International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Index Copernicus Value (2016): 79.57 | Impact Factor (2017): 7.296

Study of the Precision and Feasibility of Facial Recognition using OpenCV with Java for a System of Assistance Control

Rene Cuamatzi Briones[1], Juan Ramos Ramos[3], Rodrigo Tlapa González[3] José Juan Hernández Mora[4]

[1, 2, 3, 4]Tecnológico Nacional de México, Instituto Tecnológico de Apizaco, Fco. I Madero s/n, Barrio de San José, 90300, Tlaxcala México

Abstract: ***Computer vision is a computational discipline which attempts to emulate human vision and has been done presently in current systems and applications. Facial recognition is a biometric method that allows identifying different people by means of unique features that the human eye is simply unable to perceive. Currently, most systems and applications require greater security, the reliability of use and protection of data, which is why we opt for the use of technologies based on facial recognition as the case of Google and Facebook, to name a few. This face detection technology is mainly based on visible images and despite being a discipline with more than 30 years of study, still have certain problems at the implementation time, such as the light under which the images are captured, the posture, face expressions and the quality with which the device captures the image, among others. However, there is a wide variety of it tools that allow making improvements in the image and the detection of faces in an easier way; among the most used are MatLab and Phyton that employ the use of open source libraries as OpenCV in the case of this study, has been opted for the use of Java in conjunction with OpenCV, Because unlike MatLab, it does not represent economic cost and facilitates the integration of computer vision algorithms with an assistance control system.***

Keywords: Face Recognition, OpenCV, Java, Systems.

1. Introduction

The integration of computer systems with computer vision Systems has been making present in recent years, applications such as [3], [4], [6] help the automation and streamlining of different processes. OpenCV (Open Source Computer Vision Library) is a library that allows performing this task, in addition to helping to alleviate the burden of software development in the computer vision area. It offers a wealth of image processing functions and algorithms that focus on real-time image analysis.

These functions are designed to accelerate the process of developing software for object identification, the segmentation, Image recognition, facial recognition, gesture recognition, among other activities [4].

However, it is not currently common to find computer vision systems developed in the Java programming language applied to real-world problem-solving. In comparison, it is more recurrent to locate these applications developed under the MatLab platform [3], or, the use of additional hardware that allows the acceleration of the computing process [8], which requires either obtaining a MATLAB license or acquiring additional hardware, which also represents an additional expense to the development of the system. This gives rise to a wide field of opportunity in the application of these technologies for the development of new solution proposals in applications with artificial vision using Java that unlike the above options offers a free license for the development of desktop systems that allow the solution of real-world problems.

Therefore, it is the subject of this study demonstrate the feasibility of integrating facial recognition algorithms Offered by OpenCV with an assistance control system developed in Java, as shown in Figure 1.

The integration of the two systems is proposed. Initially there is a system of assistance control of the personnel of government management and on the other hand, a facial recognition system is integrated by using the OpenCV library and a webcam, resulting in the Assistance Control System through Facial recognition (SCARF).

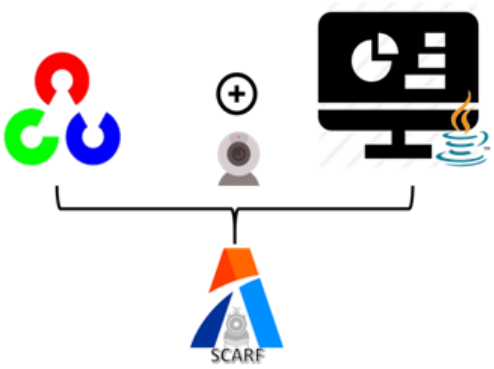

Figure 1: Systems integration methodology OpenCV / Java

2. Facial Recognition in OpenCV

It is important to keep in mind that facial or faces detection is not the same as facial recognition, since detection only focuses on locating a face object within an image, while recognition makes use of the results of the detection and is responsible for extracting the features that best describe the image to predict future appearances of the face in different images.

Through this research is aimed to implement a system that is capable of detecting, processing and analyzing faces and its integration with an application of assistance control developed in Java, specifically for desktop applications.

OpenCV allows the development on desktop applications from the 2.4.4 version so you have to be careful at the time of your selection and installation of this library. This version currently has three algorithms for face detection, known as Eigenfaces, Fisherfaces y LBPH (Local Binary patterns Histograms).

2.1 EigenFaces

Eigenfaces is a facial recognition algorithm based on the principal component analysis (PCA), developed by the Massachusetts Institute of Technology (MIT). This algorithm is used for the features extraction of the face images and is capable of producing an accuracy of 73% from 3 different angles of the face [5].

