Implementation of an Evolutionary Facial Recognition Algorithm on JETSON TK 1

Mohamed Salah Salhi, Atef Alaaeddine Sarraj, Hamid Amiri
Ecole Nationale d'Ingénieurs de Tunis
Tunisia
msalah.salhi@gmail.com
atefsarraj@gmail.com
hamidlamiri@enit.rnu.tn

ABSTRACT: *This paper aims to contribute in visual computing over the implementation of face recognition algorithms on Jetson TK1. This powerful tool is a NVIDIA's embedded Linux development platform featuring a Tegra K1 SOC which contains a CPU, a GPU and an ISP in a single chip. The Tegra K1 'class one' is very faster than the former Tegra 2 'class two' and is the most efficient GPU in the ARM world. This particularity added to some optimizations we have made on implementation strategy and on recognition face algorithms conduct us to reach good results in real time systems RTS.*

Keywords: Face Recognition Algorithm, Jetson TK1 Tool, Tegra K1 SoC, ARM World, Implementation Over RTS

Received: 26 August 2018, Revised 9 October 2018, Accepted 17 October 2018

DOI: 10.6025/jmpt/2019/10/1/1-17

1. Introduction

The biometric definition of individual characteristics occupies actually the major interest of international community by carrying a lot of solutions for the goods security. Thus, the development of embedded computing architectures, coupled with a succession of enormous technological advances in digital photo sensors, with or without wires and their communication protocols on the internet or on mobile systems, has promoted the biometric of facial recognition versus the other individual recognition indicators as described in the comparative histogram established according to the literature.

Similarly, it is considered that each day of everyday life, the human brain captures and memorizes several faces by learning effects. Thus, crossing one day with a person, a comparison of the target face with those recorded will be determined at the brain level to make an identification decision. This system with this recognition mechanism is currently the object of multiple attempts of computational modeling in order to establish an algorithm that optimizes better the recognition rate.

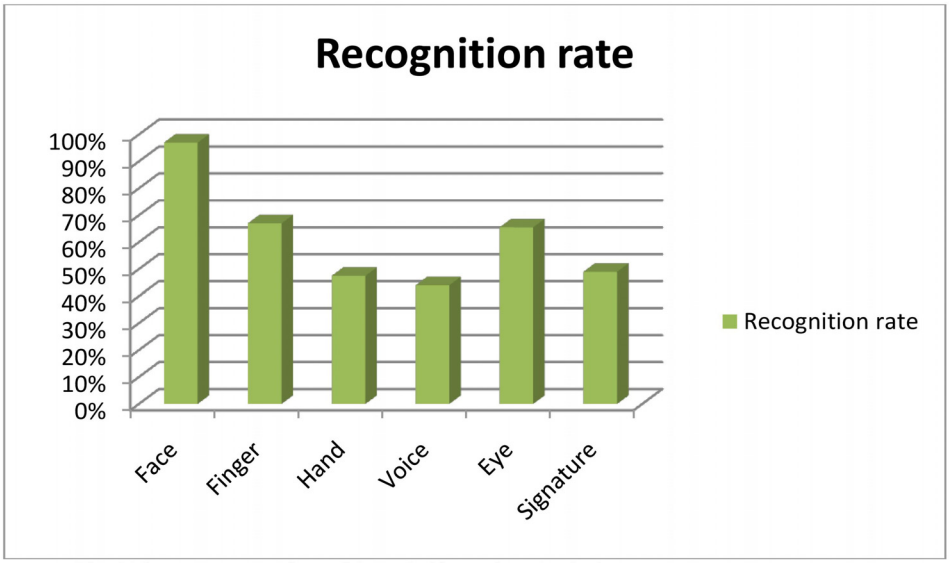

Figure 1. The scores comparison of the main biometric technologies

Until the firm NVIDIA invented in 1999 the graphics processing unit GPU, this product has made it famous and still the engine of modern visual computing. The latest generation of NVIDIA Kepler GPU and its 7 billion transistors are one of the most complex processors ever created. NVIDIA GPUs are given birth to a new area of visual computing, which now encompasses many fields of application such as video games, film production, product design, medical diagnostic procedures and scientific research. In addition, GPUs have proven their usefulness in new fields such as computer vision, computational photography, image processing and augmented reality.

Scientists and researchers worldwide are now using the GPU Tesla to speed the most intensive computational procedures in areas such as the human brain modeling or research against AIDS. NVIDIA offers them a complete ecosystem of technologies, development tools, libraries, and advice based on powerful algorithms and architectures. In addition, NVIDIA GPU accelerated the fastest supercomputers in the world, including the TITAN system of Oak Ridge National Laboratory. One of the GPU embedded solution is the JETSON TK1.

2. Facial Recognition Challenge

The facial recognition, by extraction of the distance measurement parameters between singularity points of the face, is a basic interesting static technique. However, this technique does not meet the enormous conditions of variability occurring on the individual face when he is in an emotional state; Happy or angry or when it strengthens to change these facial qualities.

Several approaches have been identified in this direction. Each supports a solution for improving the facial recognition state by adopting a mathematical and scientific model that treats the subject on one side from its own point of view. Overall, these different biometric approaches revolve around:

• The determination of the DNA Deoxyribo-Nucleic Acid.

• The skin color.

• The distance measurement.

The main disruptive elements of facial recognition are summarized in:

• The difference in the brightness of the perceived face.

• The difference in the face orientations.

• The difference in facial expressions such as opening or closing the eyes or mouth lips.

• The difference in wearing embellishment components such as long hair, whiskers and glasses.

The different adopted strategies in facial recognition are summarized in:

• An approach that focuses on the characterization of the face fixed organs.

• An approach based on the forms matching 'Pattern matching', using square and rectangular windows to check the different regions of a face.

• An approach based on PCA principal component analysis which describes the method of Karhunen loeve transformer KLT.

• An "Eigenface" approach based on the introduction of representative faces matrix of the different face versions, and then applying a neural or projection classification followed by a distance computation of and a comparison to a predetermined threshold. This approach represents a local form of the PCA model.

• A deep-learning approach.

The experimentation of all these paradigms separately, offers facial recognition rates more or less diversified and remain far from decisive in the exact determination of the concerned person.

3. Proposed Recognition Strategy

Our contribution consists in combining all the data of different characteristics extracted by the different approaches into a single vector representative of primitives. This vector will be considered as a chromosome where each data extracted from a recognition model represents a chromosome gene.

