University of Prince Mugrin

College of Computer and Cyber Sciences

Artificial Intelligence Department

AI 385 – Computer Vision

Usability Testing Analysis with Mouse Tracking and Facial Expressions

Course Project – Semester II (Spring 2024)

Team Members:

Jana Aldubai 4010372

Salwa Shama 4010405

Samah Shama 4010403

Sana Shama 4010404

Instructors:

Dr. Ahmed Elhayek

May 12, 202

TABLE OF CONTENTS

TABLE OF FIGURS

Abstract

This report delves into the development automated usability testing systems, utilizing the capabilities of computer vision (CV) to detect user emotions and track mouse movements as they perform tasks. It begins by outlining the problem at hand then presenting the proposed solution. It then offers a brief description of the algorithms used in this project. The report then examines both successful and unsuccessful design features, explaining why they work or don't. It also offers additional functions and improvements to the system. Then it mentions the contributions and changes in the code, along with explanations for these changes. In conclusion, the report summarizes the findings and emphasises the significance of the developed system in advancing usability testing methodologies.

Keywords: Computer Vision, Emotion Detection, Mouse Tracking, FER.

CHAPTER 1: INTRODUCTION

1.2 Problem Description

Usability testing is an essential test conducted during the Software Development Life Cycle (SDLC). It is used to assess how easily and effectively users can interact with a software product, such as a website or mobile application. This testing typically involves observing real users while they are attempting to complete certain tasks within the software product to gather feedback and insights about the system from the user's point of view. Usability testing is crucial because it helps developers uncover issues that can hinder user experience during the use of the system, such as inappropriate labeling, complex design for components, and more. However, many problems can arise in this type of testing, such as users being unable to clearly describe their feedback or there being biased interpretation in results, which makes it ultimately a subjective measure.

1.2 Proposal Solution

To address these issues, computer vision can be applied. Basically, we are planning to utilize emotion detection, which builds on computer vision concepts, and mouse tracking in our proposed solution. The idea is to detect users' facial expressions while performing tasks and map these emotions to usability levels (easy, difficult). To elaborate more, if a user's emotion is anger, we can consider the system not usable, and so on. In terms of determining the usability of the system by mouse tracking, we will build a reference path of the mouse movement according to the instructions and compare it with the actual path the user followed to perform the task. This path highlights the steps the user has taken; the closer the user's path (steps) is to the reference path (steps), the more it indicates that the system is usable. This approach helps accurately assess if the software has ambiguities in executing the target tasks and measures the usability of the software in an objective manner.

CHAPTER 2: COMPUTER VISION ALGORITHMS

2.1 Algorithm A- Mouse Track in Real Time

To implement real-time mouse tracking, we utilized the "pyautogui" library, which provided us with a convenient way to access the mouse position in relation to the screen. We collected these positions at regular intervals and stored them in files for further analysis.To visualize the mouse paths, we converted the stored positions into white dots in the image. This approach allowed us to represent the paths visually, making it easier to observe and interpret the data. Storing the paths as images was particularly useful when comparing paths of unequal lengths as well. Once the paths were converted into images, we proceeded to compare them with a standard path. We employed the XOR operation to calculate the percentage of mismatches between the tracked paths and the standard path. This measure of mismatch served as an indication of the usability or accuracy of the mouse tracking system, Figure 1 shows mouse tracking outputs.

a. Reference path

b. User’s path

c. XOR output

Figure 1- Mouse Tracking Outputs

To sum up, the following lines summaries our mouse tracking algorithm steps:

- We collected mouse positions at regular intervals and stored them in files.
- The positions were converted into white dots in an image to facilitate comparing paths of unequal lengths.
- We compared the tracked paths with a standard path using the XOR operation.
- The percentage of mismatches indicated the system's usability and accuracy.

