Face Recognition via webcam or files videos

Martinez Luna Adad Marcel
Departamento de Ciencias e Ingeniería de Computación
Instituto Politécnico Nacional - ESCOM
Ciudad de México, México
adadmarcelmartinez@gmail.com

Rodolfo Romero Herrera
Departamento de Ciencias e Ingeniería de Computación
Instituto Politecnico Nacional -*ESCOM*
Ciudad de México, México
rromeroh@ipn.mx

Gabriela De Jesús López Ruiz
Sección de Estudios de Posgrado e Investigación
Instituto Politécnico Nacional - ESCOM
Club de Biorobotica
gavy_dlib@hotmail.com

***Abstract*— This article will talk about the Viola-Jones algorithm for face detection, and the Histogram Oriented Gradients or HOG algorithm and the ways in which it can be used for the detection of faces, as well as the data obtained through the use of software created with the purpose of determining the effectiveness of the Viola-Jones algorithm, and analyzing its implementation in real-time, surveillance systems as well as proposing ways to improve Face detection. The main factor that influences face detection is illumination. Faces are detected at a distance of 8 meters.**

Keywords— Facial recognition, Viola-Jones, HOG.

I. INTRODUCTION

Facial recognition through the use of automated systems or computers has been a topic of interest due to the number of applications in which this can be implemented, some of the uses of facial recognition are:

- Emotion Analysis
- Storage of biometric data
- Support systems for authorities in investigations
- Access control in high-security facilities
- Among others.

Over time, multiple facial recognition algorithms have been created, each characterized by the way it detects faces, in the case of this article we will mainly talk about 2 algorithms the HOG algorithm (Histogram Oriented Gradients) and the Viola-Jones Algorithm.

The Viola-Jones Algorithm works by image pixelation and looking for patrons specific of pixels to determine if what is being compared is a face or not [1].

Today there are already multiple libraries in multiple programming languages that include a version of this algorithm, the most popular is OpenCV, although there are alternatives such as Skimage in python, Keras, etc.

The Viola-Jones algorithm uses what is known as "Haar cascades" or cascade classifiers that contain the patterns to identify each object, so to speak a human face, eyes, mouth, inanimate objects, etc. [2].

The algorithm of Histogram Oriented Gradients or HOG allows by dividing images into small spatial regions that accumulate a one-dimensional histogram of directions, which are characterized by the "luminosity" of each of its edges [3].

To be more precise the algorithm of HOG calculate by means of the luminosity of an image section, the orientation of these, always orienting them towards the most luminous pixel of that section.

Having done that through the use of a descriptor, the pixel orientations are obtained and unified the feature that can be used to detect biometric elements [4].

In what corresponds to programming languages, We will talk about python, as it will be the basis of the facial recognition system of this writing. Python is characterized by being high level, strongly typed, imperative and object-oriented [5,6].

Python is currently implemented in multiple areas of computer science [7], such as:

1. Computer security.
2. Artificial Intelligence
3. Software engineering.
4. Scientific computing.
5. Data science.
6. Web development.
7. Etc.

The objective of this article is to show the results obtained in the development of software that by using the algorithms mentioned above detects the faces of some video source and to save facial features.

II. MATERIAL AND METHODS

The materials used in the elaboration of this software were:

1. 32-bit Windows 7 desktop computer with 2 GB of RAM
2. Integrated Development Environment or IDE by its abbreviations in free code, called Spyder, which comes included when installing a distribution of programming in python called Anaconda [8].
3. A high-speed HDMI cable.
4. 8 GB USB
5. Webcam for computer model Microsoft LifeCam VX-800[28].

Identify applicable funding agency here. If none, delete this text box.

The main characteristics are the ability to analyze live video or through a video file and store the faces detected in the selected source, as well as storing an image with the HOG algorithm to analyze facial features or expressions

The process of creating this software began with the creation of a graphical interface that allowed the selection of a video source, so multiple sources were used [9, 10, 11, 12, 13], through which also the possibility of obtaining video material that was on a digital platform [14, 15, 16], a feature that was not added due to the legal problems that the use of online materials could bring without the authorization of the creator or creators of this.

After having managed to create an appropriate graphic interface, we began to analyze how to obtain the appropriate video frames, for later analysis.