This chromosome vector is complemented by a factor gene that promotes the chromosome power. This gene is obtained by an infrared (IR) temperature matrix of the face. This is done by considering that each emitted and then reflected light is assigned to a fixed temperature data. Thus, this IR light is characterized by a wavelength λ:

$$0.7\mu m < \lambda < 1000\ \mu m$$

Similarly, each temperature of a reflected light (*IR*) has a hue (a color).

Each color of the temperature matrix is linked to a frequency (*F*), where in:

$$F = \frac{\lambda}{c} = \frac{h.\lambda}{Eg} \qquad (1)$$

With, *h*: Planck coefficient expressed in Joule/hertz (J/Hz) or in Joule Second (J.S).

Eg: Energie of Gap expressed in electron-volt (ev).

λ: Is the wavelength expressed in meters (m).

C = 3.10^8 *m/s*, It is a constant term that designates the light celerity.

The obtained chromosome which defines widely face primitives is represented as follow.

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene F
Features Data model 1	Features Data model 2	Features Data model 3	---------- ----------	---------- ----------	Matrix of color and temperature frequencies

Figure 2. Representation of the chromosome content determining the person face

The facial treatment steps, beginning with the image capture to be identified up to the reconstitution phase of the characteristic vector called chromosome followed by a comparison and a decision-making, will be detailed by the following synoptic diagram.

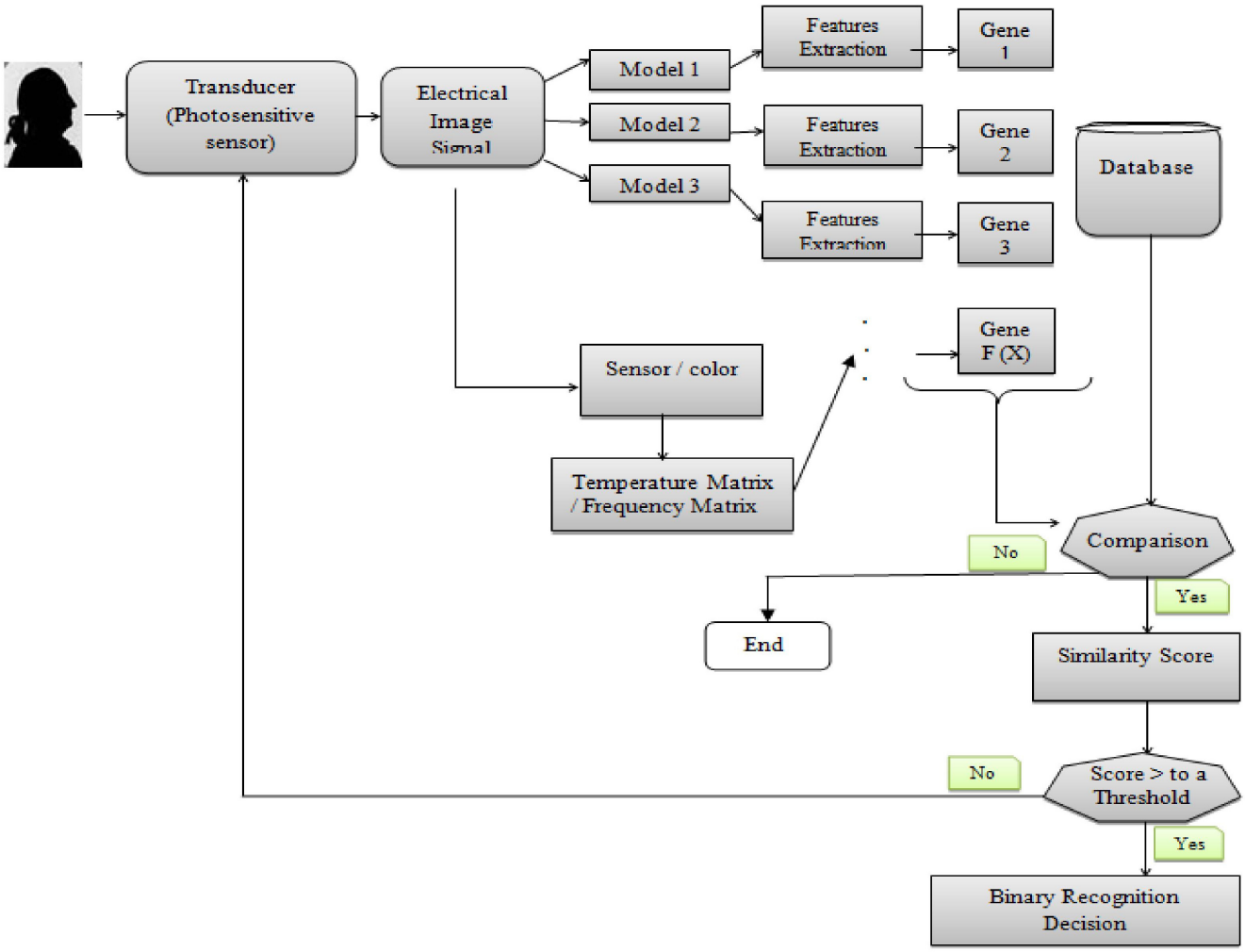

Figure 3. Synoptic diagram of the adopted recognition face paradigm

Namely, that the reflected IR rays of an object or of a person face vary in proportion to its color and therefore proportional to its temperature which translates into energy of the sensed face; it is the emissivity factor.

The chromosome vector that defines each person will be calculated as follows:

$$V_i = (gène\,(F)) * [(gène\,1) + (gène\,2) + ...+ (gène\,6)] \quad (2)$$

With, (gene 1): is the concatenation of the sub-vector of the gene (1) in a single coefficient.

The (gene F): is the matrix that should be concatenated in a characteristic vector of each individual (V_i) in the form of a line vector comparable to the different individuals vectors recorded in the database. The decision function 'fitness' is a cost function based on a score very close to one of the database vectors and very far from all the other vectors of the same base.

$$Fitness = min\,||\,V_{ci} - V_{BD}\,|| \quad (3)$$

Where, V_{BD}: a database vector.

The factor gene contributes to data removal of the characteristic vectors for individuals in order to avoid any confusion.

The machine vision algorithm relies essentially on two sub algorithms of different functionalities:

• One concerns the face detection in a captured image. This algorithm aims to establish an approximation space. It allows determining if there is a skeleton of a human face among the landscapes of an image by targeting eyes, nose and ears.

• The other sub algorithm has the function of establishing the detail space for a face skeleton detected from an image. It aims to compare this later with the recorded data which leads to an identification decision phase for the concerned person.

