Summarization of focused objects in Eye-tracking toolkit through video mining approach

ABSTRACT

In these days, technology is being used in almost every field of life. Of these innovations, Eye-tracking toolkit is one of the best innovations to operate and control the devices around us with the help of our eyes. It is also helpful in the field of marketing and behavioral economics to understand the behavior of a consumer. This toolkit measures the point of gaze or motion of eyes to capture the behavior of eyes relative to the head. It comes with specialized glasses which are synchronized with human eye, computer, and eye tracking device to record the activity of a person. The recording activity of a consumer is then watched and analyzed by the stakeholders. In addition, several graphs are generated from the recorded video such as heat graphs. However, this toolkit has one major limitation that is the time to analyze the recorded video of user activities. This is because the stakeholders watch the whole video to analyze the user activity. Though, the eye-tracking software provides some useful graphs (such as heat graph). However, these graphs provide less user-friendly information. To overcome this limitation, the aim of the project is to analyze the whole recorded video and mine it to dig out the useful information from it. In addition, our proposed solution will take video as an input and provide the summarized information comprising duration of all those objects which were focused by the customer during the activity. We believe that our proposed solution will save a lot of video analysis time and efforts for the stakeholders and provide useful information in minimal time.

INTRODUCTION

Technology has a great impact on our lives, it has not only facilitated but improved our lives too by introducing a wearable technologies, which can be worn to know the information about consumer's activity time to time. The eye-tracking toolkit is also one of the wearable technology which lets us to track the focus of consumer's activity that which products he or she has focused. This eye-tracking toolkit Figure 1 is a portable pair of glasses having embedded sensors and cameras. This device is synchronized with user's iris to record the movement of eyes with respect to the direction of forehead. In order to know the region of focus of iris, this toolkit encircles the focused region with red unfilled circle throughout the video recording as shown in Figure 2. Here, it can be observed that the eye-tracking toolkit users focused on many Sunsilk shampoo products. However, the user has focused more on the product which is in golden color. The recorded video is usually watched by the stakeholders to analyze the activity of consumer that what objects he or she has focused during his or her activity. For instance, take an example of Tesco shopping center. This center has thousands of products shelfed in store. Several customers come on daily basis to visit the Tesco store and to shop. Though, these customers do not buy all the products. However, they may look or show interest in many products. Thus, with the help of eye-tracking toolkit, the Tesco stakeholders can illicit the interest of their consumer.

This toolkit can be beneficial in many application areas other than super stores like Tesco. These application areas include, learning, designing, marketing, driving, and performing actions with the gaze of eyes as a channel of communication for the people who can't express themselves. In addition, this technology is used majorly for the research purpose to study the behavior of the human mind in various fields, which require the direction of eyes to perform anything. For instance, if there is a player playing basketball wearing these glasses then the analysis will tell us his behavior towards the game, the way he played and tackled his opponents. Similarly, this toolkit can be used by a driver to see his reaction when he faces any hazard. Moreover, in terms of marketing research, it helps us to determine the consumer decision making by finding the all those factors of product (such as, color, shape, ingredients and price) which made him to take the decision and also to select the appealing products by analyzing the number of times consumer has focused the products.

For analyzing the toolkit users' activities, this toolkit has many useful features such as, heat graphs to represent the toolkit users' activities analysis. Heat graphs work on the basis of color intensities, the more color intensity the more focused objects we have and it's shown in Figure 3. However, these heat graph are incapable of exposing the exact object where the user put attention. Hence, in this projects we overcome this issue and develop an application to enhance the existing capabilities of eye-tracking toolkit to get more useful analyses results. Our proposed application will take the recorded video as an input and apply various image processing techniques on the given input to extract and summarize the focused objects. In addition to the focused objects, our developed application is capable of retrieving the time duration of those focused objects as shown in 6. It is believed that our developed application will be an added value for eye-tracking toolkit. In addition, it will save the enormous time of researchers by analyzing the video in couple of minutes. Finally, it is believed that our application will produce more meaning analysis results for the researchers.

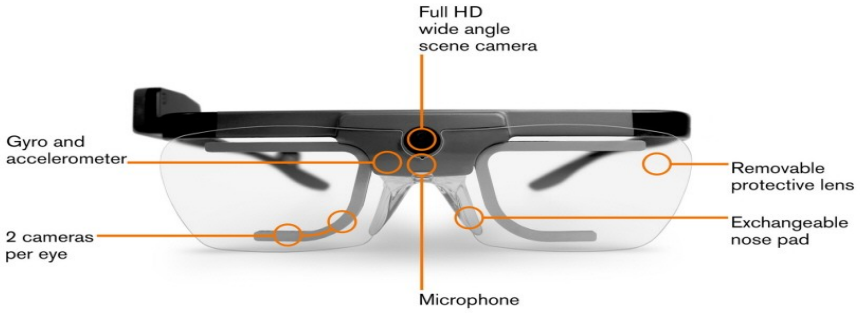

Figure 1 Eye-tracking toolkit, which captures the focus of eyes.

Figure 2 shows the region of interest encircled that which object consumer has focused.

Figure 3 heat graph showing the focused objects with color intensity.