2.2 Algorithm B- Emotion Recognition Using FER Model

The algorithm starts by capturing video from the camera, processing each frame, and converting it to grayscale. This helps us to reduce image dimensions and speed up processing. In addition, we believe that it could reduce the effect of variation in light, which is considered one of the challenges that most computer vision algorithms face and enables us to obtain more reliable results.

Next part of the algorithm, which is considered a super important piece of the algorithm, is applying face detection and considering this area our interesting area to apply emotion detection on it. By doing this, we reduce our analysis scope and avoid unnecessary computation, besides reducing incorrect prediction of the emotion. We achieve the face detection part by employing the Haar Cascade Classifier, which is a well-known library in OpenCV. The Haar Cascade is a machine learning-based technique where a cascade function is trained using numerous positive and negative images. What makes this algorithm special is how fast it performs the process, which depends only on addition and subtraction operations to extract the interesting features.

Then, we apply emotion detection by using the FER (Facial Expression Recognition) model, which is a pre-trained model that detects seven different types of emotions: happiness, neutrality, sadness, anger, surprise, fear, and disgust. So, the input for this black box model is the cropped image that includes the detected face, and the output is the seven emotions with their probabilities. We can specify the emotion that represents the frame among these previous seven emotions output by choosing the emotion with the highest probability.

The last part, which is mainly related to our system for measuring the overall usability of a software product, involves mapping between the output emotion and the usability level. In our case, we have two levels: easy and difficult. We achieve this by mapping each of neutrality and happiness emotions to the easy level, and both sadness and anger to the difficult level. Then, we store the usability level for each frame, and the most apparent usability level in the task will be considered the overall usability of the system.

To sum up, the following line and Figure 2 summaries our emotion detection algorithm steps:

a. Read the frame from the video and apply some preprocessing steps to enhance the image quality.
b. Detect the face from the pre-processed frame to consider it our interesting area.
c. Apply emotion detection to the interesting area.
d. Map the output emotion to usability level and save the result.

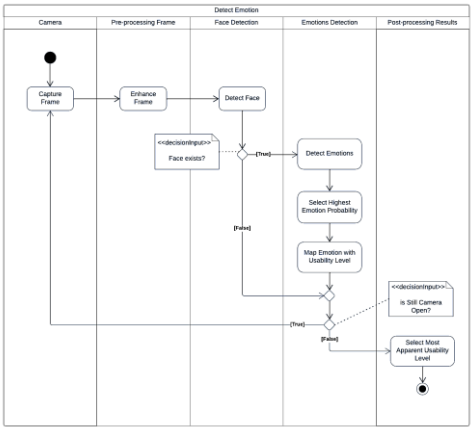

Figure 2 - Activity Diagram for Emotion Detection

This algorithm provides three primary outputs: Firstly, it offers a real-time video stream that highlights the user's emotions as they perform the task as shown in Figure 3. Secondly, it generates a real-time plot that depicts the evolving level of difficulty throughout the task, this depicted in Figure 4. Additionally, the algorithm displays various statistics. These include the breakdown of the percentages of each emotion that the user expresses during performing the task. Moreover, it presents the usability testing results for individual users, shedding light on their perceptions of task difficulty—whether they found it easy or challenging. Finally, the algorithm also summarizes the collective usability testing outcomes of all users who performed the same task, this feature is illustrated in Figure 5.