The linking circle between the two sub-algorithms for facial recognition is the facial normalization step as shown in the following diagram.

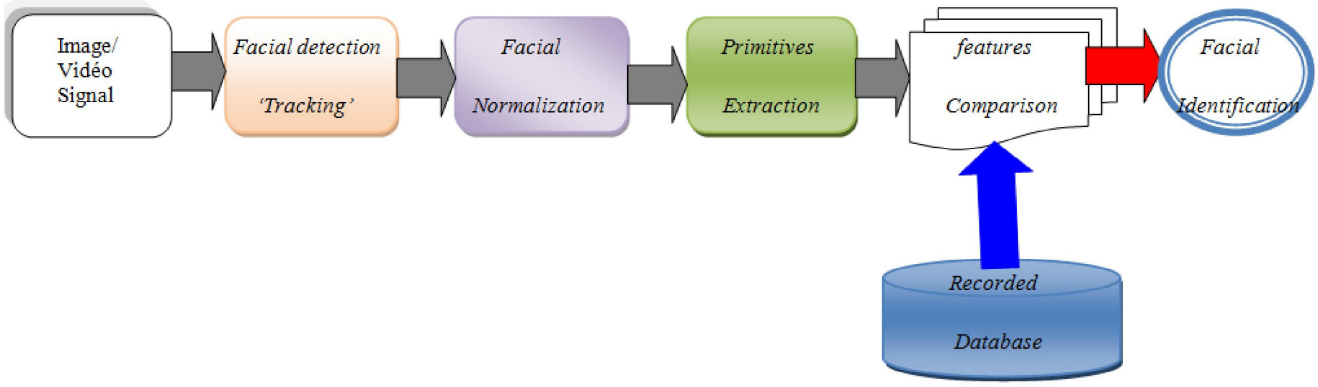

Figure 4. Facial identification diagram

4. The Images Normalization

The normalization phase consists of transforming the captured image, of different dimensions, into another image which represents a square window having a standard size generally 150 pixels on each side. During this transformation of image size reduction and consequent compression of relative characteristic data, basic criteria should to be taken into consideration such as the location of the nose, the two boundaries of the mouth lip, and both eyes. This coding operation should be followed by a linear segmentation of the obtained image in atoms of 25 pixels on each side. These atoms are often endowed with a certain stationarity with an overlap of ½ so as not to lose any characteristic information. In addition, each atom will be translated into a matrix of values and then concatenated into a data vector that can be compared with other reference vectors stored in a database. The representation of an image by a characteristic vector, called in our case chromosome vector, is compatible with the technical specificities of the memorization registers for the individuals determinant data. Thus, we notice that there are two essential elements helping to facial recognition:

• The first is based on the image simplification; this is why a black and white image is easy to be identified.

• The second relates to the verification of the appropriate contrast for an image; this operation consists in applying color filters successively in order to find the filter which is the closest to important areas of the face [1].

5. Main techniques in facial recognition

Multiple approaches have been identified in facial recognition. The main paradigms are cited:

• The 'Eigenface' method is the most used technique. It represents a local form of the principal component analysis, known as 'PCA', based on a transformation through an orthogonal eigenvectors basis.

• Independent component analysis 'ICA', known by the method of sources blind separation. In contrast to the PCA, the eigenvectors are not orthogonal.

• The 'Local Binary Pattern' LBP, is a binary classification descriptor of an image.

• The 'Gaussianface', this algorithm learning is based on the use of four databases of images instead of a single basis for the classical methods. These 4 bases contain photos with multiple poses and multiple chromatic luminance conditions. Thus, the test of this model will be done on a totally different basis from the 4 learning bases, such as the base 'Labeled Faces in the Wild' LFW which represents a validation tool widely applied in face recognition. Indeed, this technique offers results that can even exceed those of the human faculty. However, challenges lie in a high execution time, almost 4 times the allocated period by a conventional model, with the need for a much higher capacity that can support the algorithm operation.

Our contribution consists in applying to this algorithm a parallel and crossed processing system followed by an aggregation of results which adapts well for the analysis of 'Big Data' as envisaged in the following figure.

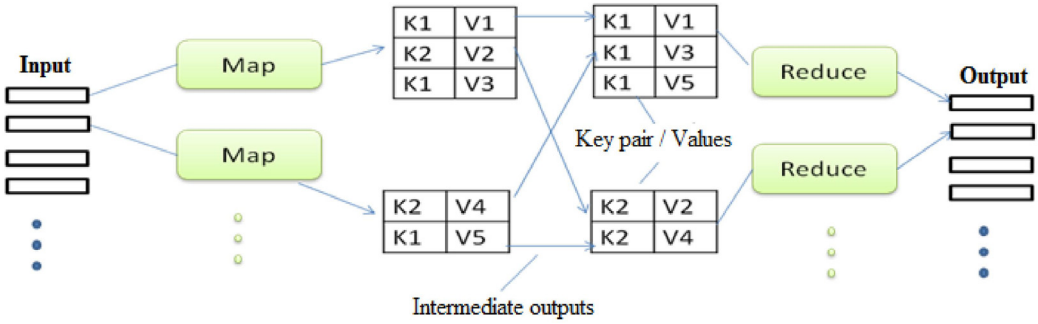

Figure 5. Synoptic diagram of the model 'Map-Reduce'