After an investigation of how to analyze videos in python [17, 18], it was concluded that the OpenCV library, combined with basic elements of python, was the most suitable for solving the problem.

After solving the problem of saving the video frames, the problem arose of how to analyze the existence of faces in the saved frames, which led to another investigation [19, 20, 21, 22] that allowed finding the existence of a detector of faces in the Python OpenCV library.

Then the implementation of the HOG algorithm began, so an investigation had to be carried out [23-26], where was determined that the use of the image library was the most convenient way to obtain the images of the HOG algorithm, with the purpose of being implemented in the future the detection of some system that depends on the HOG algorithm. The figure 1 showed the diagram of software.

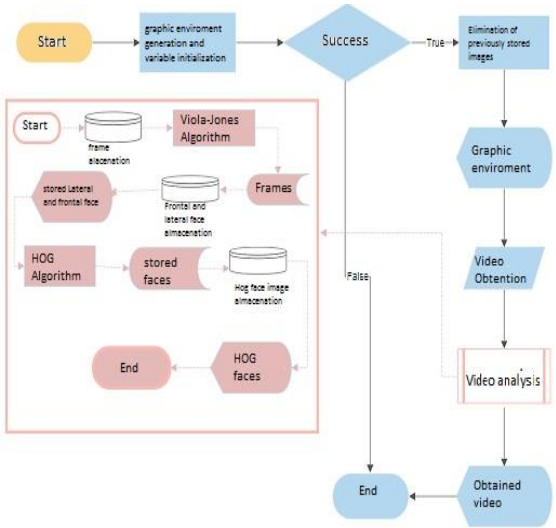

Fig. 1. Software flowchart

III. RESULTS

After doing 14 tests, it was determined that the effectiveness of the software is of 98.82% since a total of 43410, where frames were analyzed with a total of 36738 detections, of which 36305 were expensive and 433 false positives, see Table 1 and 2:

Table 1. Data obtained by analyzing data from the frontal detection of the software in multiple tests for a better appreciation from the data go to section V.

TABLE I. DATA OBTAINED BY ANALYZING DATA FROM THE FRONTAL DETECTION FACE

Frontal detection data						
Test Number	Frames	Detections	Successes	Failures	Successes (%)	Failures (%)
1	922	255	238	17	93.3%	6.63%
2	1181	448	430	18	95.98%	4.02%
3	971	948	940	8	99.15%	0.85%
4	171	27	26	1	96.29%	3.71%
5	30,473	15,973	15,955	18	99.88%	0.12%
6	1,167	442	429	13	97.05%	3.95%
7	1,020	87	35	52	40.22%	59.88%
8	900	3	3	0	100%	0%
9	900	16	14	2	87.5%	12.5%
10	900	19	16	3	84.21%	15.79%
11	1429	584	525	59	89.89%	11.11%
12	1500	1	1	0	100%	0%
13	550	62	58	4	96.6%	3.4%
14	1326	1448	1423	23	98.41%	1.59%
Totals	43410	20313	20095	218	98.92%	1.08%

TABLE II. DATA OBTAINED BY ANALYZING DATA FROM THE LATERAL DETECTION

Lateral detection data						
Test Number	Frames	Detections	Successes	Failures	Successes (%)	Failures (%)
1	922	149	142	7	95.3%	4.7%
2	1181	221	217	4	98.20%	1.80%
3	971	602	532	70	88.37%	11.63%
4	171	25	23	2	92%	8%
5	30,473	14,416	14,350	66	99.54%	0.46%
6	1,167	114	106	8	92.98%	7.02%
7	1,020	43	17	26	21.22%	78.78%
8	900	9	8	1	88.8%	11.22%
9	900	50	49	1	98%	2%
10	900	27	25	2	92.59%	7.41%
11	1429	228	210	18	92.10%	7.9%
12	1500	5	5	0	100%	0%
13	550	86	78	8	90.69%	9.31%
14	1326	450	448	2	99.5%	0.5%
totals	43410	16425	16210	215	98.75%	1.25%

In the case of test No. 7, the large number of errors is due to the fact that in the set of analyzed frames there were multiple similarities between faces and elements with fake faces, for example, see Figure 2:

Fig. 2. Set of images showing detection errors with similarities to human faces in the first three images and detection errors as in the last two images.