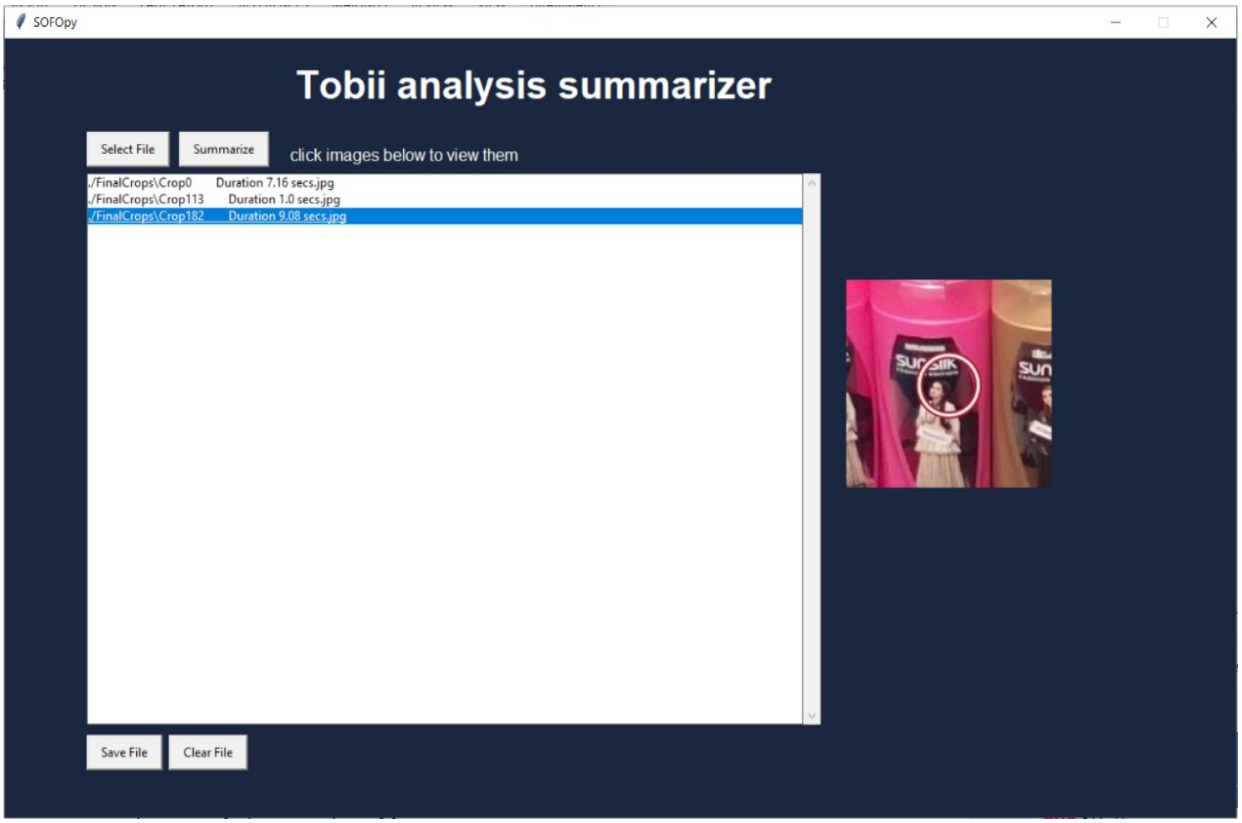

Figure 4 Desktop application, it shows the summary of focused objects along with their duration.

Problem Statement

The eye-tracking toolkit provides the software to visualize its video analysis with the help of heat graphs to study consumer behavior. Heat graphs are based on the color intensities which is not a user-friendly approach. In addition, to find the focused objects in the recorded video and the duration of those objects is time consuming. Hence, an intelligent system is needed in this eye tracking toolkit to analyze the video analysis, to report the focused objects and to calculate the duration spent on those objects within the minimal time.

METHODOLOGY

Methodology for summarization of focused objects

To summarize the focused objects from eye-tracking toolkit generated video we used the following methodology.

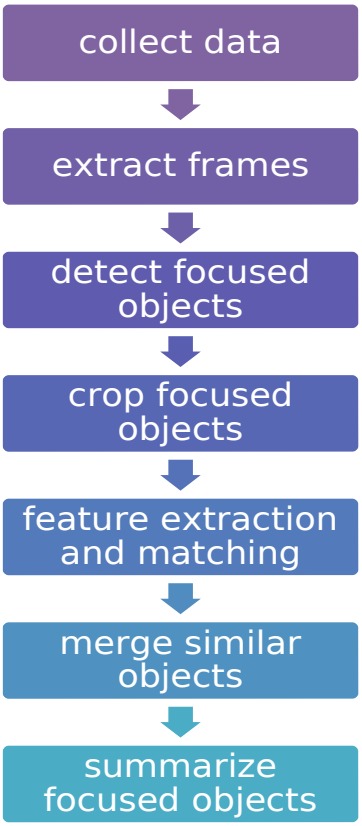

Figure 5 Summarization methodology

Collect data

First step of our methodology is to collect the data and for that we met the focal person of the behavioral lab of the university to generate the video from the eye-tracking toolkit as a sample to work on it.
Toolkit generated video contains

- Encircled ROI
- 25FPS
- Mp4 format

Extract frames

At second we extract the frames from the video with the help of Opencv methods which are below

```
video = cv2.VideoCapture(sourcefile)
while (True):
    ret, frame = video.read()
```

Each frame has 1920x1080 resolution, which is about 516kb in size.

