Available online at www.ijrp.org

International Journal of Research Publications

Character Recognition in a Human-Computer Interface Environment for Users with Motor disabilities

Andrea L. Piroddi[a*]

[a]*University of the People, Pasadena, 91101 California, U.S.A*

Abstract

The aim of this research is to explore a possible method for identifying an alphanumeric character produced by an individual with motor and vocal disabilities and whose possibilities to interact with the surrounding environment are limited to the movement of the face. The reason behind the creation of an alphanumeric character interpretation algorithm is linked to the fact that attempts to use existing algorithms (such as Tesseract) gave poor results in this scenario. The reason of the low success rate was not investigated; the focus was on finding a functional solution to the objective. Obviously subsequent studies in this sense are desirable. A Human-Computer Interaction algorithm has been developed in order to allow a person with limited mobility to interact with a computer, to which commands can be given. This project is based on three main elements: the identification of facial parameters, their tracking and the interpretation of the word expressed. A Linux based Python application has been coded to elaborate information from a camera. The main action is to identify the face of the potential user, activate tracking when the subject closes his eyes for a defined numbers of times, then follow the trajectory of a point of the face (in our case the tip of the nose), interpret the letter thus drawn, buffer the letters and finally convey the command (entire word) thus obtained. At the end, statistics on the success percentage are provided.

Keywords—Pearson Index, Image linearization and correlation, Character Recognition, Image Tracking

* Corresponding author. Tel.: +44.122.3790503.
E-mail address: andrea.piroddi@uopeople.edu.

1. Introduction

In Human-Computer Interaction and in Computer Vision, field gestures are of high Importance. In particular, to date there are many applications that use the recognition of the hand-gesture such as sign-language recognition, or those that use a finger as a pointer during Video conference presentations. From this comes the idea of our research, in which we expounded the applicability of these mathematical models in more complex areas as can be that of a severe motor disability combined with pathologies of the nervous system that can lead to serious forms of patient's communication difficulties. Over the years different approaches have been developed to make the patient independent in some way. Each of these solutions has potential disadvantages depending on the spinal pathology in place. Our objective, at the base of the study, was to obtain a software able to guarantee the patient a good level of autonomy in issuing commands to a centralized home automation server. The only limitation related to this solution requires that the user had a minimum motility of the head, lips and eyelids. On the other hand, the system does not require for any connection of the patient to sensors, or electrodes. The user himself activates and deactivates the streaming of commands. The algorithm takes as input the video signal of a webcam normally pointed at the user, identifies the face and extracts the main points (landmark detectors) such as the eyes, the nose, the mouth, and waits for the patient's command. In our case the beating of the eyelashes will be used, by the subject, to express the intention to write a command. This way, the algorithm switches status and starts tracking the tip of the nose, which the patient will move in order to describe the first letter of the command. It will be enough that the patient opens his mouth for a short period of time to move to the next letter. The algorithm will go on to interpret the next letter, until the patient completes the entire word. Finally, the user will give the command to proceed with the interpretation of the entire word and its association with the command to be activated. For example, the word “WATER” means that the subject wants to drink, so the automaton will activate the chain of events that will lead to the satisfaction of the expressed need. If the user changes position with respect to the plan of the screen during the use of the device, the system recovers automatically when the face returns to the image plan, starting from the point where it had stopped. The paper is structured as follows: the first part is dedicated to the presentation of the used methodology, algorithm and its specific characteristics, such as the eye blink detector to enable the process, nose tracking used to write digits and the algorithm used for digit/letter recognition, while the second part is aimed at showing the statistical results obtained.