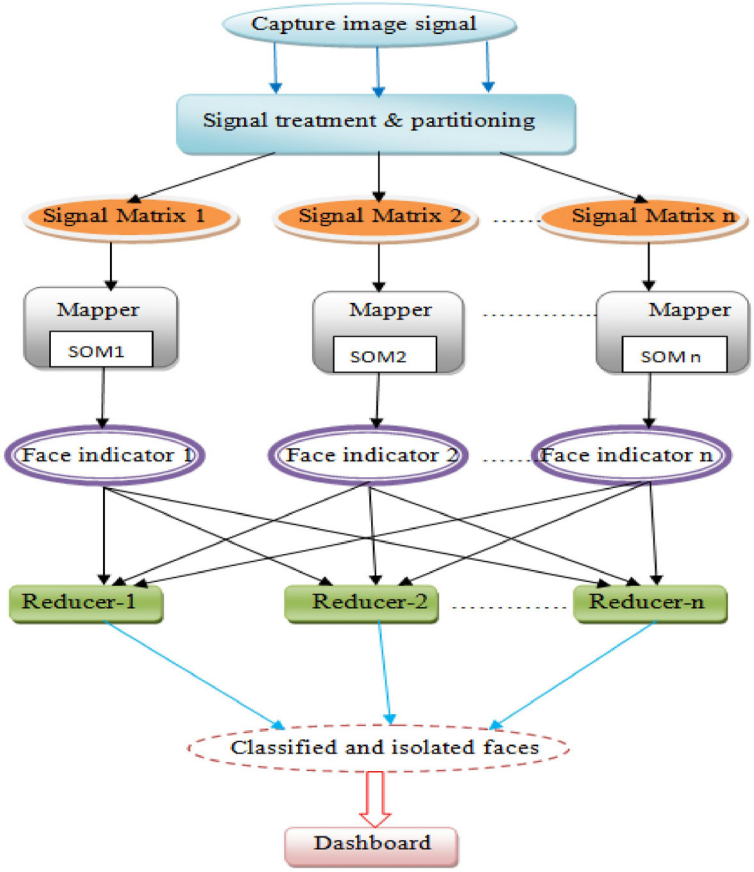

Figure 6. Description of the "SOM Map-Reduce" approach

The 'Map' module consists of a succession of self-organizing neuron maps 'SOM' for the classification of the parallel input image vector data. This will be followed by an evolutionary cross of data. The 'Reduce' modules then aggregate and store these data into output classes that can be compared to reference vectors stored in the database for facial identification making-decision. The detailed paradigm is illustrated in figure 6.

The functioning of this evolutionary parallel approach is detailed by the following representation.

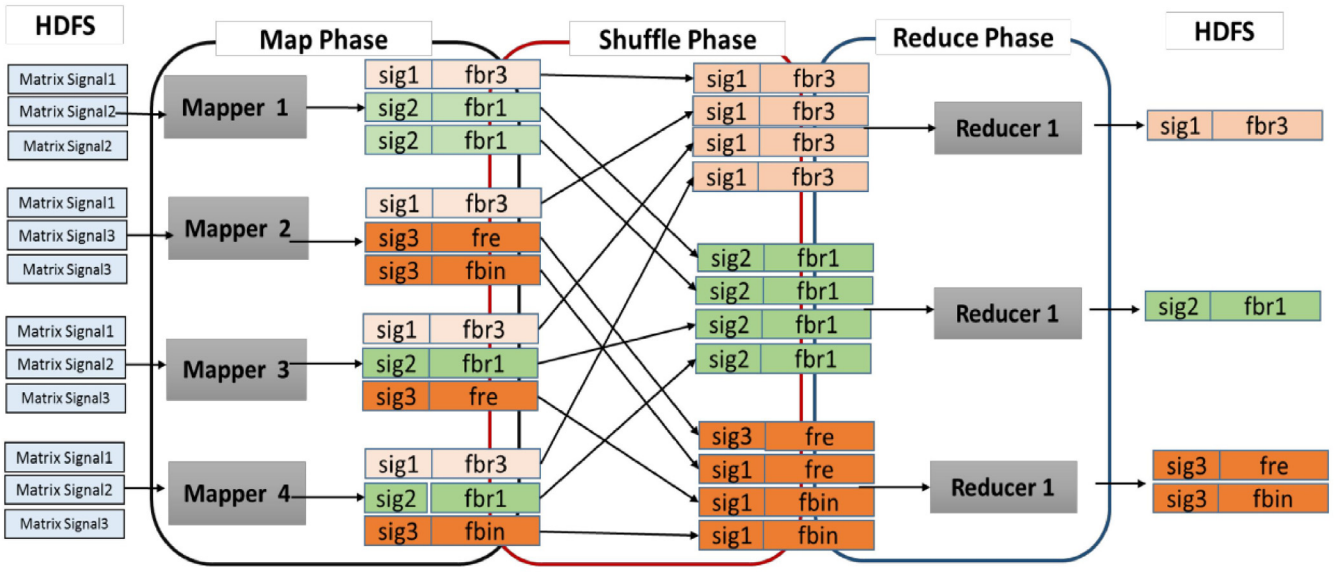

Figure 7. Execution modality of the evolutionary parallel approach

Fbr (n): is the index of an identified face.

It should be noted that the performance of the obtained results identification depends strongly on the adopted validation basis as shown in the following table and histogram:

Base type	PCA algorithm	LBP algorithm
Faces base ORL	91.82 %	97. 50 %
Faces base FEI	81.47 %	95.62 %

Table 1. Comparison of facial recognition techniques according to the adopted databases

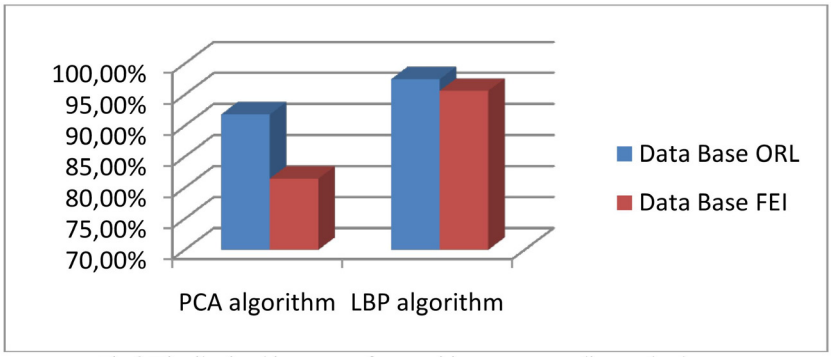

Figure 8. Distribution histogram of recognition rates according to databases

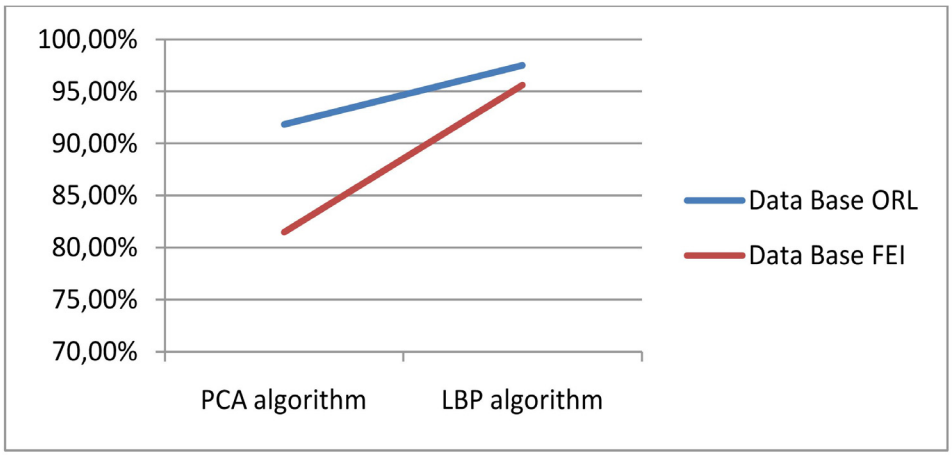

Figure 9. Dispersion of recognition rates according to databases

The facial recognition performance also depends largely on the environmental effects of image acquisition, the number of possible versions of an image corresponding to an individual used in the learning phase, and the different techniques applied to the identification as shown by the comparative table and the following curves [2].