Detect focused objects

After frames extraction we find the region of interest by finding the encircled area in frames and for that we have used Hough circle transform function of Opencv to find the circles. Function takes a few parameters like picture in which to find the circles and the size of parameter and radius to find the circles in it. When where ever it finds the circle in picture according to given parameters it draws the outer layer and center of circle to locate it.

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1, frame.shape[0] / 1,
                           param1=200, param2=10, minRadius=20, maxRadius=35)
if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0, :1]:
        # draw the outer circle
        cv2.circle(img, (i[0], i[1]), i[2], (0, 255, 0), 3)
        # draw the center of the circle
        cv2.circle(img, (i[0], i[1]), 2, (0, 0, 255), 2)
```

Crop focused objects

When the region of interest is found then we crop that region with 200x200 resolution where each crop is about 16kb in size.

```
x = i[0] - 100
y = i[1] - 100
crop = frame[y:y + 200, x:x + 200]
```

Feature extraction and matching

In this step we compare all of the cropped images with each other to find out the similar ROI and we do this by extracting the features from images and then compare those features and if the number of features of two pictures meet the certain threshold then we label them as similar otherwise dissimilar.
For features extraction we tested a few algorithms like SURF and SIFT and we found SURF very much helpful for our work.
Results of SIFT and SURF on a picture of 200x200 are mentioned below.

	SIFT	SURF
Key points	253	303
Time	0.029963254928588867	0.05496788024902344

Table 1 Results of SIFT and SURF

So we decided to use SIFT because it is scale invariant.

For features matching we went through a few algorithms mentioned below

- BFMatcher
- FlannBasedMatcher

BFMatcher works like brute force to match all of the points which was not a precise and fast approach where as FlannBasedMatcher used nearest neighbor approach to find the similar descriptors that's why we used FlannBasedMatcher approach.

Merge similar objects

By feature extraction and feature matching we find the number of similar crops. Similar crops help us in finding the duration of focused objects by dividing the count of similar objects with the frame rate of video which is 25fps.

