avenida del Consulado, Calle 30 # 39B-192, Cartagena, Colombia*
[2] *Universidad del Cauca, Campus Tulcán, Popayán, Colombia*

***Abstract* – Mouse tracking tests play a crucial role in evaluating software usability, but their efficient image processing remains challenging, especially with high-resolution images and numerous participants. This article introduces a novel method, leveraging parallel computing, for the efficient analysis of interaction zones in mouse tracking test images. Following Pratt's iterative research pattern, the proposed method is implemented using Python libraries Dask and OpenCV, validated through a proof of concept on Eclipse software. Results demonstrate the parallel approach's remarkable efficiency, being 252 times faster than the sequential method across various executions. The method's potential impact is discussed, providing a valuable reference for developing tools in usability and other application contexts. The open-source tools employed, Dask and OpenCV, prove suitable for parallel image analysis, offering versatility for broader application in diverse fields. This work contributes to advancing the field of mouse tracking analysis by significantly improving processing efficiency and lays the groundwork for future tools and methodologies.**

***Keywords* – Parallel computing, sequential computing, mouse tracking, usability, usability test.**

DOI: 10.18421/TEM132-04
https://doi.org/10.18421/TEM132-04

**Corresponding author:** Gabriel Elías Chanchí G,
*Universidad de Cartagena, Avenida del Consulado, Calle 30 # 39B-192, Cartagena, Colombia*
**Email:** gchanchig@unicartagena.edu.co

*Received: 30 December 2023.*
*Revised: 13 March 2024.*
*Accepted: 22 March 2024.*
*Published: 28 May 2024.*

## 1. Introduction

Given the vast number of applications available in online stores, one of the most crucial attributes to enhance both software quality and user experience is usability [1], [2], [3], [4]. Usability not only contributes to improving the competitiveness of software companies but also enhances user productivity in interaction [5], [6], [7], [8].

According to Nielsen [9], usability can be understood as an attribute of software quality, allowing the evaluation of how easy interfaces are to use. Similarly, according to ISO 9241-11, usability can be defined as the extent to which a user achieves specific goals within software with effectiveness, efficiency, and satisfaction [10], [11], [12], [13]. One way to assess the usability of software and its defining attributes (effectiveness, efficiency, and satisfaction) is through user tests, where end-users are observed in a controlled usability lab while performing a set of tasks in specific software [14], [15], [16].

One of the complementary tests that is performed within a user test and that contributes to the improvement of interaction efficiency and therefore to the improvement of the user experience is the mouse tracking test, in which the capture and analysis of the mouse trace is used to identify the areas of the screen where there is greater interaction between the user and the software, revealing patterns and preferences in mouse movement [17], [18], [19], [20]. Significance of these tests lies in their ability to determine the optimal placement of interface components by aligning with high-priority zones. This strategic alignment not only enhances efficiency but also significantly contributes to reducing the user's mental load during interaction [21], [22]. Thus, mouse tracking tests provide a detailed perspective that facilitates the creation of more intuitive interfaces focused on user needs [23].

One of the challenges of mouse tracking tests is the efficient analysis of the interaction zones in the images with the mouse trace, given the amount of data or pixels contained in the increasingly high quality images and the number of users performing the test [24], [25]. In this regard, the conventional or sequential computing approach relies on pixel counting for areas of interest by iterating through the image from left to right and top to bottom. Consequently, computational efficiency becomes a challenge in the analysis of mouse tracking tests. Based on the above, it is convenient to take advantage of the benefits in image analysis in different application contexts provided by parallel computing [26], which corresponds to a computing approach that involves the simultaneous execution of multiple tasks or processes, with the objective of improving computational efficiency and performance. Thus, instead of performing one task at a time, as in sequential computing, parallel computing divides complex problems into smaller tasks that can be executed simultaneously on multiple processors or cores [27], [28].

Different works have been conducted on the topic of mouse tracking. In [21], a tool is proposed for the analysis of interaction zones in mouse tracking tests using sequential computing. Similarly, in [25], a tool was developed for the analysis of mouse traces using unsupervised learning models, such that the main interaction zones are detected by obtaining clusters and their associated centroids. In [29] a study based on eye tracking and mouse tracking was developed on commercial and educational portals in Nigeria, using OGAMA software, in order to identify search and browsing patterns employed by users interacting with these web portals. In [30] a study was conducted using machine learning techniques and sequential computing with the main objective of predicting users' implicit interest in products of an online store based on their mouse behavior through various product page elements, which allows businesses to acquire the understanding of their customers interests to innovate and develop new products and services. In [23] a framework that employs methods for eye and mouse tracking, keyboard input, self-assessment questionnaire and artificial intelligence algorithms was proposed to evaluate user experience and categorize users in terms of performance profiles. This framework makes use of the Tracking Techniques User eXperience Tool T2-UXT to collect, collate, process, and visualize data obtained from users' interactions. Furthermore, the evaluation of the mouse tracking method is conducted in [31], aiming to explore perception and digital map cognition.

Considering that the practical implementation of mouse-tracking studies requires the use of software tools that are capable of supporting the process of experimental design, data analysis and visualization, this work includes a concise compilation of software tools discussed in previous scientific articles, such as MouseTrack, OGAMA, Qualtrics mouse-tracking and MatMouse. Additionally the analysis of the use of this method in term of its advantages and limitations is presented, demonstrating that mouse tracking could serve as one of the most powerful methods in cartographic research. From the previous works, it can be observed that while the advantages of mouse tracking tests are leveraged to enhance interaction or detect areas of interest in the field of marketing and other contexts, these works do not focus on improving the efficiency of the analysis of areas of interest, nor do they make use of the parallel computing approach for the analysis and processing of images with mouse traces.