Identification tool	**Identification rate**	**Number of image versions**
Eigen face (PCA)	65% à 95%	2 photos to 9 photos
ICA	67% à 96.36%	2 photos to 9 photos
LBP	94.5%	—
Gaussianface	98.52%	—
Humain recognition	97.53%	—

Table 2. Comparative table of facial identification rates for the main tools

6. About Jetson TK1

Jetson TK1 is a powerful development board based on the Tegra K1 chip. It comes with a GPU capable of OpenGL 4.4, OpenGL ES 3.1 and CUDA 6.5.

Tegra K1 is NVIDIA's first mobile processor to have the same advanced features and architecture as a modern desktop GPU while still using the low power draw of a mobile chip. Therefore Tegra K1 allows embedded devices to use the exact same CUDA code that would also run on a desktop GPU, with similar levels of GPU-accelerated performance as a desktop.

Jetson TK1 is NVIDIA's embedded Linux development platform featuring a Tegra K1 SOC (CPU+GPU+ISP in a single chip). Jetson TK1 comes pre-installed with Linux4Tegra OS (basically Ubuntu 14.04 with pre-configured drivers).

Besides the quad-core 2.3GHz ARM Cortex-A15 CPU and the revolutionary Tegra K1 GPU, the Jetson TK1 board includes similar features as a Raspberry Pi but also some PC-oriented features such as SATA, mini-PCIe and a fan to allow continuous operation under heavy workloads.

The Tegra K1 is about 15 times faster than the former Tegra 2, and is the most powerful GPU in the ARM world.

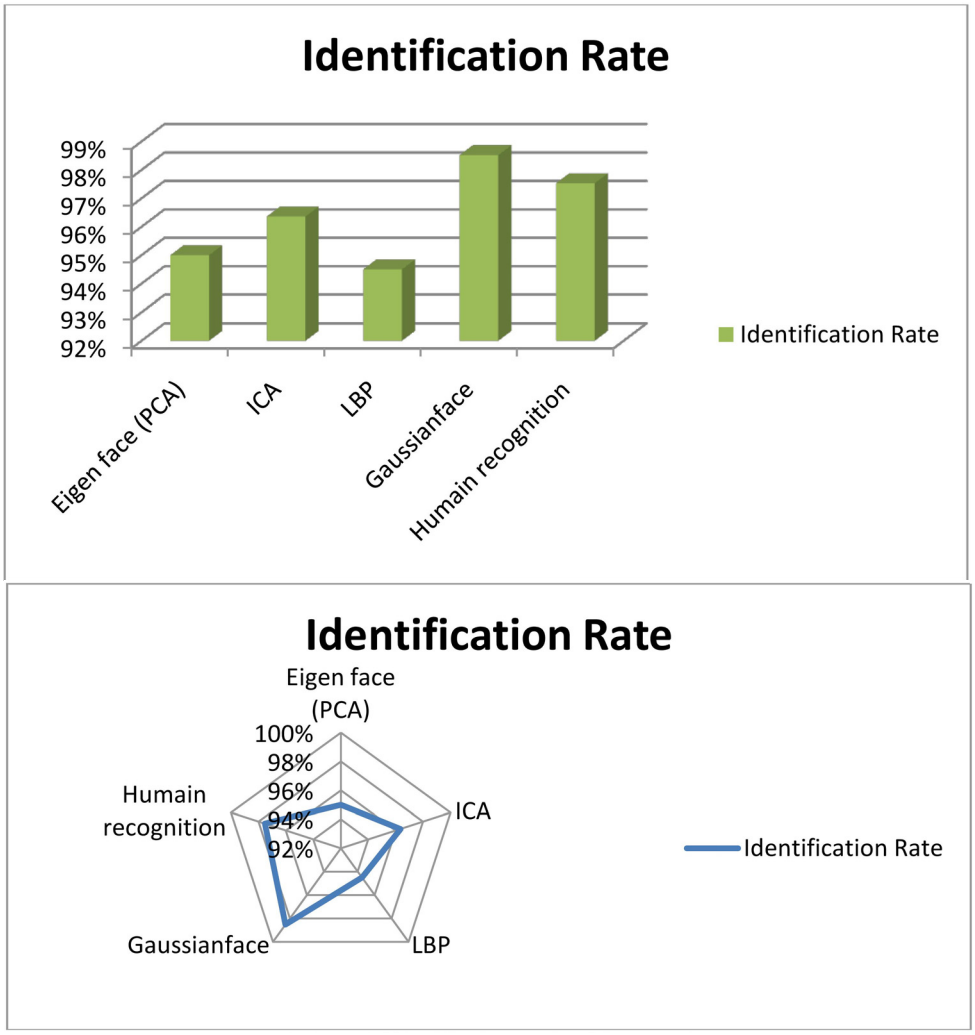

Figure 10. Facial identification rate for different tools

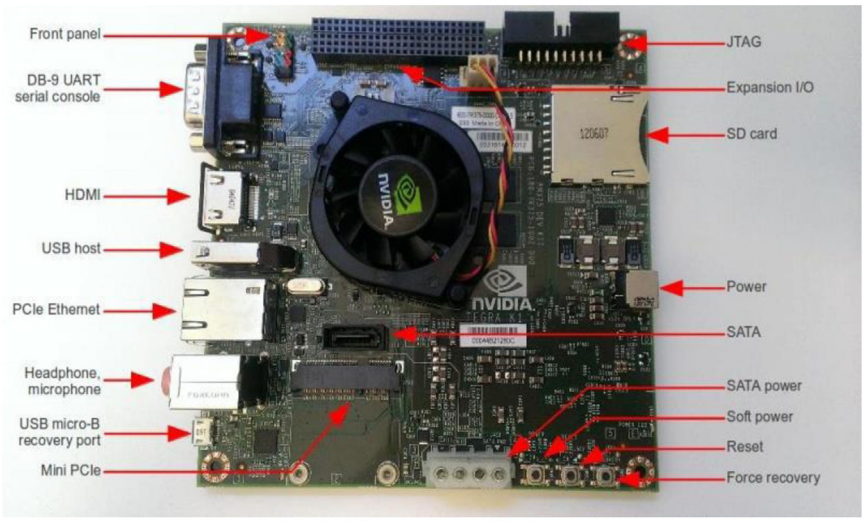

Figure 11. The Jetson TK1 board

JETSON TK1 is used in different field as Computer Vision, Robotics, Automotive, Medicine and Avionics.