1.1. Methodology

The first objective we pursued was to identify the subject's face in the real time image, so as to move on to identifying landmarks. One of the most used techniques for face detection in an image is the Haar Cascade Face Detector. The Haar Cascade based Face Detector has been the state of the art for many years since 2001, when it was introduced by Viola and Jones (Piroddi, Borgna, & Landoni, 2015). The main features of this type of mathematical model is that it works almost in real time on the CPU. The architecture is quite simple and can detect faces at different scales. On the other hand, the main disadvantage of this method is that it provides many false predictions, which in our application would significantly reduce the reliability of the entire project. Therefore, we have identified HoG Face Detector in Dlib as a valid alternative. It is a widely used face detection model based on the features of HoG (Histogram of Oriented Gradients) and SVM (Support Vector Machine) (Bristow & Lucey, 2014). Main advantages are greater speed on the CPU, it works very well for the frontal faces and slightly non-frontal, it manages to operate even in the presence of small occlusions. Obviously, some aspects can be improved. The biggest drawback, in fact, is that it does not detect small faces as it is trained for a minimum face size of 80 × 80. Therefore, it is necessary to make sure that the size of the face is within the minimum required value. Finally, it does not work for non-frontal faces, such as when looking down or up. The characteristics of the HOG features can be described as taking a non-linear function of the orientation of the edge in an image and bringing them together in small spatial regions to remove the sensitivity to the exact location of the edges. The choice of the non-linear function is discretionary. One possibility is to use $f(x) = |x|^2$. This function removes the edge direction, leaving only a function of the amplitude. The main reason for this choice, derives from the fact that in (Hyvarinen & Koester, 2007) is shown that from a statistical point of view squaring rectification fits image data very well. In HOG, when transforming from pixels to output, features can be represented as,

$$\Phi_f(x) = \boldsymbol{Db} * |(g_f * x) \odot (g_f * x)| \tag{1}$$

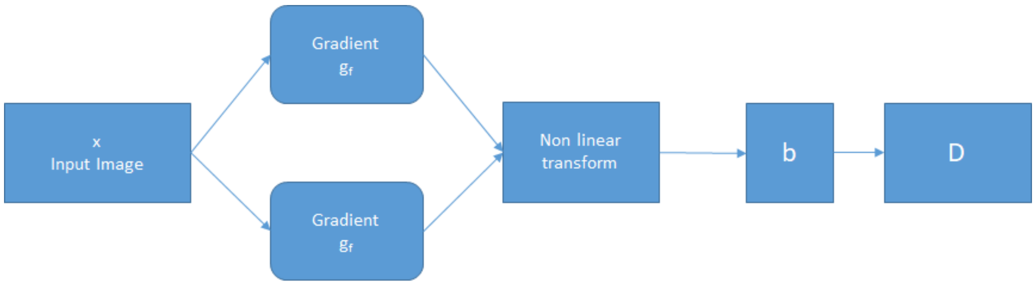


Figure 1 HOG Feature extraction process

Where $x \in \mathbb{R}^D$ is a vectorised input image, and it's convolved with g_f, an oriented edge filter, then it's rectified using the Hadamard operator, and finally blurred (using $\boldsymbol{b}$) and down sampled with $\boldsymbol{D}$, that is a sparse selection matrix, to achieve histogramming (pooling). Executing this action through a bank of OEF and concatenating the output, we get something like this:

$$\Phi(x) = [\Phi_1(x)\Phi_2(x)\Phi_3(x) \dots \Phi_F(x)] \tag{1}$$

In (Bristow & Lucey, 2012) is showed how each sub-descriptor can be written as:

$$\Phi(x) = L(x \otimes x) \tag{2}$$

So, we can compact

$$\Psi^*(x) = \begin{matrix} (e_1 * x)^T \circ x^T \\ (e_2 * x)^T \circ x^T \\ (e_m * x)^T \circ x^T \end{matrix}$$

(3)

In which $\{e_m\}_{m=1}^{M}$ Is the set of impulse filters which encode the local interactions in the signal.
This way we're preserving pixel's local quadratic interactions and so the classifier is able to discriminate natural from synthetic quite perfectly because they have enough capacity. Once we identified the algorithm to detect the patient's face, we focused on real-time eye blink detection.

1.2. Landmarks and Real-Time Eye Blink Detection

Detecting the eye-blink is fundamental to activate the tracking functions of the webcam. Basically, when the patient decides to give a command to the HCI, it is enough that he blinks. This way the algorithm will be activated to track the movement of a predefined point of the face and to interpret with a certain approximation what kind of letter the patient is trying to trace. The model we referred to is the one described in (Soukupova & Cech, 2016), which focuses on landmark detectors, demonstrating that the landmarks are detected precisely enough to allow reliable detection of eye closure. In particular, from the landmarks found in the image, an estimate of the eye opening is calculated. To do this, we rely on the EAR (Eye Aspect Ratio).