In this paper we propose as a contribution a method based on the parallel computing paradigm for the analysis of interaction zones on images obtained from usability tests under the mouse tracking approach. In order to evaluate the effectiveness and efficiency of the implemented method, a proof of concept was performed in which the results of the interaction zone analysis using the sequential approach and the parallel computing approach were compared on an image obtained from a mouse tracking test performed on the Eclipse development tool and obtained from the IOGraphica portal. For the comparison, the times obtained in performing different numbers of executions on the conventional and parallel computing methods implemented were taken into account. Likewise, the efficiency of the parallel computation method with respect to the conventional one was calculated in different number of executions and in average. On the other hand, based on the results obtained in different executions, the average time taken by the methods to perform the zone interaction analysis in the proof of concept was determined. For the implementation of the proposed method in this article, the advantages provided by the open-source parallel computing library Dask were utilized. This library allows the processing of large volumes of data by organizing them into blocks and enabling parallel and/or distributed processing. Additionally, the OpenCV image processing library, the NumPy library for vectorized operations, the SciPy library for curve fitting to measured data, and the scikit-learn library for applying result evaluation metrics were employed.

The proposed approach in this article aims to serve as a reference for the development of software tools for the efficient analysis of images derived from mouse tracking tests.

Similarly, such approaches can be extrapolated for the analysis of heatmaps derived from eye-tracking tests in the context of usability or at the usability and marketing levels. In this regard, the use of powerful open-source libraries such as Dask and OpenCV facilitates the harnessing of parallel computing in the analysis and processing of images in various application contexts.

The rest of the article is organized as follows: Section 2 presents the methodological phases considered in the development of this work. In Section 3, the results obtained in this research are described, including the specification of the proposed method, the description of method implementation, and finally, the case study where the conventional method and the parallel computing-based method are compared. Lastly, in Section 4, conclusions and future work derived from this research are presented.

## 2. Methodology

For the development of the present research, the four methodological phases of the iterative research pattern proposed by Pratt were taken into consideration (Figure 1) [32], [33].

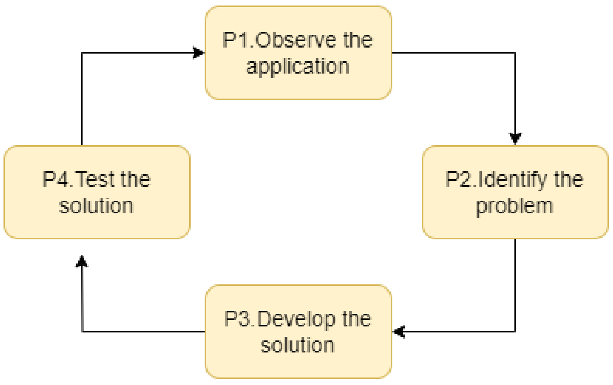


*Figure 1. Methodology considered*

In Phase 1 of the methodology, the conceptualization and characterization of the analysis carried out in conventional mouse tracking tests were performed to identify challenges in processing images with the mouse trace. Thus, the flowchart presented in Figure 2 was created, depicting the overall process conducted under the sequential computing approach to obtain interaction percentages in areas of interest in images with the mouse trace.

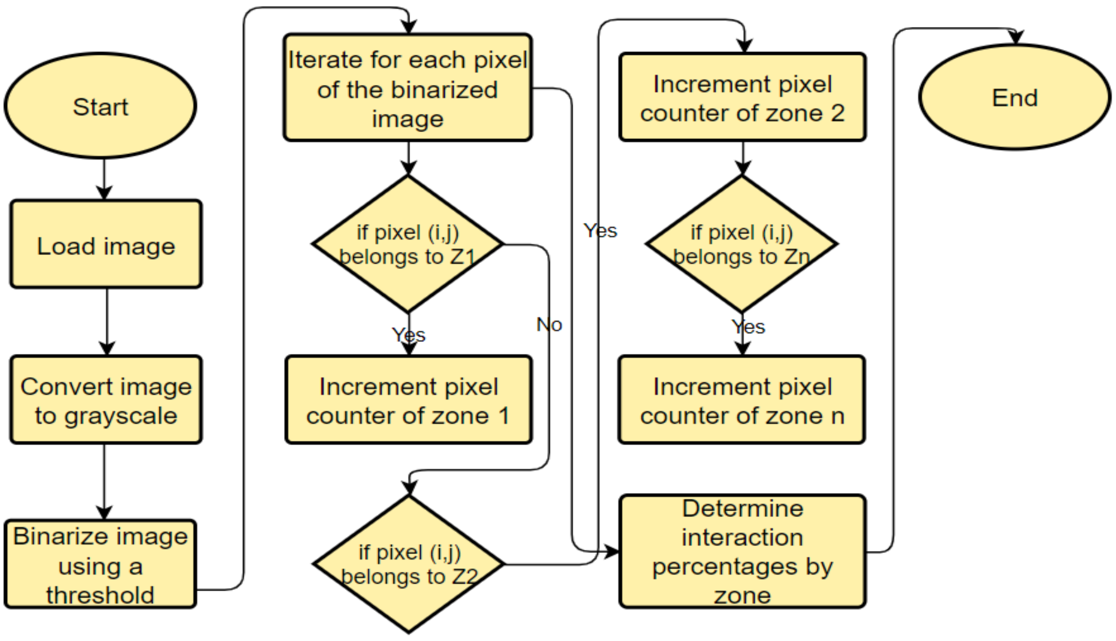

Start
Load image
Convert image to grayscale
Binarize image using a threshold
Iterate for each pixel of the binarized image
if pixel (i,j) belongs to Z1
Yes
No
Increment pixel counter of zone 1
if pixel (i,j) belongs to Z2
Yes
Increment pixel counter of zone 2
if pixel (i,j) belongs to Zn
Yes
Increment pixel counter of zone n
Determine interaction percentages by zone
End

*Figure 2. Sequential computing approach*

According to Figure 2, once the image with the mouse trace is loaded, the process involves converting it to grayscale and binarizing the image using a reference threshold. From the binarized image, iteration is performed for each pixel to count the pixels belonging to a specific area of interest. Thus, based on the counters associated with each area of interest, the calculation of interaction percentages for each of the zones is carried out.