By creating the above tables, the creation of the graphs of figure 3, showing the software detection successes as well as

their false positives or failures was allowed when the face is in Frontal posición:

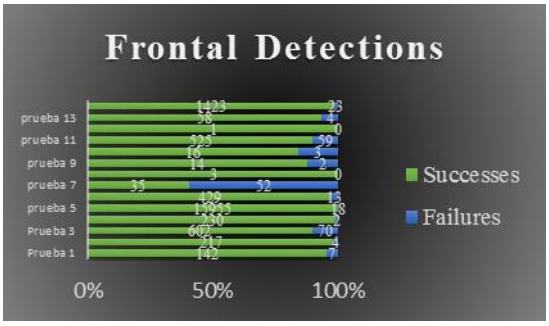

Fig. 3. Graph for face detection in Frontal position.

In Figure 4, it can be seen that in test 7 only 17% success was obtained because the lighting conditions were not favorable.

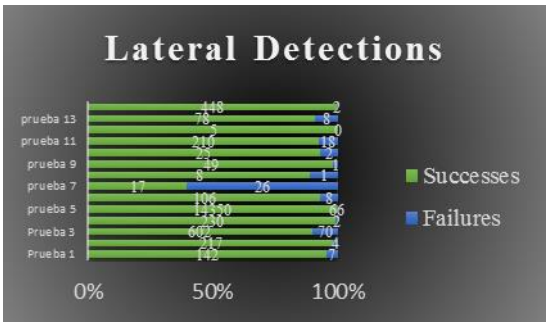

Fig. 4. Graph for face detection in lateral position

Comparing both frontal and lateral positions, it is concluded that the effectiveness exceeds 90%, where the tests were carried out from 50 cm to 8 m distance. Ver figure 5.

Fig. 5. Total results of lateral and frontal position.

By analyzing the data obtained by the tables, it can be determined that is quite efficient since by presenting a percentage of 98.69 % detection accuracy of lateral faces with 16425 detections and a percentage of 98.92 in frontal detections with 20313 detections, both percentages obtained from 43410 frames, indicates that it has good reliability.

Another issue to consider is the execution time because when analyzing 30,473 frames obtained from a video stored in memory it takes a total of 537 minutes or 8 hours with 57 minutes.

IV. Conclusions

After analyzing the statistics obtained by the multiple experiments carried out with the software, we can conclude that the effectiveness of the software depends on the ambient lighting as well as the number of elements that are in the environment where the tests are being developed.

As an example of this, we must consider all the cases that occurred during the tests, there were cases where the accuracy is very close to or equal to 100%, while in other cases the accuracy is less than 90% and even reached 21%.

While in tests conducted with a webcam, it was possible to determine that the approximate detection distance for side face detection is approximately 8 meters and for frontal detections is approximately 6 meters while the detection range of the camera is 355 ° at 14 °.

This leads us to the conclusion that the use of the Viola-Jones algorithm for face detection even with slightly old technologies allows us to obtain acceptable results as long as it is under the right lighting conditions.

One way to improve detections with the Viola-Jones algorithm is through the use of some artificial intelligence, neural network, creating more sophisticated classifiers or some type of software capable of better detecting the characteristics of a face, such as eyebrows, lips, nose, etc.

Similarly, making the detection process in pre-recorded videos more efficient could be beneficial, since the execution of the program is somewhat delayed by the way in which it processes each frame stored in memory, slowing the detection of faces in videos prerecorded.

Acknowledgment

The authors thank the IPN (Instituto Politécnico Nacional) for the support received.

References

[1] M. Guevara, J. Echeverry and W. Urueña, Detección de rostros en imágenes digitales usando clasificadores en cascada. Colombia, Pereira: Universidad Tecnológica de Pereira, 2008, pp. 2-3.

[2] E. Parra Barrero, Aceleración del algoritmo de Viola-Jones mediante rejillas de procesamiento masivamente paralelo en el plano focal. Sevilla, España: Universidad de Sevilla, 2015, pp. 26-27.

[3] N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection. Montbonnot, Francia: INRIA Rhone-Alps, pp. 1-3.

[4] O. Déniz, G. Bueno, J. Salido and F. De la Torre, Face recognition using Histograms of Oriented Gradients. ELSEVIER, 2011, pp. 2-3.

[5] D. Rodo, El lenguaje Python. Catalunya, España: Universitat Oberta de Catalunya, 2019, p. 5.