The Jetson TK1 has a dimension board of 5" x 5" (127mm x 127mm), with typical power consumption between 1 to 5 Watts.

For the Jetpac installing we need to add exec permission for the jetpack.run through this command: $ chmod +x jetpack-${VERSION}.run.

7. GPU and CPU Comparison in Evolutionary Facial Recognition Algorithm

In this section, we are going to implement our approach for evolutionary facial recognition in both Jetson TK1 board and the computer CPU desktop, to compare the different results in terms of computational cost and time.

7.1 HoughLines

We are going to use the OpenCV functions HoughLines and HoughLinesP to detect lines in an example of four images, regarding our proposed model, in both Jetson GPU and Desktop CPU.

Figure 12. Example of a Source picture

Figure 13. Result of the CPU houghlines

Figure 14. Result of GPU Houghlines

	Picture 1		Picture 2		Picture 3		Picture 4	
	Time (ms)	**found Lines**	**Time (ms)**	**found Lines**	**Time (ms)**	**found Lines**	**Time (ms)**	**found Lines**
Jetson GPU	191.056	4096	84.065	482	120.10	82	245.148	5210
Desktop CPU	311.245	39	112.415	31	200.230	42	390.546	75

Table 3. Results of detecting lines in different pictures

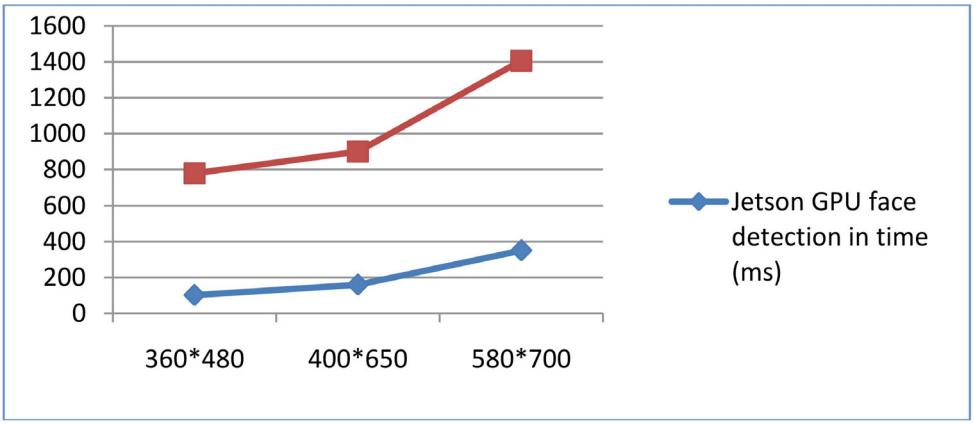

Figure 15. Comparison of detection times for image lines through Jetson and CPU

The results show that Nvidia GPU outperforms the Desktop computer CPU in processing time and performance of results. The results of detecting lines provided from Nvidia GPU, in all tested pictures, are greater than 80%.

7.2 Detecting face using Haar

For detecting face we are using Haar feature-based cascade classifiers. It is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is

a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images [3].

The algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used. They are just like our conventional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder.

First we need to load the required XML classifiers. Then load our input image in grayscale mode. Then after finding the faces in the image, the algorithm return the positions of detected faces as Rectangle (x,y,w,h). Once we get these locations, we can create a ROI for the face.

We process now to implement this algorithm in both Nvidia GPU and Desktop computer and compare results.

Figure 16. Detection faces with CPU

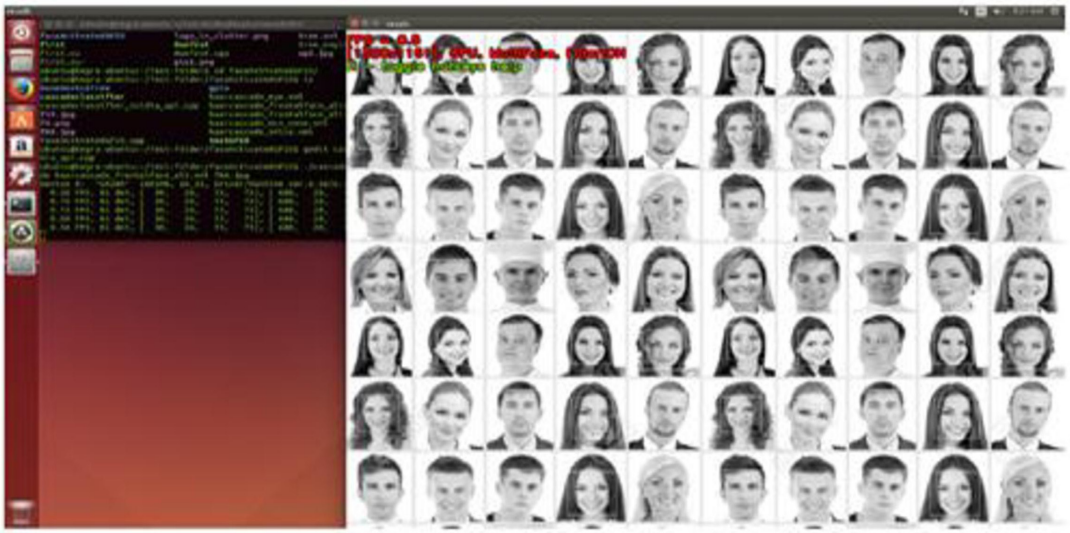

Figure 17. Detection faces with Jetson Tk1

The rate of detecting faces is different from picture to other, because it's depend on the size of picture, number of faces by picture and the scale factor used. As the above figure shows, the Nvidia GPU detects all 80 faces with rate of detecting one nonexistent face. In the other side, the desktop computer CPU detects a seventy-eight faces from 80 with also a rate of detecting one nonexistent face. Processing time is sensitive to the size of picture and the Jetson GPU is always faster than the desktop computer CPU as the Table shows below:

Image size	**360*480**	**400*650**	**580*700**
Jetson GPU face detection in time (ms)	102	160	350
Computer CPU face detection in time (ms)	780	900	1406

Table 4. Taken time for face detecting over GPU and CPU

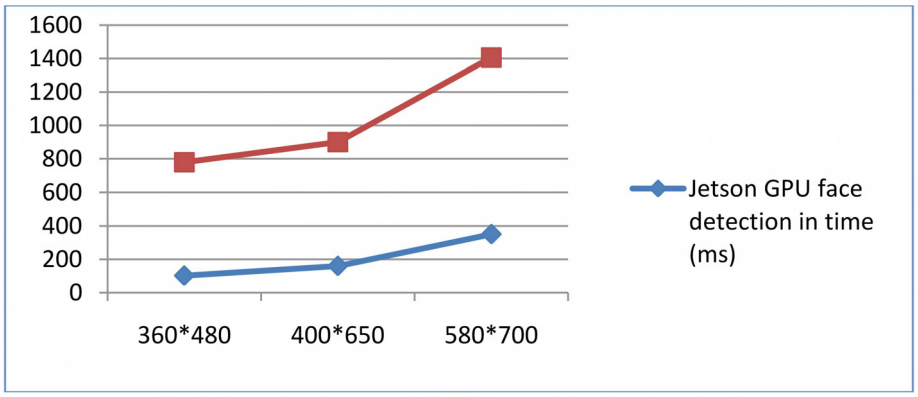

Figure 18. Comparison between CPU and Jetson Tk1 in face detection time

7.3 Eigenface Algorithm

The Eigenface technique is widely used for Face Recognition, which is a very active area in the Computer Vision and Biometrics fields, as it has been studied vigorously for 25 years and is finally producing applications in security, robotics, human-computer-interfaces, digital cameras, games and entertainment. "Face Recognition" generally involves two stages:

1. Face Detection, where a photo is searched to find any face, then image processing cleans up the facial image for easier recognition.

2. Face Recognition where that detected and processed face is compared to a database of known faces, to decide who that person is [4], [5].

Since 2002, Face Detection can be performed fairly reliably such as with OpenCV's Face Detector, working in roughly 90-95% of clear photos of a person looking forward at the camera. It is usually harder to detect a person's face when they are viewed from the side or at an angle, and sometimes this requires 3D Head Pose Estimation. It can also be very difficult to detect a person's face if the photo is not very brightor if part of the face is brighter than another or has shadows or is blurry or wearing glasses, etc [6].

Eigenfaces system use greyscale images. So we will easily convert color images to grayscale, and then easily apply Histogram Equalization as a very simple method of automatically standardizing the brightness and contrast of the facial images. For better results, we could use color face recognition, ideally with color histogram fitting in HSV or another color space instead of RGB, or apply more processing stages such as edge enhancement, contour detection, motion detection, etc.

So, for pre-processing, we need some data face. So we need some face images! We can either create our own dataset. But for not wasting time, we used one of the available face databases which is AT&T Face database.

The AT&T Face database, sometimes also referred to as *ORL Database of Faces*, contains ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling), and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

Once we have acquired a faces data, we ll need to read it in our program. We have to create CSV file to read the images from it. For creating this CSV file, we used a python code "create-csv.py" to make it easy.

In other case, the mathematically model of Eigenface algorithm is expressed as follow:

Let $x = \{x_1, x_2, ..., x_n\}$ a random vector with observations: $x_i \in R^d$.

1. Compute the mean μ.

$$\mu = \frac{1}{n} \Sigma_{i=1}^{n} x_i \tag{4}$$

2. Compute the covariance matrix S.

$$S = \frac{1}{n} \Sigma_{i=1}^{n} (x_i - \mu) . (x_i - \lambda)^T \tag{5}$$

3. Compute the eigenvalues: λ_I and the eigenvectors: V_i of matrix S.

$$Sv_i = \lambda_i . V_i, I = 1, 2, \ldots, n. \tag{6}$$

4. Order the eigenvectors descending by their eigenvalues. The K principal components are the eigenvectors corresponding to the K largest eigenvalues.

The K principal components of the observed vector X are then given by:

$$Y = w^T . (x - \mu) \tag{7}$$

Where, $W = (v_1, v_2, \ldots, v_k)$

The reconstruction from the PCA basis is given by:

$$X = W_y + \mu \tag{8}$$

The Eigenfaces method then performs face recognition by:

- Projecting all training samples into the PCA subspace.
- Projecting the query image into the PCA subspace.
- Finding the nearest neighbor between the projected training images and the projected query image [7], [8].

We used the jet colormap, so we can see how the grayscale values are distributed within the specific Eigenfaces. We can see that the Eigenfaces do not only encode facial features, but also the illumination in the images (see the left light in Eigenface #4, right light in Eigenfaces #5):

After that, we will try to reconstruct a face from its lower dimensional approximation.

We used the file "fichier3.csv" as a database of face and ~/Desktop/output/ as an output for our results.

Finally, by constructing a chromosome vector for each face, according to our proposed paradigm, taking into consideration the

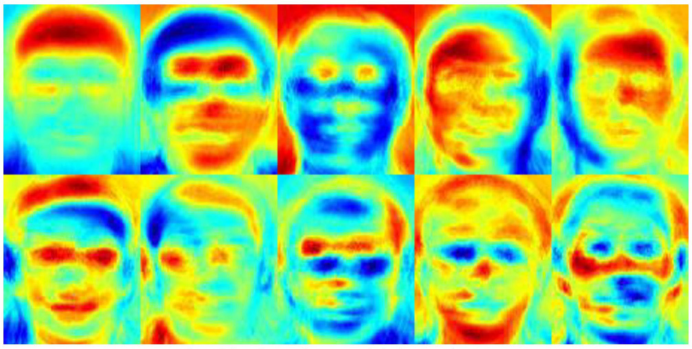

Figure 19. Eigenfaces with jet colormap

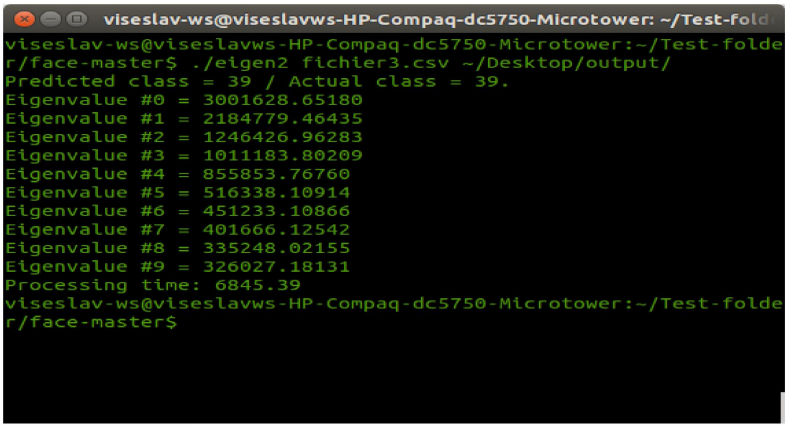

Figure 20. Reconstruction of face (1/2)