```
for i in similar_images_count.keys():
    if len(i) > 6:
        ind = i
        value = ind[-6]
        s_value = ind.find('p', -9, -3) + 1
        e_value = ind.find('.', -5, -2)
        s_index = (int(ind[s_value:e_value]))
        d = similar_images_count[i] / 25
        if d > dur:
            name = "./FinalCrops/" + str(d) + " secs.jpg"
            tmp_img = cv2.imread(i)
            cv2.imwrite(name, tmp_img[:])
```

Summarizing focused objects

After merging the similar objects and finding their duration we summarize in a form of pictures named with the number of duration as jpg.

Use Case Model

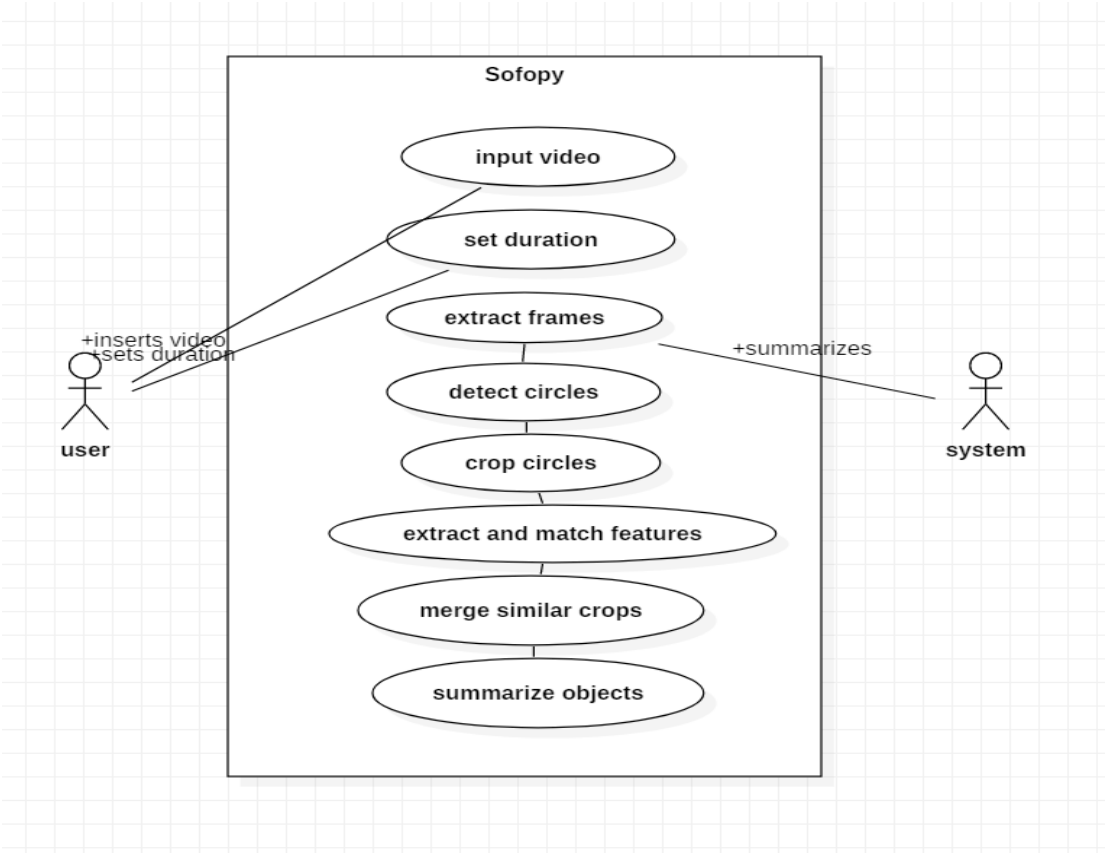

Figure 6 Use case

Sequence Diagrams

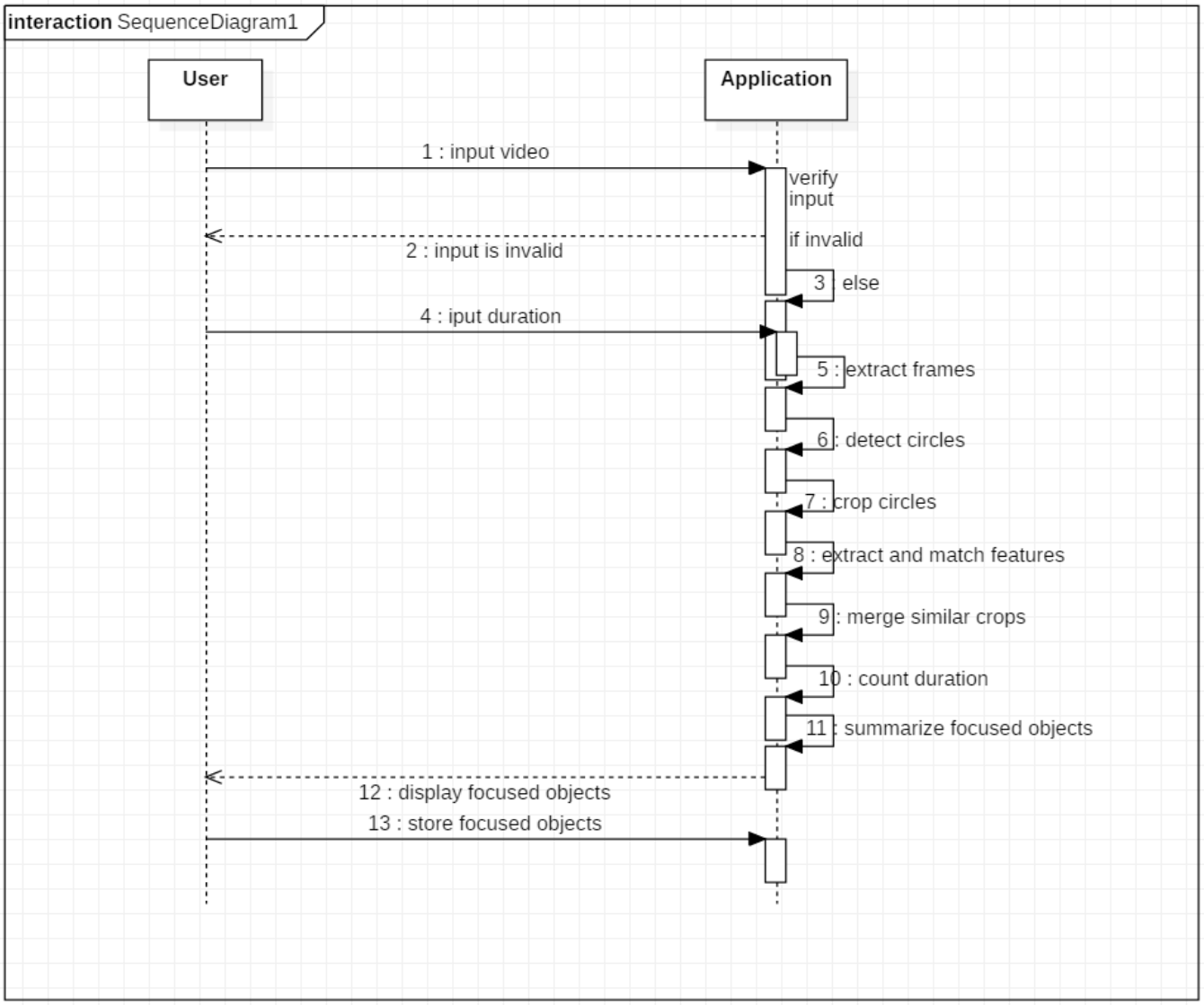

Figure 7 Sequence diagram

RESULTS

Results include the evaluation of interface and its functionality.

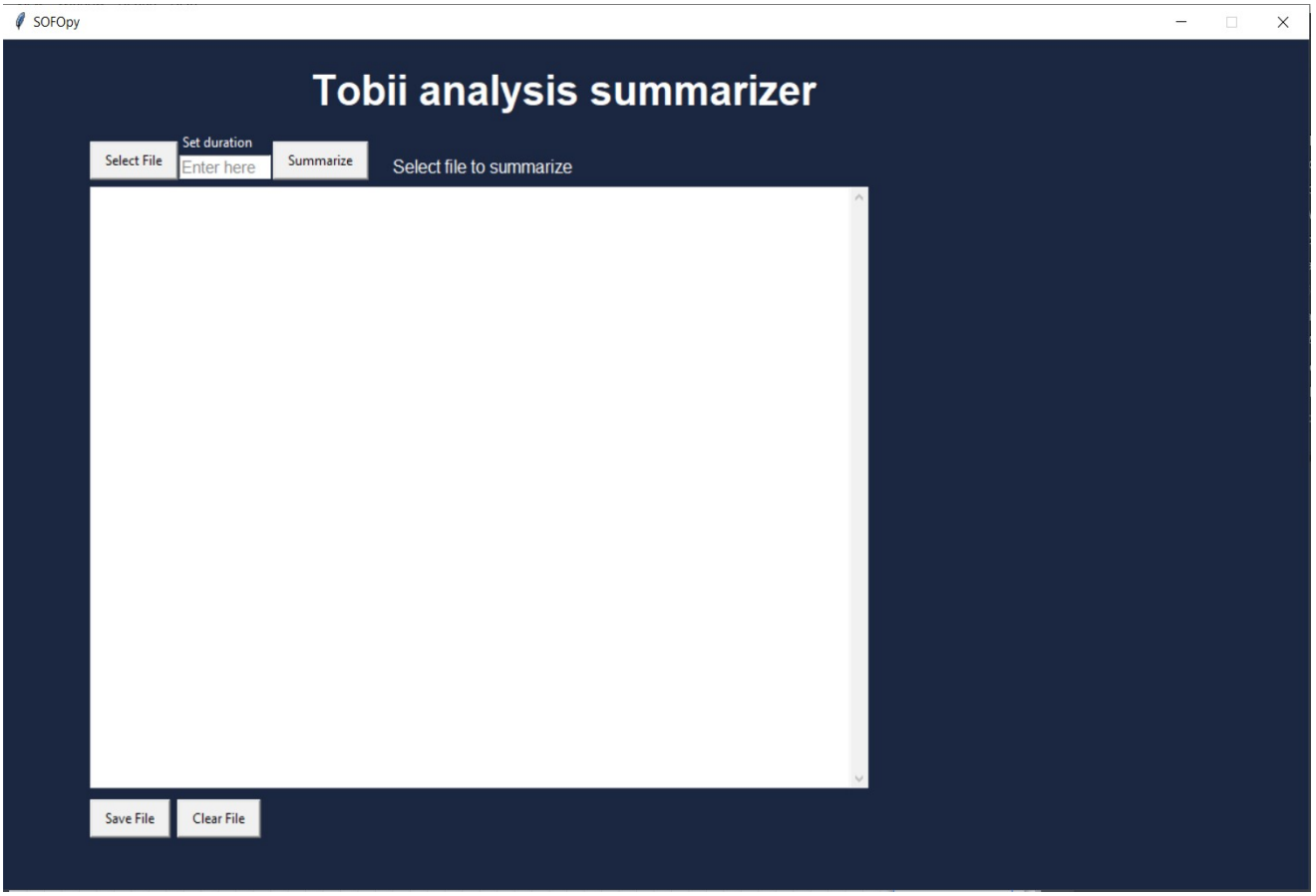

Figure 8 shows main interface screen

This is the main screen of desktop application for a user to interact with. First of all a user has to select the file and then set the duration for the required analysis and then click the summarize button the start the process.

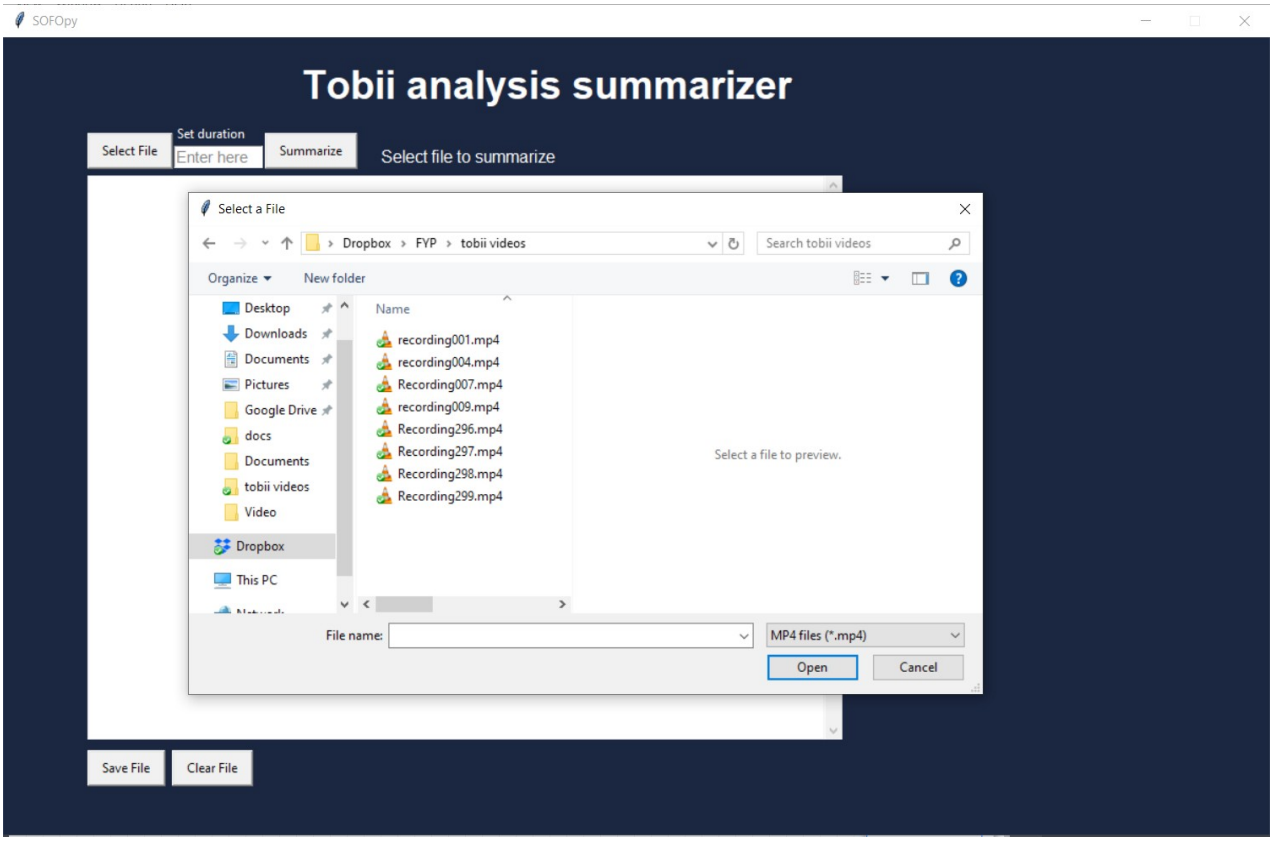

Figure 9 Select file screen

