Mobile App for Assessing Hemifacial Spasm Treatment Response Using Machine Learning

João Luiz Sotini da Silva, Carlos Marcelo Gurjão de Godoy, Tammy Hentona Osaki, Midori Hentona Osaki, Cristina Yabumoto and Regina Célia Coelho

January 15, 2023

Mobile App for Assessing Hemifacial Spasm Treatment Response Using Machine Learning

J. L. S. da Silva[1], C. M. G. de Godoy[1], T. H. Osaki[2], M. H. Osaki[2], C. Yabumoto[2] and R. C. Coelho[1]

[1] Science and Technology Institute, Federal University of São Paulo, São José dos Campos, SP, Brazil
[2]Department of Ophthalmology, Paulista School of Medicine, Federal University of São Paulo, São Paulo, SP, Brazil

Abstract— **It is challenging to assess hemifacial spasm (HFS) patients as they exhibit high-frequency and heterogeneous anomalous eyelid movements. This study aimed to develop an application for a smartphone to objectively determine the eyelid movements frequency so that treatment responses in these patients can be assessed accurately. The smartphone application was developed mainly using Python, a prominent and broadly used programming language focused on machine learning and data science tasks. The application can precisely predict the movement of the patient's eyes using an SVM regressor and classifier. The results are plotted for better visual inspection by using data visualization techniques. Thus, the application enables a continuous study of each patient using an integrated database in Google spreadsheets, which could better track the results of each treatment response. The application showed to be an efficient method to identify and represent eyelid movement occurrences in patients, objectively measuring the eyelid movement frequency and, thus, assessing the treatment response in patients with hemifacial spasm. This system could enable customized and fine adjustments to botulinum toxin doses based on each patient's needs.**

Keywords— **Machine Learning, Data Science, Eye Aspect Ratio, Hemifacial Spasm, Application**

I. Introduction

Hemifacial spasm (HFS) is a neurological condition characterized by involuntary, tonic, and clonic spasms of the muscles innervated by the ipsilateral facial nerve. This condition is usually secondary to facial nerve compression at the root exit zone caused by an aberrant artery [1, 2]. Spasms usually originate in the periocular region (orbicularis oculi muscle) before affecting the midface and lower third of the face over months to years. Most cases present with an affected and a non-affected side in this condition [1]. Botulinum toxin-A (BTX-A) injections in the affected muscles are considered the treatment of choice for this condition [1-4], reducing facial spasms. The efficacy of botulinum toxin administration is generally assessed using clinical rating scales, such as the Jankovic rating scale. [3, 4] Although grading systems facilitate the classification of clinical symptoms, these tools are not accurate in assessing treatment response. [5]. Furthermore, in HFS, it is challenging to impartially assess the eyelid spasms due to high-frequency anomalous eyelid movements [6]. Previously employed objective approaches to assess this condition relied on less accurate systems based on manually reviewing videotapes or other indirect approaches [7, 8]. An accurate system has been described to assess this condition's eyelid movements fairly (spontaneous blink + anomalous eyelid spasms) [5]. However, that system is complex and cannot apply in clinical practice.

This study aimed to develop an application for a smartphone to objectively determine the eyelid movements frequency so that the therapeutic effect of botulinum toxin can be practically assessed in patients with HFS. To the best of our knowledge, no previous smartphone application has been reported for this purpose.

II. Materials and methods

A. Libraries and Data Sources

The data was collected in 2-minute videos from subjects who agreed to participate in the study. Patients with HFS and normal subjects from the Division of Ophthalmic Plastic Surgery, Department of Ophthalmology and Visual Sciences, Federal University of Sâo Paulo, were recruited to participate. Python manipulated and extracted information from the videos. It is a commonly used programming language for machine learning and data science in general [9]. Python contains some libraries used to build the application: *OpenCV*, *Imutils*, *Dlib*, *Matplotlib*, *Scipy*, *Streamlit*, *gspread*, *secrets*, *datetime*, *Pillow and Seaborn*. Each library had a different use. *OpenCV* was responsible for interacting between the webcam and the script. It also applied to image transformation (thresholds and resizing) and frame manipulation. *Imutils* applied for manipulating the data points that corresponded to the landmarks. *Dlib* contained the model that detected the eye points based on the 68 landmarks (with HOG [10] and SVM [11]). Using *Scipy*, we calculated the area within the eye, and with *Matplotlib* the data was plotted to visualize the results of the tests.