[6] S. Paque Martin and D. Albolafia Cañete, El lenguaje Python. Málaga, España: Universidad de Málaga, 2009, p. 6.

[7] "¿Qué puedo hacer/construir con Python?", *Es.quora.com*, 2017. [Online]. Available: https://es.quora.com/Qu%C3%A9-puedo-hacer-construir-con-Python. [Accessed: 01- Oct- 2019].

[8] "Anaconda Python/R Distribution - Free Download", Anaconda. [Online]. Available: https://www.anaconda.com/distribution/. [Accessed: 27- Sep- 2019].

[9] "Video Player with Qt5",Welcome to python-forum.io, 2016. [Online]. Available: https://python-forum.io/Thread-Video-Player-with-Qt5. [Accessed: 27- Sep- 2019].

[10] M. Carmelo, "TutorProgramacion/pyqt-tutorial", GitHub, 2017. [Online]. Available: https://github.com/TutorProgramacion/pyqt-tutorial/blob/master/13-multimedia/multimedia.py. [Accessed: 27- Sep- 2019].

[11] "Código de Python - Reproductor de vídeo", Lawebdelprogramador.com, 2018. [Online]. Available: https://www.lawebdelprogramador.com/codigo/Python/4481-Reproductor-de-video.html. [Accessed: 27- Sep- 2019].

[12] M. Hetland, Beginning Python, 2nd ed. Berkeley, CA: Apress, 2008, pp. 277-288.

[13] F. Romano, Learning Python. Birmingham: Packt Publishing, 2015, pp. 258-271.

[14] L. Salcedo, "Descargar vídeos de YoutTube con Python y Pafy", Pythondiario.com, 2017. [Online]. Available: http://www.pythondiario.com/2017/10/descargando-videos-de-youttube-con.html. [Accessed: 27- Sep- 2019].

[15] "How do I download videos from any website using Python?" Quora, 2018. [Online]. Available: https://www.quora.com/How-do-I-download-videos-from-any-website-using-Python. [Accessed: 27- Sep- 2019].

[16]]"Código de Python - Descargar un Archivo de internet y guardarla en nuestro disco con urllib2", La web del programador.com, 2012. [Online]. Available: https://www.lawebdelprogramador.com/codigo/Python/2250-Descargar-un-archivo-de-internet-y-guardarla-en-nuestro-disco-con-urllib2.html. [Accessed: 27- Sep- 2019].

[17] "Empezando con vídeos - Cursos de Programación de 0 a Experto © Garantizados", unipyhton Cursos de Programación de 0 a Experto © Garantizados, 2018. [Online]. Available: https://unipython.com/empezando-con-videos/. [Accessed: 27- Sep- 2019].

[18] "Warning: the frame size for reading (864, 1080) is different from the source frame size (1792, 1080) · Issue #415 · imageio/imageio", GitHub, 2018.

[19] J. Brownlee, "How to Perform Face Detection with Deep Learning in Keras", Machine Learning Mastery, 2019.

[20] S. Domínguez Pavón, Reconocimiento facial mediante el Análisis de Componentes Principales (PCA). Sevilla, España: Escuela Técnica Superior de Ingeniería Universidad de Sevilla, 2017, pp. 9-11.

[21] "Face detection using a cascade classifier — skimage v0.16.dev0 docs", Scikit-image.org, 2019.

[22] L. Moreno Villalba, Reconocimiento Facial con OpenCV y Python. Ciudad de México, México, 2018, pp. 1-11.

[23] S. Mallick, "Image Recognition and Object Detection : Part 1 | Learn OpenCV", Learnopencv.com, 2016. [Online]. Available: https://www.learnopencv.com/image-recognition-and-object-detection-part1/. [Accessed: 01- Oct- 2019].

[24] "Obtener características de imagen HOG de OpenCV + Python?", Es.switch-case.com, 2011. [Online]. Available: https://es.switch-case.com/52605903. [Accessed: 01- Oct- 2019].

[25] "selected block normalization method is invalid ·Issue #3 · BUPTLdy/human-detector",*GitHub*, 2017

[26] "Histogram of Oriented Gradients — skimage v0.16.dev0 docs", Scikit-image.org. [Online]. Available: https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html#sphx-glr-auto-examples-features-detection-plot-hog-py. [Accessed: 01- Oct- 2019].