In above figure user clicks the select file button to select the file from local directory.

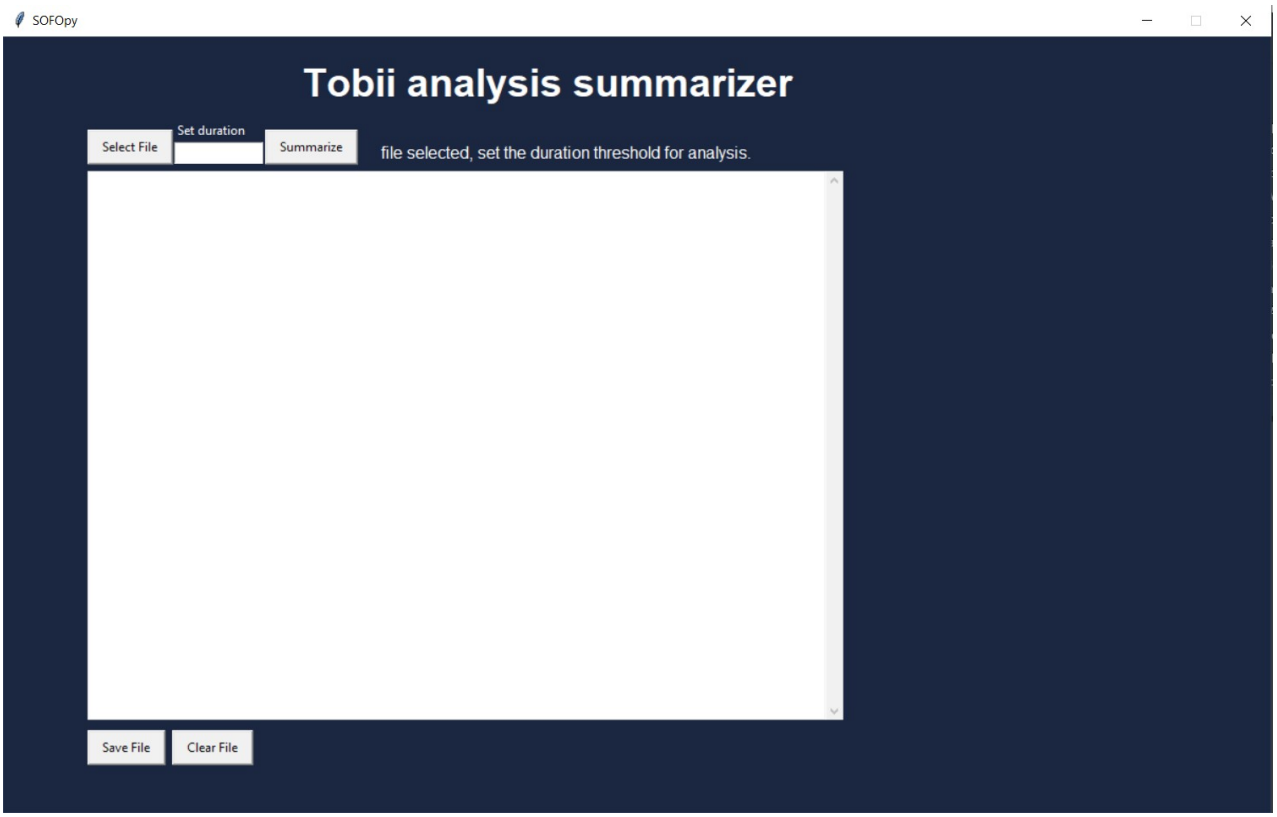

Figure 10 Set duration screen

In above picture user has to set the duration after selecting the file, when the file is selected it shows the status in a label beside that file is selected.

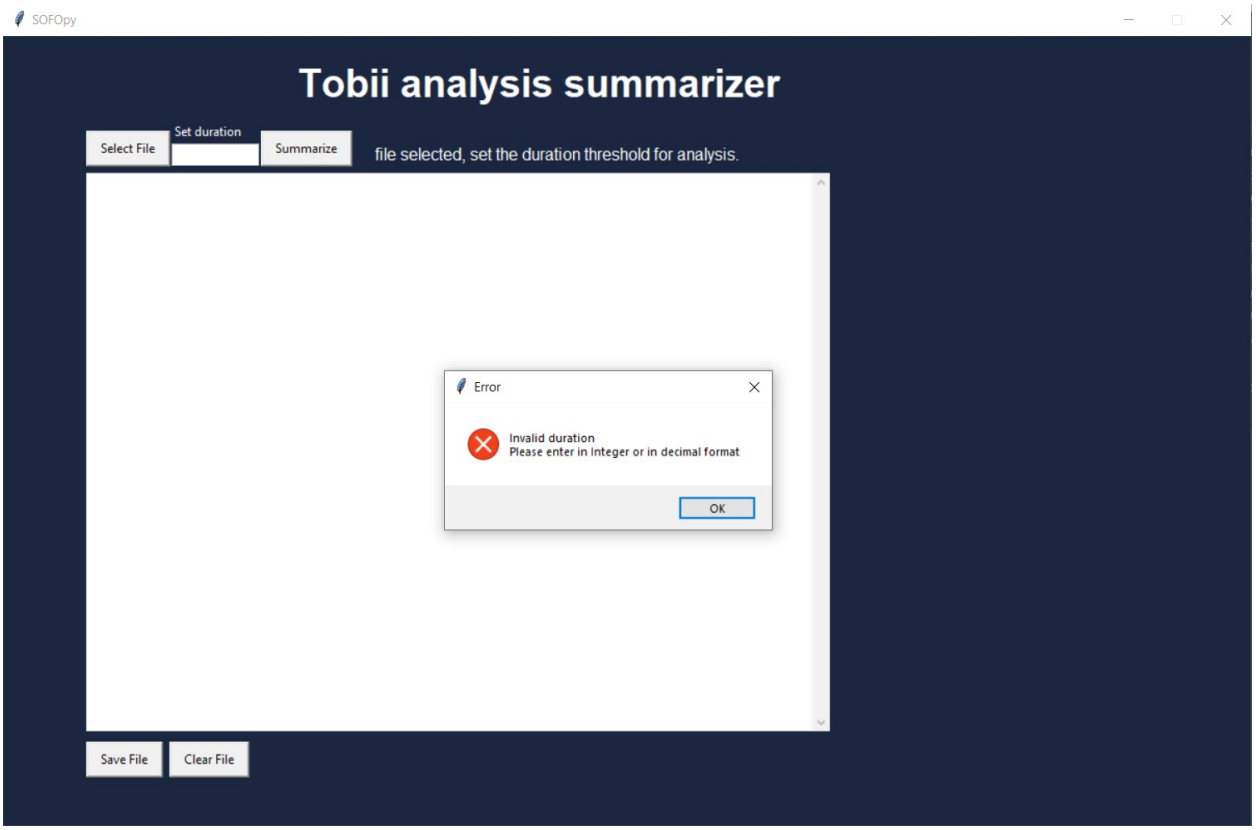

Figure 11 Exception screen

This above screen gives the error because we have not set the duration for the application to start the analysis.

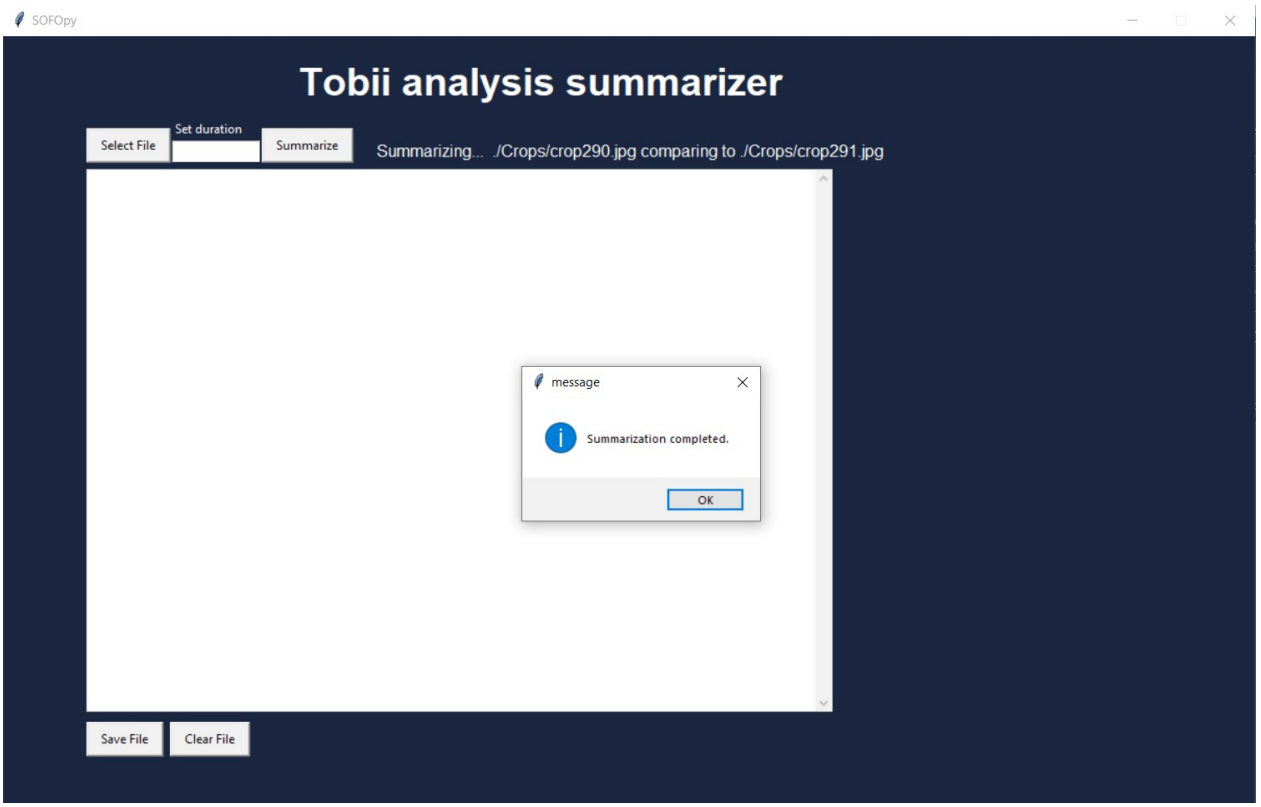

Figure 12 shows when summarization is completed