On the other hand, in Phase 2 of the methodology, based on the computational efficiency challenges identified in Phase 1, a method based on parallel computing was designed to process mouse tracking images and identify interaction percentages by screen zones. In Phase 3 of the methodology, the method designed in Phase 2 was implemented using open-source tools. Specifically, the Dask, OpenCV, and NumPy libraries were selected for the parallel processing and analysis of images with mouse traces.

Finally, in Phase 4 of the methodology, a proof of concept was conducted using an image obtained from a mouse tracking test performed on the Eclipse development environment by the company IOGraphica, in order to evaluate the proposed method. Thus, based on the test image, a comparative analysis of the efficiency of the parallel computing-based method versus the sequential computing-based method was carried out. For this purpose, both methods were implemented as functions in the Python language and executed 10, 20, 30, 40, 50, 60, 70, 90, and 100 times to obtain execution times per run and the average time. Likewise, the efficiency achieved per execution and the average efficiency of the parallel computing-based method was determined.

For these tasks, the advantages provided by the timeit library were utilized, enabling the measurement of the time taken to execute a specific function.

## 3. Results

Based on the challenges identified in characterizing mouse trace analysis through sequential computing, the first step involved designing a method based on parallel computing for processing images obtained in mouse tracking tests and identifying the corresponding percentages for interaction zones in these images. Thus, Figure 3 shows the method based on parallel computation designed from the process described in Figure 2.

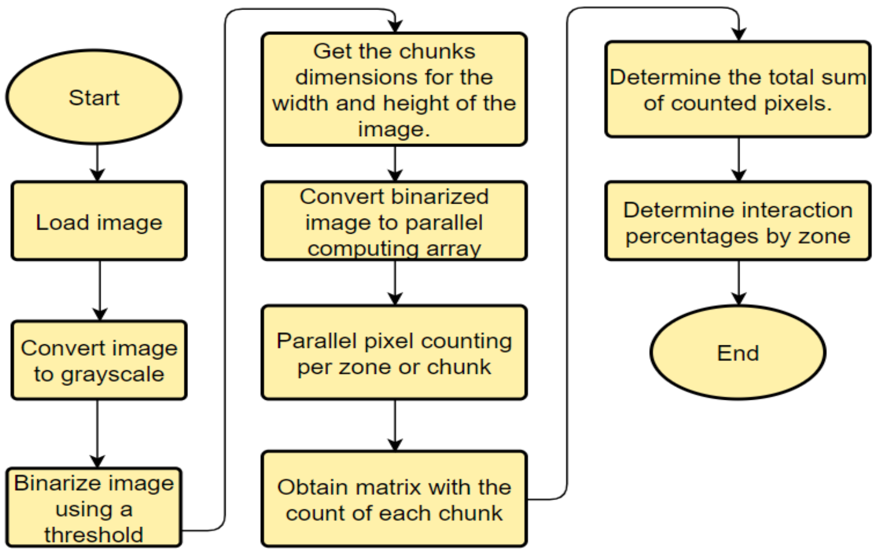

Start
Load image
Convert image to grayscale
Binarize image using a threshold
Get the chunks dimensions for the width and height of the image.
Convert binarized image to parallel computing array
Parallel pixel counting per zone or chunk
Obtain matrix with the count of each chunk
Determine the total sum of counted pixels.
Determine interaction percentages by zone
End

*Figure 3. Proposed method based on parallel computing*

According to Figure 3, it is possible to observe that once the image with the mouse trace is loaded, the image is converted to grayscale and binarized from a reference threshold. From the binarized image, the dimension of the chunks is obtained for the width and height of the image, which correspond to the blocks in which the processing will be performed in parallel and in this case have been made to correspond to the number of interaction zones. For example, if the analysis is intended for 4 interaction zones, 4 chunks are obtained with dimensions proportional to the width and height of the image. With the obtained chunk dimensions, the binarized image is then converted into a parallel computing array, which is segmented into the determined processing blocks. From the segmented array, parallel pixel counting is carried out per block or zone, resulting in a matrix with the total pixel count per block.

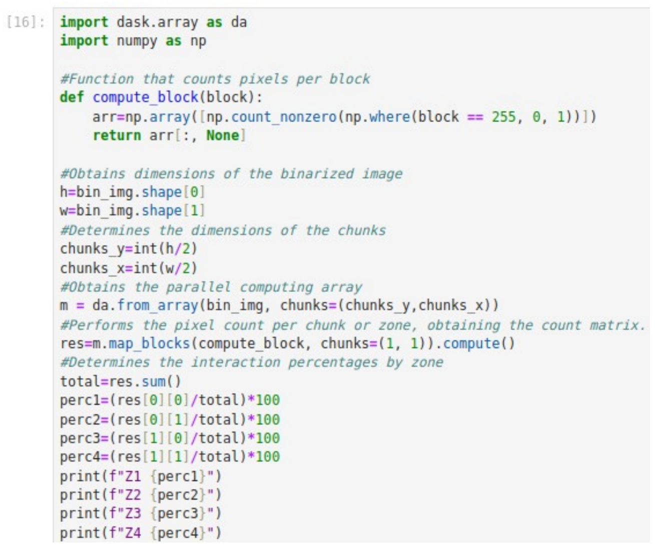
This matrix is used to calculate the overall sum of interaction pixels and, consequently, to calculate the percentage of interaction per area of interest.