The model was trained on the public online database Ibug 300W ICCV 2013 [12], which was part of a face identification challenge. It contains four different datasets: Helen, LFPW, AFW, and IBUG, corresponding to more than 11.500 images and their labels. These datasets were also manipulated to reduce the size of the XML files. It reduced from 68 to 12 points of data for each image (only the eye points were preserved). Using *Streamlit* library, we developed the web application that contains the machine learning model. That allows any person to use the application with a smartphone. *Gspread* library established a connection between the app and Google Sheets, where all the data collected was stored, as is an easy access platform enabled by Google to create tables that attend to the needs of this task. *Secrets* is a built-in library from Python used to create a unique ID for each patient to relate the registration database with the data collection database. *Datetime* library allowed to retrieve the day the data was collected. *Pillow* was used to opening the main image. Finally, *Seaborn* was used alongside Matplotlib to plot the line charts that describe the variation of the eye's area over time.

B.Machine Learning Script Structure

The script, built-in Python, is based on the algorithm of Rosebrock [13]. In his work, he utilizes a machine learning model which relies on the Histogram of Oriented Gradients (HOG). HOG is a descriptor that extracts features from an image based on a convolutional operation between kernels and the image. It models the resulting data in a concatenated vector of components of the normalized cell histograms from all the captured block regions. Such extractor is commonly used with Support Vector Machine (SVM), a supervised learning model that classifies and makes predictions based on regression analysis. Thus, the data points are located in the image after finding the features that result from the HOG extractor. The face detector is available by the *Dlib* library and is used to identify the 68 landmarks in videos or the webcam.

Based on that, the model was adapted to segment only the eye as a region of interest. As previously said, the XML data was manipulated so that only the 12 eye points were focused. Therefore, the data is collected in the form of lists, so it is reproducible as line plots to identify the patient eye aspect ratio over time.

This script structure is the main part of the developed app. It provides a friendly user interface, as it can be used on mobile and desktop devices without accessing any background code.

C. Eye aspect ratio function and measurement

The eye aspect ratio calculus was inspired by an article that focuses on the results of the SVM model in predicting facial landmarks [14]. It uses the Euclidean distance of the height and width of the eye to calculate its area, as shown in Eq. 1. This Eq. refers to the points of the eyes from p_1 to p_6 in Fig. 1.

$$EAR = \frac{||p_2 - p_6|| + ||p_3 - p_5||}{2 * ||p_1 - p_4||} \tag{1}$$

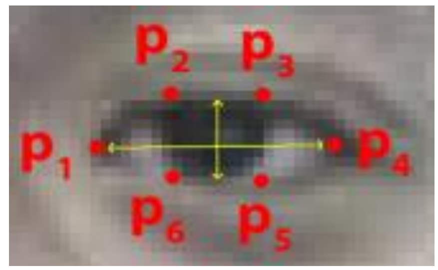

Fig. 1: Points of the eye used in Eq. 1.

The results are primarily constant with slight variations when the eyes are open. That is a very accurate method to classify between open and closed eyes, as the *SVM* model had an excellent performance in classifying the landmarks in many situations. This calculus is performed for both eyes as all types of blepharospasm disease can be measured.

The resulting values accumulate in a Python list so that it can be plotted into a line chart containing, for each eye, the eye aspect ratio over time. Fig. 2 is an example of how the results are shown.

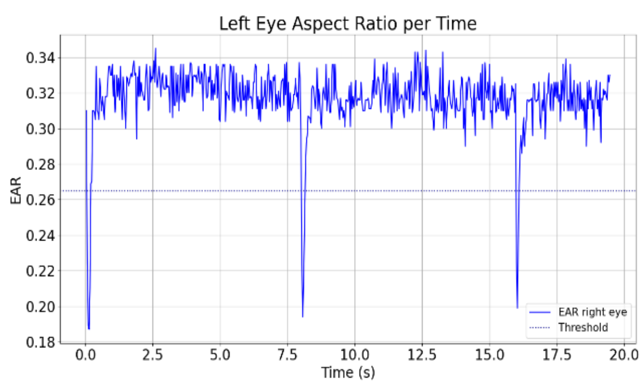

Fig. 2: Example of a line chart of an eye aspect ratio (EAR) over time.