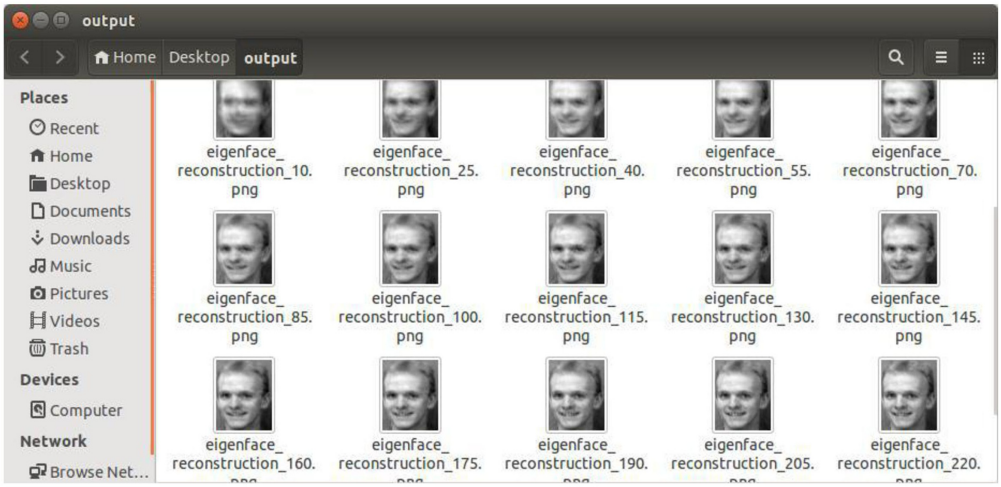

Figure 21. Reconstruction of face (2/2)

appearance of each model cited above, we can also make a comparison in time of execution of this algorithm by a CPU or a Jetson GPU.

The codes applied in the implementation of this technique on Jetson TK1 are instructions of the Python software.

This comparison will be done then with a big database of faces.

The results will be presented at the level of the table and the following curve.

Number of processed faces	100 faces	200 faces	300 faces	400 faces
Processing time (ms) over a CPU	8.1024	15.0977	21.7815	28.0049
Processing time (ms) over a Jetson TK 1	5.1132	11.0159	17.6704	24.1517

Table 5. Processing time for adopted model

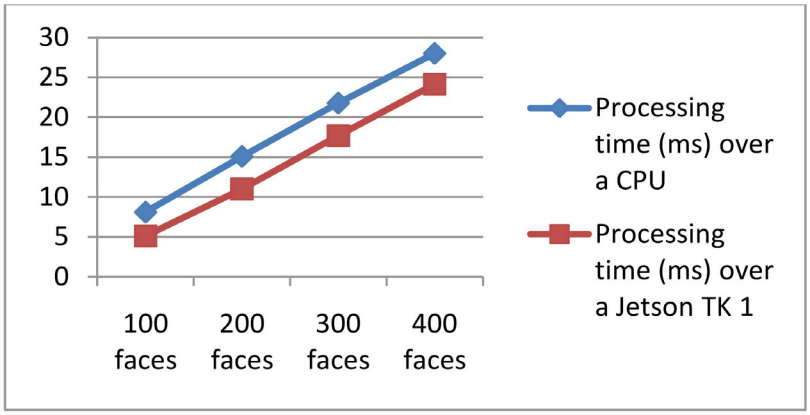

Figure 22. Comparison of detection times for proposed model over CPU and Jetson Tk1

The reconstruction of faces was done perfectly on both targets. The Jetson Tk1 GPU was faster in all different attempts with different size of faces database, as is shown above.

8. Conclusion

The face recognition subject is an area of current interest. Several researchers compete to treat this, each one with its own azimuth. In this sense we tried, in this paper, to present in the first case the main existing models. We have established the objectives and the recent challenges of this axis. Then we establish a contribution that can benefit from the common advantages of all the previous models by presenting an evolutionary vector called chromosome. It contains the specificities of each technique. Each of the specificities, alongside a so-called tint gene, constitutes a chromosome vector gene of the static and dynamic data which is involved in enhancing identification decision making. So, the obtained results represented in different pictures then in different tables and curves show well that the implementation of our evolutionary facial paradigm on Jetson TK1 board offers the possibility of the processing mechanism optimization of all sample vectors introduced in parallel, and then of stretching towards an RTS real-time system.

References

[1] Lu, Chaochao., Zhao, Deli., Tang, Xiaoou (2013). Face Recognition Using Face Patch Networks . University of Hong Kong, ICCV 2013, IEEE xplore.

[2] Cheng-Chin, Tai, Wen-Kai.,Yang, Mau-Tsuen., Huang, Yi-Timg., Huang, chi-Janng. (2003). A novel method for detecting lips, eyes and faces in real time. *Real-Time Imaging*, 9 (4) 277-287.

[3] Soyez, Fabien (2013). Reconnaissance faciale : plus vraiment de la science- fiction. CNET France. Janvier 2013.

[4] Willich, Martin (2008). 2nd End-User Group Meeting on 3D Face Recognition , February 2008, Berlin.

[5] Ming-Hsuan Yang, David J., Kriegman, Narendra Ahuja: (2002). Detecting faces images: A survey. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24 (1) 34-58.

[6] Van Wambeke, M. (2010). Reconnaissance et suivi de visages et implémentation en robotique temps-réel. Master Ingénieur Civil en Génie Biomédical. Belgique. Université catholique de Lovain.

[7] Viola, Paul., Jones, Michael. (2001). Robust real-time object detection . *In:* Second international work shop on statistical and computation theories of vision, Vancouver, Canada, (July 13).

[8] Viola, P., Jones, M. (2004). Robust real-time face detection . *International Journal of Computer Vision*, 57 (2) (2004), p. 137-154.