2.2 Fisherfaces

Unlike eigenfaces, this algorithm takes into account how light is reflected in the face and facial expressions. The accuracy and performance of this algorithm depend largely on input data, so, if it is trained only by images with good lighting and tries to recognize faces in poorly lit scenes, there is a great possibility that the method finds the wrong components.

2.3 LBPH (Local Binary Patterns Histograms)

Binary local patterns (LBP) are a simple but highly efficient texture operator that labels the pixels of an image by the neighborhood threshold of each pixel and considers the result to be a binary number. LBP offers very good results in terms of the speed and performance of discrimination, as well as in different lighting conditions [6]. Thus, the main idea of the LBP is not to look at the whole image as a high-dimensional vector, but to describe only the local characteristics of an object and summarize the local structure in an image. The result of this process can now be divided into N number of local regions of the type (grid x * Grid y), allowing the possibility of generating a histogram of each one of them. These histograms are known as LBPH.

3. Implementation

3.1 General Description

The diagram in Figure 2 shows the process performed by the Facial recognition system and the connection that allows communication with the assistance control system.

Initially, there is the facial recognition system which consists of three different modules. The first one allows the collection of samples (pre-processed photographs); these images correspond only to the face (region of interest) of the personnel from which the management of their assistance is required. It is important to mention that each and every image is labeled and stored in a folder with the name "faces" within the local System folder with a unique ID, that in the case of this study corresponds to the control number of each employee who in turn, that label, is stored in a database related to the information of the workers. Once all the desired samples have been collected (n number of employees), the training module can be used.

The training module performs the processing of the image and its features extraction, from the Faces folder that is created in the Sample collection module. Working with the OpenCV library requires the use of the format that the library itself requires. So the objective of this module is to get a file with the extension. YML, which is an efficient data serialization document and can be read by any plain text editor developed to create, open, and edit plain text files. This file has the name of FEATURES.YML and contains the socialization of the features vectors of each image in relation to the ID that is stored, both in the database and on the label of each of the images stored in the "Faces" folder.

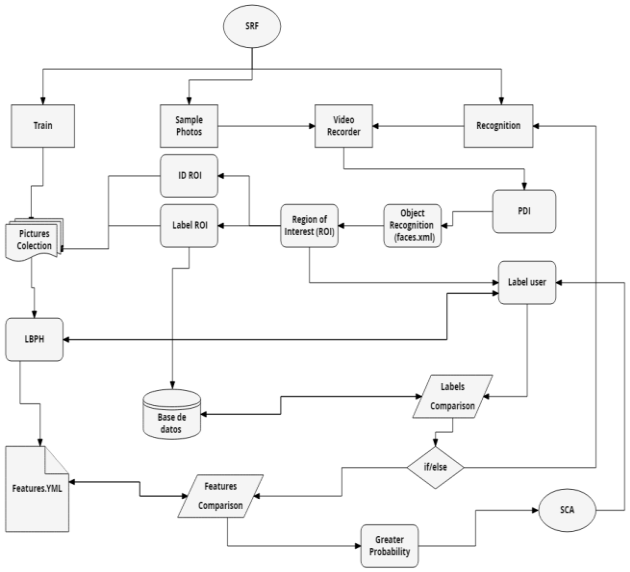

Figure 2: General diagram of the facial recognition system.

The third module belongs to the recognition and validation of the face. That is, on the one hand, the procedure is made to recognize a face in the video captured by the webcam, and on the other is verified that the features of this coincide with those of some of the vectors stored in the file FEATURES.YML, However, it is very likely that the facial features, face position, and stage lighting obtained during the training process are completely different from those of recognition, so it is sought to find the most approximate vector and not an exact one initially captured.

3.2 Detection Process OpenCV /Java

OpenCV initializes the webcam in video recorder mode, although it is not the objective to perform the recognition of faces on video, it is required that the camera is active for a long period since its application will be for a set of people who make its registration In and out consecutively in your working stay and there can be no place to stop the execution of the camera. An event will then capture the images in real time. In the case of this study, 20 captures of the face identified in the video are obtained from four different profiles per person. It is important to remark that the captured images only take the region of interest (ROI), in this case, it is of interest only the face, so it is used the use of an algorithm for facial detection.

OpenCV contains pre-trained classifiers for face, eyes, smiles, among others. These classifiers are hosted in XML files within the OpenCV installation folder. To work on them in the Java environment, first, it is necessary to load the required XML classifiers since in this study it is intended the facial detection so it makes use of the file "Haarcascade_frontalface. XML ".