A threshold was based on the average size of the person's eye to measure the EAR as an eye blink. For most cases, EAR rounds about 0.25, or 0.3. However, it depends on the eye size; thus, it needs to be fine-tuned. As for the fine-tuning, it was defined a function that retrieves a modified EAR to keep track of each patient threshold. The modified EAR, based mainly on the actual size of the eye [14], is calculated as the sum between the open eye size and the closed eye size, as can be seen in Eqs. 2 to 4.

$$EAR_{closed} = \frac{||p_2-p_6||_{min}+||p_3-p_5||_{min}}{2*||p_1-p_4||_{max}} \quad (2)$$

$$EAR_{open} = \frac{||p_2-p_6||_{max}+||p_3-p_5||_{max}}{2*||p_1-p_4||_{min}} \quad (3)$$

$$Modified_{EAR_{Thresh}} = (EAR_{Open} + EAR_{Closed})/2 \quad (4)$$

By using those equations, we obtain the proper EAR of the patient. After that, another counter starts after the value of the EAR is below the threshold. If the EAR does not rise for one frame, it is considered only one short blink is not counted. Therefore, it avoids the risks of getting false negative values.

D. Streamlit Application

The app was based on the *Streamlit*, a recently released library containing features to develop online web applications for mobile and desktop services. A user interface was created and divided into three sections: the home page, the analysis/login page, and the sign-up page. The opening page summarizes the project and how to use the application. Fig. 3 illustrates the app's initial page.

The analysis/login page (Fig. 4) leads to the app's main functionality, which is the eye recognition and evaluation using the *Dlib* facial landmarks detector. It contains a separator between pre and post-treatment evaluation and a history section. In that section, it is possible to visualize the patient's records as a data frame. In the Charts section, the patient or physician can select and visualize specific patient records plots. Finally, the sign-up page is where the user, a patient, or a doctor can register in the app to use its resources and features.

E.Google Sheets Database

As the app relies on retrieving patients' data for further evaluation of the symptoms and severity of spasms, it is inherent to store and relate this data to each patient who uses its services. Therefore, a simple and efficient solution was to create a Python automated Google Sheet that is updated

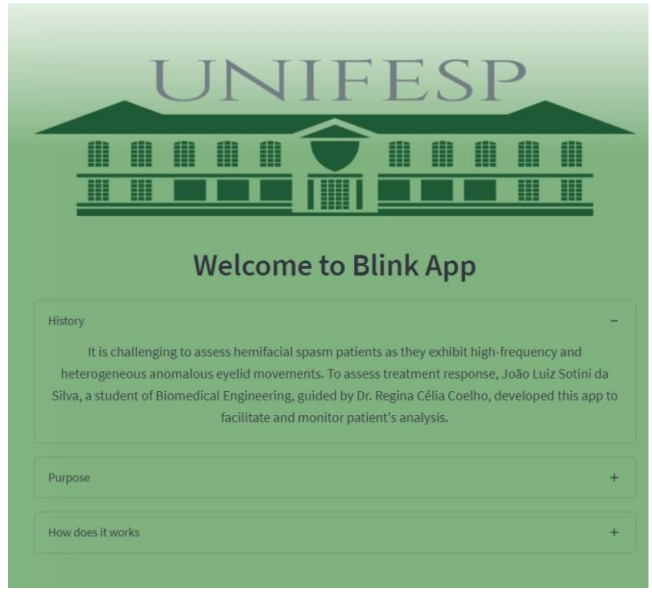

Fig. 3: The app interface.

Fig. 4: The analysis page.

whenever the user wants to store his data records. The library *Gspread* was used to access the Sheet called '*Person'*, which contains the main working sheet called '*Registration'* and the ID-related sheets, which will be explained later. The *'Registration'* working sheet contains the personal data, including login name, surname, city of residence, age, password, and ID. The user defines the login name and the password in the sign-up steps. The ID is an automatically generated random unique key. When the patient registers an account, the script also generates a new working sheet related to the user's ID (its name is the user's ID), which is exclusive to every new user who registers. It contains information about the date that the data was collected, the eye aspect ratio collected from the video from each eye (left and right eyes), the number of blinks that the person had during the analysis, the stage of the analysis (pre or post-treatment) and the threshold that indicates whether the patient has blinked or not.

III. Validation

To validate the efficiency of the application in recognizing blinks, the control group had their total number of blinks annotated for 2 minutes. After this, the results were stored in a Google spreadsheet and compared to the application results using the root mean squared error (RMSE), as the results are integers derived from a regression task. The results were normalized using the functions StandardScaler, and the evaluation metric used was '*mean_squared_error'*, both from *Scikit-Learn* [11]. The results are between 0.0 and 1.0. If the RMSE results are closer to 0.0, the model predicts effectively.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(y - \hat{y})^2}$$

where ŷ is a vector with the predicted values, and y is the ground truth value for each value analyzed. The result is the root of the difference between the predicted and the ground truth values divided by the number of values (value represented by n).