Based on the process described in Figure 3, the parallel computing method was implemented for the

analysis of 4 interaction zones, using Python libraries such as Dask, Numpy, and OpenCV, as illustrated in

Figure 4. In Figure 4, it is observed how, from the binarized mouse tracking image, the dimensions of the image are obtained in the variables w and h. Since the analysis was programmed for 4 zones, the dimensions of the chunks (chunk_x, chunk_y) are obtained by dividing the variables w and h by two. Using the dimensions of the chunks, the parallel computing array (m), specific to the dask library and compatible with numpy arrays, is obtained. This array has been segmented according to the defined blocks or chunks.

The m array is used to invoke the map_blocks function, which allows executing the same function for each of the defined blocks or chunks. In this case, the compute_block function is executed in parallel for the blocks and obtains as a result a matrix with the count of black pixels per block (res), that is, the pixels where interaction was detected in a particular zone. From the results of the res matrix, it is possible to determine the total number of interaction pixels and the interaction percentages associated with each of the 4 zones of interest.



```
import dask.array as da
import numpy as np

#Function that counts pixels per block
def compute_block(block):
    arr=np.array([np.count_nonzero(np.where(block == 255, 0, 1))])
    return arr[:, None]

#Obtains dimensions of the binarized image
h=bin_img.shape[0]
w=bin_img.shape[1]
#Determines the dimensions of the chunks
chunks_y=int(h/2)
chunks_x=int(w/2)
#Obtains the parallel computing array
m = da.from_array(bin_img, chunks=(chunks_y,chunks_x))
#Performs the pixel count per chunk or zone, obtaining the count matrix.
res=m.map_blocks(compute_block, chunks=(1, 1)).compute()
#Determines the interaction percentages by zone
total=res.sum()
perc1=(res[0][0]/total)*100
perc2=(res[0][1]/total)*100
perc3=(res[1][0]/total)*100
perc4=(res[1][1]/total)*100
print(f"Z1 {perc1}")
print(f"Z2 {perc2}")
print(f"Z3 {perc3}")
print(f"Z4 {perc4}")
```

*Figure 4. Implementation of the proposed method using dask*

In order to evaluate the proposed approach, a proof of concept was developed in which an analysis was conducted on a mouse trace obtained from a mouse tracking test conducted in the Eclipse programming environment by IOGraphica. This test was performed using the IOGraph mouse tracking tool, created by the same company. Thus, in Figure 4, the image of the mouse trace considered in the proof of concept is presented. This illustration depicts the four interaction zones under consideration, which in this case correspond to the four blocks or chunks used in parallel computation.

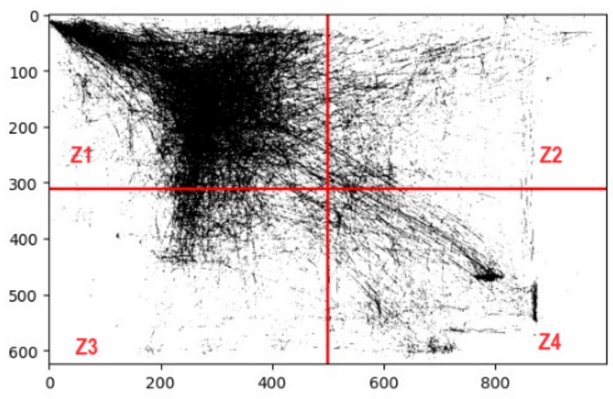


*Figure 5. Image used in the proof of concept*

As mentioned in the methodology, both the sequential method and the parallel computing-based method were implemented as functions in the Python language to assess their effectiveness and efficiency. It is worth noting that the parallel computing-based method was effective in computing the areas of interest, obtaining the same percentage results for on-screen areas of interest as the sequential computing-based method. These percentage results are presented in Figure 6.

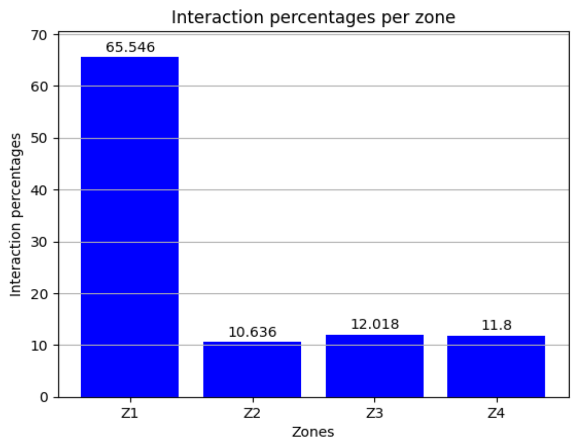


*Figure 6. Interaction percentages per zone*

The results obtained in Figure 6 are consistent with the distribution of the mouse trace presented in Figure 4, such that 65.546% of user interaction is concentrated in zone 1 of the screen. The above result can be explained by the fact that the main functions of the Eclipse development environment are located in an area of high visual hierarchy, i.e. in the upper left part of the screen, which helps to minimize the user's memory load and facilitate interaction. Once the proper functioning of the method was verified, the efficiency of the methods was compared by conducting varying numbers of executions (10, 20, 30, 40, 50, 60, 70, 80, 90, and 100) on both methods and measuring the time spent processing the different executions. Thus, Table 1 presents the results obtained in terms of time expended by both methods for the different executions conducted.