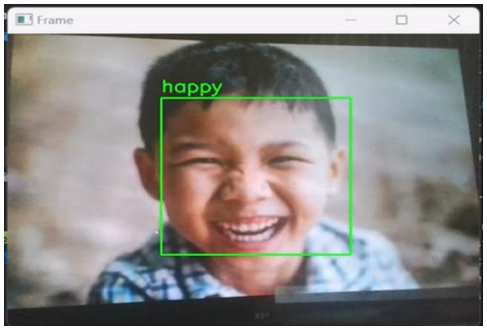

Figure 3 - Real Time Emotion Detection

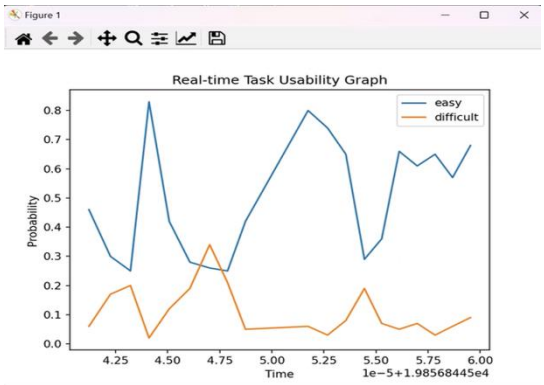

Figure 4- Emotion Tracking Graph

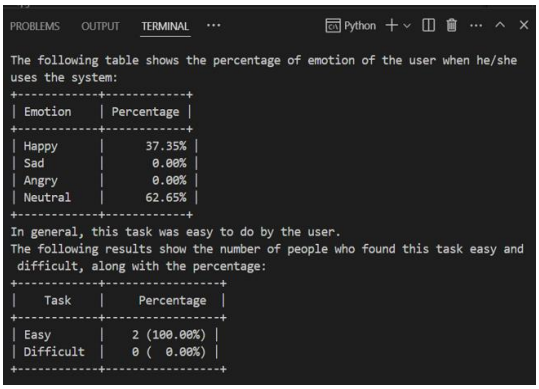

Figure 4 – Statistic for Emotion Analysis

CHAPTER 3: ANALYSIS AND DISCUSSION

This chapter is the thunder of this report. We will discuss what we did in this project by going through the following topics: Successful Design Features and Reasons, Unsuccessful Design Features and Reasons, Additional Design Features and Improvements, Reasons for Applying Some Changes and Change's Reason that Resolved the Issue.

3.1 Successful Design Features and Reasons

The successful design feature in mouse tracking is the utilization of the XOR operation to compute the mismatches between two paths. While other difference measures or similarity measures could have been used, these methods would have required the paths to have similar lengths. To overcome this constraint without adding fake points, the XOR operation proved to be a valuable solution.

In terms of successful design features for emotion detection, let us first remind you about our approach. We have two pre-trained models: a Haar cascade classifier for face detection and the FER (Facial Expression Recognition) library for emotion detection. Upon detecting a face in the preprocessing video frame, the code draws a rectangle around the face region and extracts face dimensions for further analysis. Using the FER library, it then identifies the emotion being expressed on the detected face. This emotion is displayed. The code continuously loops through frames from the video stream, detecting faces, analyzing emotions, and displaying the emotion accordingly.

So, our successful design that we have here and implemented in our algorithm is applying face detection before emotion detection, as we mentioned previously. That helps us specify the area of interest and crop this area to feed it to the emotion detection. This approach aims to localize the problem and reduce incorrect predictions by avoiding mismatching, especially when there may be cluttered scenes in the user's background. After this step, we achieve high accuracy with fewer computational operations. Therefore, including the Haar cascade classifier for face detection is a must part in our algorithm.

Another successful feature, which is not limited to our case only but applies to all computer vision cases, is the preprocessing step. It helps increase the accuracy of the model and handle challenging situations more easily. In our case, we handle light variation by converting the image to grayscale before applying any detection techniques.

3.2 Unsuccessful Design Features and Reasons

The first attempts were made to extract mouse tracking from video, and we did this in two different ways. First, we tried feature matching between the mouse shape and finding the location with the highest correlation in different frames. Another way was to use optical flow

to capture mouse movements. However, both methods failed to track the mouse movement. We believe this was due to several factors, such as sudden changes in mouse position and shape, as well as difficulties in distinguishing the mouse shape, filled with white colour, from the background.