IV Results and Discussion

Fig. 5 illustrates a frame of a patient video and the predictions computed by the developed app. Figures 6 and 7 show the charts plotted inside the app concerning the same patient's eye aspect ratio to her pretreatment and post-treatment.

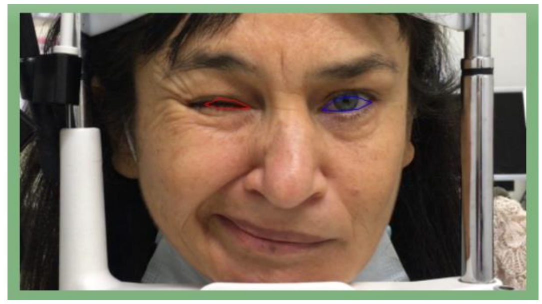

Fig.5: A patient video frame used in the app. The red and blue lines around the eyes are the predictions being computed in real-time.

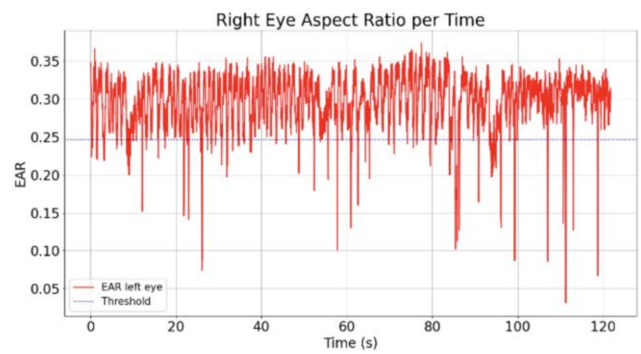

Fig. 6: Line plot of the EAR (Eye Aspect Ratio) over time and the threshold of a pretreatment patient's right eye (patient is depicted in Fig. 5).

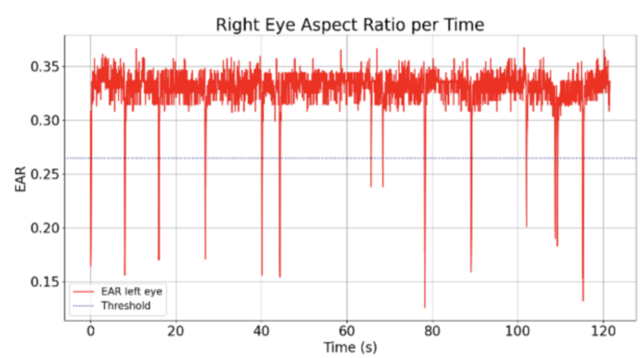

Fig. 7: Line plot of the EAR (Eye Aspect Ratio) over time of post-treatment patient's right eye (patient depicted in Fig. 5).

The results were compared to the control group's ground truth annotations (made by physicians). The results (Tab. 1) show that the mean RMSE of the normalized predictions compared to the normalized gold standard was, on an average, 0.0995. Concerning this mean RMSE outcome, it is possible to realize that the model succeeded in predicting the blinks of the control subjects.

Table 1. Annotations and Result

Control Patient	Annotated Blinks (Dr. M. H. Osaki)	Annotated Blinks (Dr. T. H. Osaki)	Predicted Blinks
1	129	127	127
2	58	58	58
3	18	18	14
4	97	97	97
5	69	68	70
6	39	40	36
7	69	70	65
8	65	63	56
9	58	58	58
10	33	32	23

The app can predict most blinks, although the EAR threshold is hard to define in every situation, considering that some patients had semi-blinks, which can configure as blinks in medical terms. However, it is not accounted for by the calculation, as it mainly measures the fully closed eyes. Nevertheless, that is not a problem, as the eyelid movement is accounted for in the chart, being possible to observe the half-blinks (see control patient number 10 in Table 1). Another advantage of the app developed in the present study is its high accuracy in detecting movement. This advantage is not possible with conventional human-based video review assessment. This app could enable more accurate and customized dose adjustments for each patient due to its accuracy in detecting eyelid movement.