*Table 1. Execution times obtained*

| Executions | Execution time sequential approach (s) | Execution time parallel approach (s) |
|---|---|---|
| 10 | 11.425 | 0.047 |
| 20 | 22.382 | 0.086 |
| 30 | 33.523 | 0.134 |
| 40 | 44.585 | 0.174 |
| 50 | 56.064 | 0.222 |
| 60 | 66.630 | 0.260 |
| 70 | 77.489 | 0.308 |
| 80 | 89.821 | 0.338 |
| 90 | 100.734 | 0.420 |
| 100 | 112.015 | 0.446 |

According to the results in Table 1, it is possible to appreciate how, when comparing the two considered approaches, the time spent in the parallel approach during different executions is significantly lower. In the worst-case scenario (90 executions), it is 240 times better compared to the time spent in the sequential approach. In the same vein, it is observed that for 100 executions, the time spent by the sequential approach is over 112 seconds, while in the parallel approach, it barely reaches 0.5 seconds. The rate of increase per requests in the sequential and parallel approaches can be more clearly observed in Figures 7 and 8, where the time spent by both approaches for a different number of executions is presented, along with the curve that best fits the data, using the **curve_fit** function from the SciPy library in Python which allows for finding the optimal parameters corresponding to the slope and intercept values that describe the data behavior.

In this context, for the case of the curve equation in Figure 7, a standard error of 0.004086 was obtained for the slope value, and 0.253514 for the intercept value. These values were calculated from the covariance matrix associated with the optimal adjusted values. This metric indicates how much estimates are likely to fluctuate when the model is fitted to diverse datasets, thus allowing for the evaluation of the reliability of the estimates for the calculated coefficients, in this case, the slope and intercept. In the same vein, concerning the mean squared error (MSE), a value of 0.110178 was obtained, which serves to assess how well the curve fits the data.

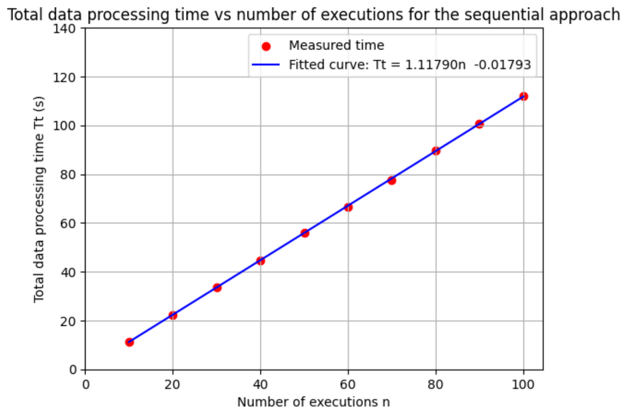


*Figure 7. Variation of the total data processing time with the number of tests for the sequential approach*

The fitted curve in Figure 7 reveals that in the sequential approach, there is a progressive increase in the total processing time from 11.425 seconds (for 10 executions) to 112.015 seconds (for 100 executions). Additionally, the equation derived from the fitted curve leads to the conclusion that, for each execution conducted, the time increases by 1.117 seconds.

This finding implies an efficiency trade-off in the implementation of mouse tracking tools, considering that usability tests under this approach involve processing mouse traces from multiple users.

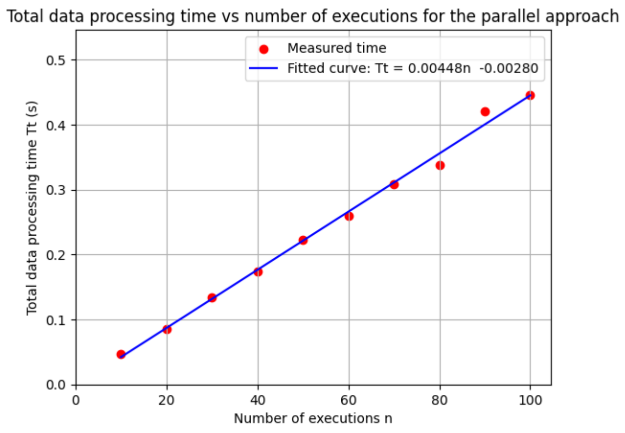


*Figure 8. Variation of the total data processing time with the number of tests for the parallel approach*

Similarly, in the case of the fitted equation in Figure 8, a standard error of 0.000108 was obtained for the slope value, and 0.006728 for the intercept value. Meanwhile, at the MSE level, a value of 0.000078 was obtained. By calculating the standard error of the estimated parameters, namely the slope and intercept, it is found that the parallel approach exhibits significantly lower standard error values compared to those presented in the sequential approach. This indicates less variability in the parameters for future estimations and, consequently, greater consistency. Additionally, it was observed that in terms of MSE, the parallel approach shows significantly lower values compared to the sequential approach, indicating reduced data fitting errors and, therefore, more consistent results.

The fitted curve in Figure 8 reveals that in the parallel approach, there is a progressive increase in the total processing time from 0.047 seconds (for 10 executions) to 0.446 seconds (for 100 executions). Additionally, the equation derived from the fitted curve leads to the conclusion that, for each execution conducted, the time increases by 0.004 seconds.

When comparing the total execution time ranges for different test quantities, a significant difference is observed between the measured time values of the parallel approach and the sequential approach, with the resulting times of the parallel approach being much lower than those corresponding to the sequential approach. This conclusively demonstrates the remarkable advantage of implementing the parallel approach in terms of efficiency when choosing the most suitable approach for data processing.

Similarly, it is observed that the sequential approach has a slope 249.53125 times greater than that of the parallel approach. This implies that, in the sequential approach, the total execution time increases at a faster rate relative to the number of executions compared to the parallel approach. This clearly highlights an advantage for the parallel approach, as increasing the number of executions results in a much smaller and less pronounced increase in execution time compared to the sequential approach.

On the other hand, Figures 9 and 10 display the average data processing time for each of the considered approaches. This is calculated as the total data processing time, Tt, divided by the number of tests, n. Thus, in Figure 9, the results obtained for the sequential approach are presented along with a dashed line representing the overall average time per execution.