In terms of emotion detection, we initially implemented emotion prediction directly in the frame without any preprocessing step because we thought we didn't really need it, assuming the model is smart enough to detect emotions on its own. However, that's not true. Even if the model is smart, it doesn't guarantee accurate emotion detection, as some situations are out of the model's control. For example, poor brightness due to using a bad camera can lead to a loss in features, which is an essential part in building any robust algorithm, which can handle it easily by preprocessing techniques. Moreover, applying emotion detection directly to the frame is not a good idea; the model might get confused with the scene or background, especially if it's cluttered, and might detect emotions even if there's no person present. Therefore, we must limit the scope to first detect the face and then detect the emotion, a lesson we learned the hard way.

3.3 Additional Design Features and Improvements

In terms of mouse tracking, we want to attempt to break down the path of the task into stages and incorporate time as a factor in computing differences between paths. This will provide a deeper understanding of how the task is performed. Additionally, we would like to compare several paths resulting from multiple attempts by the same user on the same task. By calculating the improvement percentage over time, we can obtain more accurate results regarding the system's usability in performing a specific task. Unimproved paths over time indicate a steep learning curve for the user.

In terms of the emotion detection algorithm, we can improve it by adding more preprocessing techniques, such as enhancing the image contrast by histogram equalization, and applying smoothing techniques to remove any noise from frames, especially because we are dealing with real-time streaming. Additionally, in the last steps, instead of storing the usability level for every frame, we can store usability levels based on groups of sequences of previous and next frames to avoid noise prediction by the model and using the voting technique idea.

In general, to improve the system as a whole, not limited to specific algorithms, we can incorporate multiple AI models, such as analyzing the user's voice when they perform usability testing tasks, to gain a better understanding of user emotions from different perspectives and forms rather than just facial expressions. We can log the recognized emotions and timestamps for further analysis. This data could be used to understand user behavior patterns.

3.4 Reasons for Applying Some Changes

In terms of mouse tracking, we have made a decision to shift our approach from video analysis to real-time analysis. This change aims to provide us with more informative data that can enhance our ability to perform this task effectively. It's important to note that this transition does not solve the main problem directly, but it enables us to prioritize and concentrate on the second key feature, which is emotion detection.

For emotion detection, applying face detection before emotion detection enhances the robustness, accuracy, and efficiency of emotion recognition, especially in real-world applications where images or video frames may contain complex scenes or multiple objects that affect the model accuracy. Additionally, converting to grayscale simplifies face and emotion detection by reducing computational complexity, especially when dealing with CPUs and streaming frames flow, not a single image.

3.5 Change's Reason that Resolved the Issue

Several changes were made to resolve numerous issues encountered during the implementation of this solution. These changes include improving preprocessing techniques to enhance image brightness. Additionally, we experimented with different combinations of measures to assess the differences between paths or measure their similarities. The problem with the previous methods was their inability to compute the result when two paths had different lengths. To address this issue, we generated or filled certain points to meet the length constraint. We realized that there was a better way to handle this problem. Consequently, we changed our approach and drew the different paths in two images, followed by performing the XOR operation to identify the differences. Furthermore, we made an additional change to reduce the possibility of incorrect matching by limiting the search area.

CHAPTER 4: CONCLUSION

4.1 Conclusion

To sum up, we applied our knowledge of computer vision to our project. We began by writing pseudocode to implement our algorithm for both mouse tracking and emotion detection, going through a trial-and-error approach to fine-tune the algorithm and some fundamentals of any computer vision application. Throughout the project, we investigated the most challenging tasks that most computer vision algorithms suffer from and tried to handle them or documented how to handle them to improve our algorithm. Examples include converting to grayscale frames to handle light variation, detecting faces to localize the problem and make efficient real time solution, and using voting techniques to get more reliable results and avoid noise, among others. We tried to utilize the techniques and concepts we learned in our lectures and improved our understanding through hands-on application. Overall, this project allowed us to gain practical experience and reinforce our understanding of the course material.

4.2 References

[1] R. Szeliski, Computer Vision: Algorithms and Applications. Cham: Springer Nature, 2022.