3.3 Image Capture

Initially, the webcam has an HD resolution of 1280 X 720 pixels, which is equivalent to a matrix of 921.600 pixels for each of the photos. If you consider the 20 images required for each person's profile training, you would have to analyze 18,432,000 pixels for each 1 of 4 face profiles per person, creating an excessive and unnecessary load of processing for the computer.

As a solution, OpenCV has as a basis the use of the algorithm of Paul Viola and Michael J. Jones, where they introduce the concept of the integral image, described in [7]. Thus, no matter how large the image to be processed, the calculations are reduced for each of them, giving rise to an extremely fast facial detector that allows the promise to achieve practical and reliable applications. Once the ROI is detected, it must be tagged and stored. However, in order to have better computational performance, the images are processed before their storage. It begins by resizing them to a smaller scale (125 x 150 pixels) and they pass from the space of RGB color to grayscale, in addition to being improved by the equalization of their histogram.

In the same way, each image requires a label or ID that allows its later identification since the objective of the system is the management of the personnel through his face. Thus, it takes advantage of the unique control number with which they count in the registry of the assistance control system to be used like that label of each image and stored in the folder "faces" described above.

3.4 Features

Subsequently, the LBPH is used to form features vectors that best describe each image. This is done using the images contained in the "Faces" folder. For each of them, their local binary patterns are obtained and placed in a new matrix (see fig. 3.1). To perform this procedure is used the "Circular LBP", which allows modifying two key parameters in the generation of LBP´s. Figure 3.1 shows a graphical representation of these factors. The first is the neighbors that correspond to the set of pixels that are taken into account as part of a neighborhood, these can be 4 or 8. The second is a variable radius that corresponds to the distance of the central pixel of a matrix with respect to the coordinates of the pixels that formed part of the vicinity. The variation of the selection of this neighborhood allows capturing finer details in the images. However, it should be taken into account that a larger neighborhood or radius requires a greater number of calculations, and thus has a higher demand for processing. Because the algorithms must offer a real-time response, this must be relatively fast. That is why in the case of this work has chosen to use a neighborhood of 8 pixels and a radius equal to 1, avoiding an additional calculation to calculate the coordinates of the pixels corresponding to a radius greater than or equal to 2.

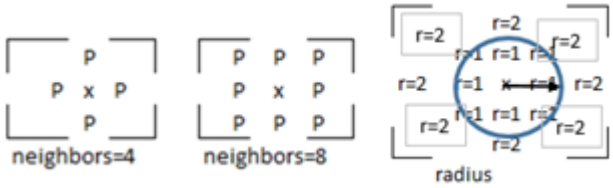

Figure 3: Circular LBP

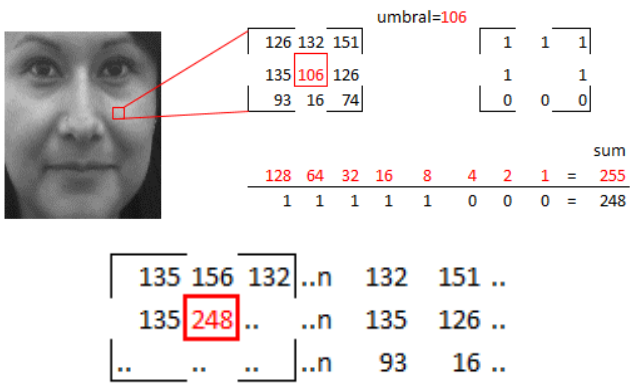

Figure 3.1: LBP process

Once obtained all the LBP of the resulting image is divided into 64 grids, 8 corresponding to the X-axis and 8 to the Y. As with the neighborhood and the radius, a too high value of grids provides better descriptors but raises processing time. Finally, the resulting regions are represented by histograms and those same are concatenated to obtain a single vector. Assuming that each histogram represents values from 0 to 255 (possible values of a grayscale image) the final vector will contain 16.384 values. That is to say, 64 histograms concatenated with 256 values in each one of them.

3.4.1 Storage of Features

At the end of the construction of the vector of characteristics, this must be stored in a specific format. OpenCV uses the management of YML files so the vectors of each image will be saved and serialized with their corresponding identification number. i.e., for each processed image a new vector is added to the file features.YML and parallel a vector with the identifier of each image, in order to maintain the relationship of both parties and quick access to any vector regardless of their position within the file. Another important point is the time required for the execution of this task since this is relative to the number of people who want to register and the number of samples obtained in the first phase. because the number of workers is considerably high, this function has been isolated from the main tasks of the system, such as the execution of the webcam, and it is recommended that their execution be at the end of the entry or exit log of the workers (period in which not required to use the camera).