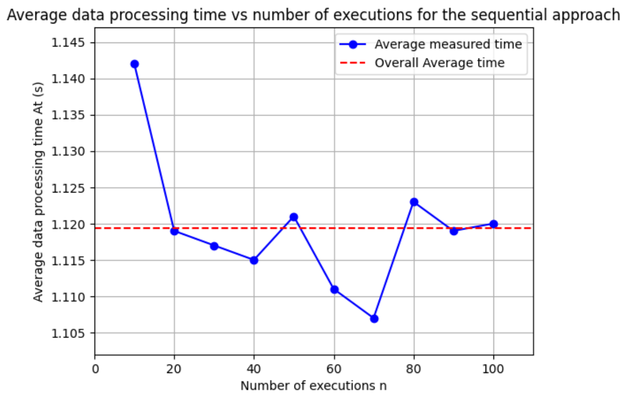


*Figure 9. Variation of the average data processing time with the number of tests for the sequential approach*

Similarly, in Figure 10, the results obtained for the parallel approach are presented along with a dashed line indicating the overall average processing time per execution.

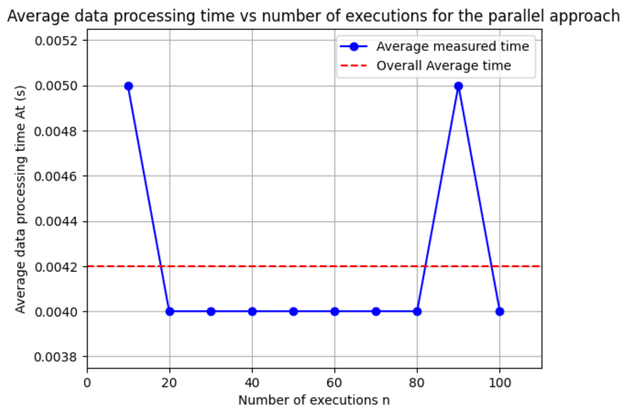


*Figure 10. Variation of the average data processing time with the number of tests for the parallel approach*

Based on the results obtained in Figures 9 and 10, a significant difference in average times per number of executions between the parallel and sequential approaches can be observed. This is evident in the overall average time obtained in all executions when processing the test image (red dashed line in Figures 9 and 10), which is 0.042 seconds for the parallel approach and 1.1120 seconds for the sequential approach. Similarly, it is observed that the variation in average times per execution for the parallel approach ranges between 0.004 and 0.005 seconds, while in the case of the sequential approach, the variation is between 1.107 and 1.142 seconds. Thus, a greater variability in times is observed in the case of the sequential approach, while in the parallel approach, there is greater consistency in the results.

Finally, in Figure 11, the efficiency of the parallel approach compared to the sequential approach is presented. This is calculated as the total processing time of the data for the sequential approach divided by the total processing time of the data for the parallel approach.

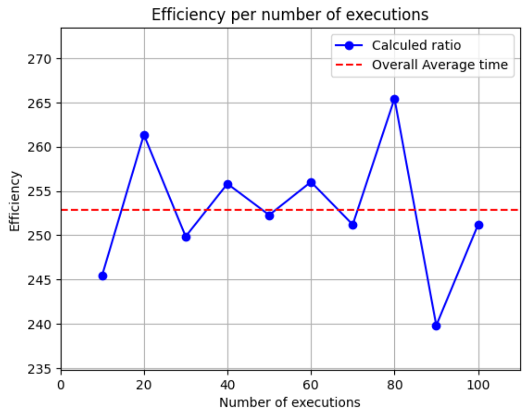


*Figure 11. Variation of the efficiency with the number of tests*

It can be observed that all the values obtained are greater than 1 and even exceed 240, indicating that in all cases the time obtained for different executions is much lower than the time obtained for the sequential approach. These values vary in the range of 239,826 times to 265,441 times, with the average corresponding to a value of 252,851 times. This means that, on average, the sequential approach presents a total execution time 252,851 times greater than the total execution time in the parallel approach.

## 4. Discussion

In this work, a contribution has been proposed involving the design and implementation of a method based on parallel computing for the analysis of interaction zones in mouse trace images obtained in usability tests under the mouse tracking approach.

Based on the results obtained in this study, it was observed that the proposed method is significantly more efficient than the method based on sequential computing. This is evident in the fact that, in various executions, the parallel computing method was, on average, 252 times faster than the sequential method. Furthermore, it is evident through the calculation of MSE and the standard error of the estimated values that the model describing the parallel approach exhibits less variability and, therefore, greater consistency than the model corresponding to the sequential approach. The aforementioned is a relevant contribution to enhance the efficiency of the tools proposed in [21] and [25], where tools based on sequential computing are proposed for the analysis of zones of interest in images derived from mouse tracking tests, involving pixel counting by zone and respective clustering techniques.

Similarly, the approach proposed in this work is a relevant contribution to academic and business level for the implementation of tools based on parallel computing for the analysis of mouse tracking tests, since the proposed method was implemented using open-source libraries and technologies. In this regard, the Dask library proved to be suitable for image processing, as it features data structures compatible with conventional ones and has the flexibility to be segmented into blocks for parallel processing. Similarly, the OpenCV library demonstrated suitability for basic image operations (image scaling, conversion to grayscale, image binarization), which is crucial in preprocessing mouse trace images. Additionally, this library uses NumPy-like arrays for various image operations, making them compatible with the data structures employed by the Dask library.

## 5. Conclusion

Considering that usability is an attribute defining software quality, it is important to integrate different types of usability tests into the software development process, where mouse tracking tests stand out for providing feedback on interface design. Thus, one of the main advantages of usability tests under the mouse tracking approach is the identification of areas of interest where user interaction is concentrated, aiming to enhance interface design by identifying key software functionalities that may not be located in areas of high visual hierarchy. In this way, these tests contribute to improving user productivity, which corresponds to one of the advantages of usability.