In above figure a message is given when the summarization completes.

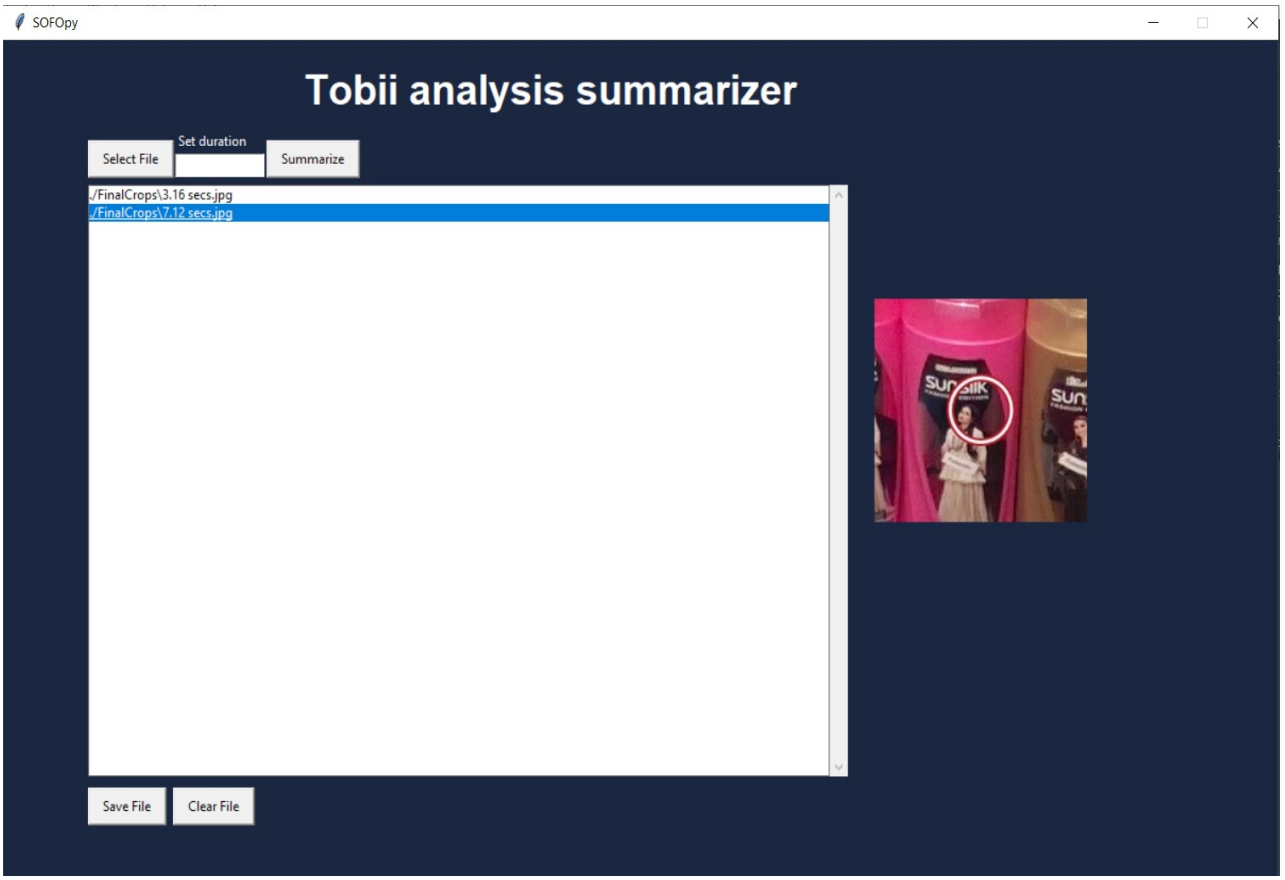

Figure 13 shows focused objects and their duration

In above figure after having summarization process completed, we get the analysis with the duration as name with .jpg extension. By clicking on the link we can see the focused object on the right side.

Summary

Product scope is based on following mentioned functions

- It takes video as an input
- It requires threshold to set duration
- After both input requirements it summarizes
- It stores results.

References

[1] Tobii Pro Glasses 2 wearable eye tracker, https://www.tobiipro.com/product-listing/tobii-pro-glasses-2/, last accessed 10/01/2019

[2] Hough Circle Transform from OpenCV-Python Tutorials's, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghcircles/py_houghcircles.html, last accessed 22/2/2019

[3] Introduction to SIFT (Scale-Invariant Feature Transform), from Opencv-python tutorial's documentation, https://opencv-python tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html, last accessed 4/5/2019

[4] 24.1. Tkinter –Python interface to Tcl/Tk, https://docs.python.org/2/library/tkinter.html, last accessed 20/6/2019