The drug duration is another outcome used to assess the effect of botulinum toxins. The objective duration associated with BTX-A treatment has been assessed indirectly in previous studies that analyzed changes in eyelid morphometric patterns in HFS patients over four months [8]. Patients were evaluated at baseline and at the 15-day and 2-, 3-, and 4-month time points. The return of the parameters to their pre-treatment status at the 4-month time point implies that the studied period during which these changes took place represents the BTX-A duration. Similarly, the smartphone app presented herein could indirectly facilitate the unbiased assessment of the BTX-A duration by evaluating the time required for the eyelid movement frequency to return to its baseline levels.

V. Compliance with Ethical Requirements

Federal University of S. Paulo Review Board approved this study (CAAE89528618.4.0000.5505), and all patients were treated according to the Declaration of Helsinki.

VI. Conclusions

The app presented in this work is an efficient resource for identifying and representing eyelid movement occurrences in patients. The easy access and facilitated usability platform improve the physicians' visualization and control of hemifacial spasm disease. That allows objectively assessing treatment response, as it delivers high accuracy metrics and charts based on the uploaded videos. It also contributes to accurately monitoring patients over time as it allows doctors to observe the disease patterns more easily (currently, there is no practical method for this purpose). This system could enable customized and fine adjustments to botulinum toxin doses based on each patient's needs.

VII. Conflict of Interest

The authors declare that they have no conflict of interest.

VIII. References

1. Ross AH, Elston JS, Marion MH, et al. (2011) Review and update of involuntary facial movement disorders presenting in the ophthalmological setting. Survey of Ophthalmology 56:54-67.
2. Green KE, Rastall D, Eggenberger E (2017) Treatment of Blepharospasm/Hemifacial Spasm. Current Treatment Options in Neurology 19(11):41-55.
3. Kenney C, Jankovic J. (2008) Botulinum toxin in the treatment of blepharospasm and hemifacial spasm. Journal of Neural Transmission 115(4):585-591.
4. Bilyk JR, Yen MT, Bradley EA, et al. (2018) Chemodenervation for the Treatment of Facial Dystonia: A Report by the American Academy of Ophthalmology. Ophthalmology 125(9):1459-1467.
5. Osaki MH, Osaki TH, Garcia DM, Osaki T, Gameiro GR, Belfort R Jr, Cruz AAV (2020) An objective tool to measure the effect of botulinum toxin in blepharospasm and hemifacial spasm. Eur J Neurol 27(8):1487-1492.

6. Osaki M. H., Osaki TH, Garcia DM, Osaki T, Gameiro GR, Belfort Jr R, Cruz AAV (2020) Analysis of blink activity and anomalous eyelid movements in patients with hemifacial spasm. Graefes Arch Clin Exp Ophthalmol 258(3): 669-74
7. Yoshimura DM, Aminoff MJ, Tami TA et al. (1992) Treatment of hemifacial spasm with botulinum toxin. Muscle Nerve 15(9):1045–1049.
8. Osaki T, Osaki MH, Osaki TH et al. (2016) Influence of involuntary eyelid spasms on eyelid morphometric changes in patients with hemifacial spasm. British Journal of Ophthalmology 100(7):963-970.
9. Van Rossum G, Drake FL (2009) Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.
10. Pedregosa et al. (2011) Scikit-learn: Machine Learning in Python, JMLR 12:2825-2830.
11. Sagonas C, Antonakos E, Tzimiropoulos G, Zafeiriou S, Pantic M (2016) 300 faces In-the-wild challenge: Database and results. Image and Vision Computing (IMAVIS), Special Issue on Facial Landmark Localisation "In-The-Wild".
12. Rosebrock A (2017) Facial landmarks with dlib, OpenCV, and Python. [S.l.]. Disponível em: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/.
13. Soukupová T; Cech J (2016) Real-Time Eye Blink Detection using Facial Landmarks. Center for Machine Perception, Department of Cybernetics Faculty of Electrical Engineering, Czech Technical University in Prague, [s.l.], Disponível em: http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf.
14. Dewi C, Chen R-C, Jiang X, Yu H (2022) Adjusting eye aspect ratio for strong eye blink detection based on facial landmarks. PeerJ Comput. Sci. 8:e943 http://doi.org/10.7717/peerj-cs.943

Enter the information of the corresponding author:

Author: Regina Célia Coelho
Institute: Technology and Science Institute, Federal University of São Paulo
Street: Talim, 330
City: São José dos Campos
Country: Brazil
Email: rccoelho@unifesp.br