Taking into consideration that mouse tracking tests involve the analysis of mouse trace images from multiple users, one of the commitments of software tools focused on analyzing this type of test is efficiency.

Thus, in this work, the main contribution proposed was the design and implementation of a method based on parallel computing for the analysis of areas of interest in mouse trace images. This method creates a parallel processing block for each analyzed area. The proposed method aims to serve as a reference for the implementation of efficient tools for the analysis of mouse tracking tests in the context of usability or other application contexts.

Based on the results obtained in the proof of concept conducted on an image from mouse tracking test using Eclipse software by IOGraphica, it was possible to conclude that the parallel computing method, in addition to being effective in determining interaction zones, is significantly more efficient than the sequential computing method. In this regard, after conducting multiple executions of both methods on the same image, it was found that, on average, the parallel computing method with four processing blocks (one for each interaction zone) is 252 times faster than the sequential computing method. These results indicate that the proposed method is suitable for integration into tools for processing images derived from mouse tracking tests in various application contexts.

The open-source tools employed proved to be suitable for implementing the parallel computing method for the analysis of interaction zones in mouse tracking tests. Specifically, the Dask library facilitated the division of the image into processing blocks (one for each zone) and the parallel counting of pixels in each block. Similarly, the OpenCV library allowed for various preprocessing operations on images with the mouse trace, such as conversion to grayscale or binarization of these images. Thus, these libraries aim to serve as a reference for consideration in academic or business settings for the development of tools for image analysis under the parallel computing approach.

As a future work arising from the present research, it is intended to: a) Build a tool supported by the proposed approach, enabling the processing and analysis of multiple mouse traces derived from usability tests; b) Combine the proposed approach with the analysis of interaction zones using unsupervised learning techniques or clustering.

**Acknowledgements**

*The authors express their gratitude to the University of Cartagena and the University of Cauca for the support received in the development of this research.*

## References

[1]. Abbas, A. M. H., Ghauth, K. I., & Ting, C.-Y. (2022). User Experience Design Using Machine Learning: A Systematic Review. *IEEE Access, 10*, 51501-51514. Doi: 10.1109/ACCESS.2022.3173289

[2]. Chanchí-Golondrino, G. E., Sierra Martínez, L. M., & Campo Muñoz, W. Y. (2022). Fuzzy Logic-Based System for the Estimation of the Usability Level in User Tests. *International Journal of Computers Communications & Control, 17*(2). Doi: 10.15837/ijccc.2022.2.4476

[3]. Haaksma, T. R., De Jong, M. D. T., & Karreman, J. (2018). Users' Personal Conceptions of Usability and User Experience of Electronic and Software Products. *IEEE Transactions on Professional Communication, 61*(2), 116-132. Doi: 10.1109/TPC.2018.2795398

[4]. T. Bantug, E., G. Luciano, R., & V. Bauat, R. (2021). Heuristic Usability Evaluation: A Case Study of Online Enrolment System of a State University. *International Journal of Advanced Engineering Research and Science, 8*(6), 360-365. Doi: 10.22161/ijaers.86.44

[5]. Çetin, G., & Göktürk, M. (2008). A Measurement Based Framework for Assessment of Usability-Centricness of Open Source Software Projects. *2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, 585-592. Doi: 10.1109/SITIS.2008.106

[6]. Dinkel, C., Billenstein, D., Goller, D., & Rieg, F. (2018). User-oriented optimization of the GUI of a finite element programme to enhance the usability of simulation tools. *2018 South-Eastern European Design Automation, Computer Engineering, Computer Networks and Society Media Conference (SEEDA_CECNSM),* 1-5. Doi: 10.23919/SEEDA-CECNSM.2018.8544936

[7]. Hering, D., Schwartz, T., Boden, A., & Wulf, V. (2015). Integrating Usability-Engineering into the Software Developing Processes of SME: A Case Study of Software Developing SME in Germany. *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering,* 121-122. Doi: 10.1109/CHASE.2015.22

[8]. Li, G., He, Y., Tang, Y., Shen, X., & He, L. (2023). Perceived Usability of Computer-Aided Engineering Software. *2023 IEEE 28th Pacific Rim International Symposium on Dependable Computing (PRDC),* 78-80. Doi: 10.1109/PRDC59308.2023.00019

[9]. Nielsen, J. (2012). *Usability 101: Introduction to Usability. N N group.*Retrieved from: https://www.nngroup.com/articles/usability-101-introduction-to-usability/ [accessed: 15 November 2023].

[10]. Blanco-Gonzalo, R., Sanchez-Reillo, R., Goicoechea-Telleria, I., & Strobl, B. (2018). The Mobile Pass Project: A User Interaction Evaluation. *IEEE Transactions on Human-Machine Systems, 48*(3), 311-315. Doi: 10.1109/THMS.2018.2791571

[11]. Chanchí-Golondrino, G. E. (2023). Estimación del atributo de satisfacción en test con usuarios mediante técnicas de análisis de sentimientos. *Prospectiva, 21*(2), 40-50. Doi: 10.15665/rp.v21i2.3248

[12]. Obermeier, M., Braun, S., & Vogel-Heuser, B. (2015). A Model-Driven Approach on Object-Oriented PLC Programming for Manufacturing Systems with Regard to Usability. *IEEE Transactions on Industrial Informatics, 11*(3), 790-800. Doi: 10.1109/TII.2014.2346133

[13]. Weichbroth, P. (2020). Usability of Mobile Applications: A Systematic Literature Study. *IEEE Access, 8,* 55563-55577. Doi: 10.1109/ACCESS.2020.2981892

[14]. Ali, W., Riaz, O., Mumtaz, S., Khan, A. R., Saba, T., & Bahaj, S. A. (2022). Mobile Application Usability Evaluation: A Study Based on Demography. *IEEE Access, 10,* 41512-41524. Doi: 10.1109/ACCESS.2022.3166893

[15]. Delgado-Aguedelo, D. M., Girón-Timaná, D., Chanchí-Golondrino, G. E., & Máceles-Villalba, K. (2021). Estimate of the satisfaction attribute in user tests from facial expression analysis. *Revista Ingenierías Universidad de Medellín, 19*(36), 13-28.