3.5 Facial Detection

The third and last module integrated to the system corresponds to the detection itself, and it has as purpose to be incorporated into the system to carry control of the entries and exits of the registered personnel. This algorithm is very similar to the one described above with the exception that the

vector obtained during this stage is not stored, but it is compared with those existing in the file "features.YML" calculating their Euclidean distance. The vector with which the shortest distance is obtained is considered to be the best candidate or the most related. As mentioned in 3.4.1, each vector is related to a unique ID that labels each image and is this identifier that interests, since that value is stored and related to the records of each employee in the system database.

4. Results

Tests have been developed for the same group of personnel. However, there is a variation of 50% of the number of samples per person and are taken from 4 different profiles. This in order to predict the time required for the operation of the system with a certain number of records.

Table 1: Performed times

Images collection	Profiles samples	Persons	Train process time	Load data time	Recognition time
2400	4 X 20	30	47.36 secs	22.98 secs	[0.97 - 1.69] secs
1200	4 X 10	30	28.38 secs	11.09 secs	[.72 – 1.21] secs

It can be observed that the training time for the 2400 samples is less than twice the time occupied to train 1200. On the other hand, before making use of the system, this like another system, you must load the necessary data during the booting of the same one. Just at that time where it has been determined to release the execution of the method that is responsible for loading the data in the YML. Thus, as shown in the same table 1, there is a delay during the system boot time with respect to the size of the image collection, however, it is compensated for the time required for the recognition process, oscillating between 1 and 2 seconds.
In conclusion, the results show that it is reliable for the development of computer systems with artificial vision using Java in conjunction with the OpenCV library. Because you can get good results with respect to the runtime.

5. Future Works

This development is the basis for the construction of subsequent Java applications in integration with computer vision, providing solutions to real-world problems and meeting the needs of the governmental and business spheres. Cloud connectivity and system migration to a mobile platform is undoubtedly the primary pointers to future deployments. The use of the language in which it was developed (Java), and the achievement in the adaptation of mobile operating systems, promise an evolution pertinent to the present research work.

References

[1] H. Lee, F. Peng, X. Lee, H. Dai, and Y. Zhu, "Research on face detection under different lighting," 2018 IEEE International Conference on Applied System Invention (ICASI), Chiba, 2018, pp. 1145-1148.
[2] Y. Derhalli, M. Nufal, and T. AlSharabati, "Face detection using boosting and histogram normalization," 2015 9th Jordanian International Electrical and Electronics Engineering Conference (JIEEEC), Amman, 2015, pp. 1-6.
[3] Khushi, "Smart Control of Traffic Light System using Image Processing," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), Mysore, India, 2017, pp. 99-103.
[4] Z. Chaczko, L. A. Yeoh and V. Mahadevan, "A Preliminary Investigation on Computer Vision for Telemedicine Systems Using OpenCV," 2010 Second International Conference on Machine Learning and Computing, Bangalore, 2010, pp. 42-46.
[5] E. B. Putranto, P. A. Situmorang, and A. S. Girsang, "Face recognition using eigenface with naive Bayes," 2016 11th International Conference on Knowledge, Information and Creativity Support Systems (KICSS), Yogyakarta, 2016, pp. 1-4.
[6] R. Samet and M. Tanriverdi, "Face Recognition-Based Mobile Automatic Classroom Attendance Management System," 2017 International Conference on Cyberworlds (CW), Chester, 2017, pp. 253-256.
[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I.
[8] L. Schaffer, Z. Kincses and S. Pletl, "FPGA-based low-cost real-time face recognition," 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, 2017, pp. 000035-000038.

Author Profile

Rene Cuamatzi Briones has a degree in Computer Engineering from the Universidad Autónoma de Tlaxcala, from 2017. He is currently studying the masters in computer systems in software engineering from the Instituto Tecnológico de Apizaco.

Juan Ramos Ramos has a degree in Computer Science from the Instituto Tecnológico de Apizaco, from 1993. He is also a Master in Computer Science and Telecommunications from the Instituto de Estudios Universitarios, A.C.; he works as a full-time professor at the Instituto Tecnológico de Apizaco in the area of Systems and Computing, teaching at the undergraduate and postgraduate level, in the areas of programming and software engineering.

Rodrigo Tlapa Gonzalez has a degree in Computer Engineering from the Universidad Autónoma de Tlaxcala, from 2015. He is currently studying the masters in computer systems in software engineering from the Instituto Tecnológico de Apizaco.

José Juan Hernández Mora has a degree in Computer engineer from the Universidad Autónoma de Tlaxcala, from 1994. Master in Computer science at the National Center for Research and Technological Development of the TecNM, 2003. Research professor at the Tecnológico de Apizaco del TecNM. Teacher of the Master of computer systems of the Instituto Tecnológico de Apizaco.