[16]. Razak, F. H. A., Hafit, H., Sedi, N., Zubaidi, N. A., & Haron, H. (2010). Usability testing with children: Laboratory vs field studies. *2010 International Conference on User Science and Engineering (i-USEr),* 104-109. Doi: 10.1109/IUSER.2010.5716733

[17]. Arroyo, E., Selker, T., & Wei, W. (2006). Usability tool for analysis of web designs using mouse tracks. *CHI '06 Extended Abstracts on Human Factors in Computing Systems,* 484-489. Doi: 10.1145/1125451.1125557

[18]. Navalpakkam, V., & Churchill, E. (2012). Mouse tracking: Measuring and predicting users' experience of web-based content. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 2963-2972. Doi: 10.1145/2207676.2208705

[19]. Souza, K. E. S., Seruffo, M. C. R., De Mello, H. D., Souza, D. D. S., & Vellasco, M. M. B. R. (2019). User Experience Evaluation Using Mouse Tracking and Artificial Intelligence. *IEEE Access, 7,* 96506-96515. Doi: 10.1109/ACCESS.2019.2927860

[20]. Katerina, T., Nicolaos, P., & Charalampos, Y. (2014). Mouse tracking for web marketing: enhancing user experience in web application software by measuring self-efficacy and hesitation levels. *Int. J. Strateg. Innovative Mark*, *1*, 233-247.

[21]. Albornoz, D. A., Moncayo, S. A., Ruano-Hoyos, S., Chanchí-Golondrino, G. E., & Márceles-Villalba, K. (2019). Software system for executing usability tests under the mouse tracking approach, *TecnoLógicas, 22*, 19-31. Doi: 10.22430/22565337.1511

[22]. Aviz, I. L., Souza, K. E., Ribeiro, E., De Mello Junior, H., & Seruffo, M. C. D. R. (2019). Comparative study of user experience evaluation techniques based on mouse and gaze tracking. *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web,* 53-56. Doi: 10.1145/3323503.3360623

[23]. Souza, K. E. S. D., Aviz, I. L. D., Mello, H. D. D., Figueiredo, K., Vellasco, M. M. B. R., Costa, F. A. R., & Seruffo, M. C. D. R. (2022). An Evaluation Framework for User Experience Using Eye Tracking, Mouse Tracking, Keyboard Input, and Artificial Intelligence: A Case Study. *International Journal of Human–Computer Interaction, 38*(7), 646-660. Doi: 10.1080/10447318.2021.1960092

[24]. Cegan, L., & Filip, P. (2017). Advanced web analytics tool for mouse tracking and real-time data processing. *2017 IEEE 14th International Scientific Conference on Informatics,* 431-435. Doi: 10.1109/INFORMATICS.2017.8327288

[25]. Chanchí-Golondrino, G. E., Ospina-Alarcón, M., & Campo-Muñoz, W. (2022). Application of clustering techniques in the analysis of mouse tracking tests. *ARPN Journal of Engineering and Applied Sciences, 17*(13), 1358-1363.

[26]. Fan, M., Zuo, X., & Zhou, B. (2024). Parallel Computing Method of Commonly Used Interpolation Algorithms for Remote Sensing Images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 17*, 315-322. Doi: 10.1109/JSTARS.2023.3329018

[27]. Rakhimov, M., Mamadjanov, D., & Mukhiddinov, A. (2020). A High-Performance Parallel Approach to Image Processing in Distributed Computing. *2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT),* 1-5. Doi: 10.1109/AICT50176.2020.9368840

[28]. Wu, Q., Spiryagin, M., Cole, C., & McSweeney, T. (2020). Parallel computing in railway research. *International Journal of Rail Transportation, 8*(2), 111-134. Doi: 10.1080/23248378.2018.1553115

[29]. Oyekunle, R., Bello, O., Jubril, Q., Sikiru, I., & Balogun, A. (2020). Usability Evaluation using Eye-Tracking on E-Commerce and Education Domains. *Journal of Information Technology and Computing, 1*(1), 1-13. Doi: 10.48185/jitc.v1i1.43

[30]. SadighZadeh, S., & Kaedi, M. (2022). Modeling user preferences in online stores based on user mouse behavior on page elements. *Journal of Systems and Information Technology, 24*(2), 112-130. Scopus. Doi: 10.1108/JSIT-12-2019-0264

[31]. Krassanakis, V., & Misthos, L.-M. (2023). Mouse Tracking as a Method for Examining the Perception and Cognition of Digital Maps. *Digital, 3*(2), 127-136. Doi: 10.3390/digital3020009

[32]. Pratt, K. (2009). *Design Patterns for Research Methods: Iterative Field Research*. Kpratt. Retrieved from: https://kpratt.net/wp-content/uploads/2009/01/research_methods.pdf [accessed: 19 November 2023].

[33]. Anacona-Campo, F. J., Cobos-Lozada, C.-A., & Mendoza-Becerra, M. (2020). Algoritmo greedy para predecir el índice de servicio de pavimento basado en agrupación y regresión lineal. *Investigación e Innovación en Ingenierías, 8*(3), 119-134. Doi: 10.17081/invinno.8.3.4708