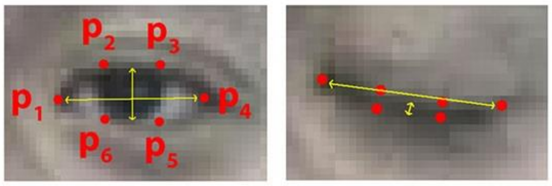


Figure 2 Open and closed eyes landmark P_i

The EAR between height and width of the eye is obtained as:

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

(4)

where p_i , with $i \in [1,6]$ are the 2D landmarks locations, see Figure 2. From the equation above we can see that the EAR is substantially constant when an eye is open, while it becomes null when an eye closes. The algorithm, iteratively, measures the value of the EAR parameter, if the patient keeps his eyes closed for a minimum of 5 seconds, the value of the EAR falls below a threshold value for a minimum time interval (EYE_AR_THRESH = 0.3, EYE_AR_CONSEC_FRAMES = 3) and this causes the web cam to track the landmark to follow. Let us focus for a moment on the accuracy in detecting the landmarks of a face. Quantitatively the latter is calculated with the mean relative error of localization, defined as:

$$\epsilon = \frac{100}{\delta N}\sum_{i=1}^{N}\|x_i - \hat{x}_i\|_2 \qquad (5)$$

Where x_i is the ground-truth location of landmark i in the image, see Figure 3, $\hat{x}_i$ is its estimate, N the number of landmarks that we consider in the model, and δ is the Euclidean distance between the centres of the eyes in the image, also known as the inter-ocular distance (IOD).

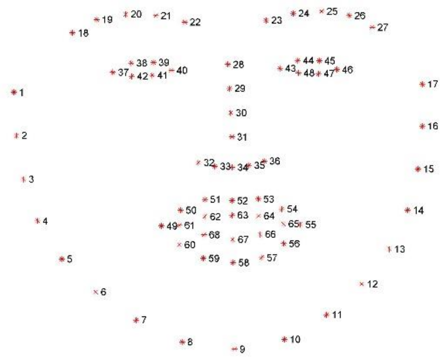


Figure 3 the full set of facial landmarks that can be detected

Through the recognition of the landmarks, once the webcam is ready to track, the algorithm identifies the 34th landmark (corresponding to the tip of the nose) and identifies it with a coloured dot on the screen so that the patient has clear the fact that the HCI is in waiting for his command. Consequently, the patient can begin to trace, with the tip of his nose, the letter he intends to transmit.

1.3. Landmark Tracking

The 34th landmark is identified on a reference system associated with the plan to which the image, received from the webcam, belongs. Therefore, we will instantly have a pair of x_i, y_i coordinates associated with the landmark position. From the instant (t_0) when the patient blinks, the coordinate vector is filled. When the letter has been composed, the algorithm stops filling the vector that is then passed to the plotting function. This function interpolates the points sequentially, thus creating the image of the letter that is saved in an image file.

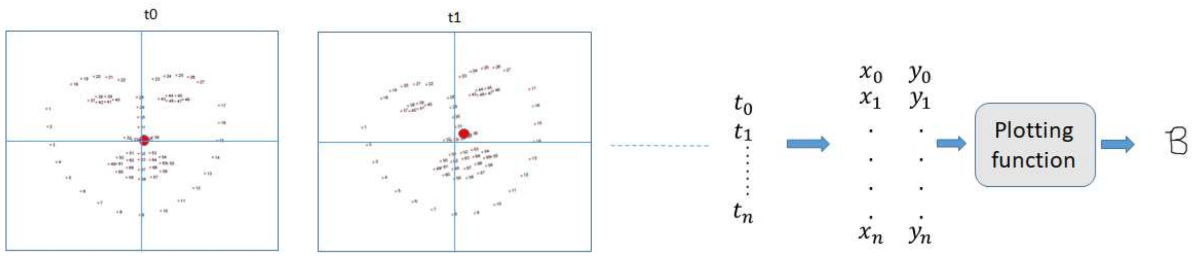


Figure 4 Sequence of events, starting from tracking landmark to image saving

The x_i and y_i coordinates belong to a reference system integral with the image. Being the width and the height of the image, respectively (w , h) we will have that $x_i \epsilon [0,w]$ and $y_i \epsilon [0, h]$. The image thus obtained will have a size of 256x256 pixels. Let us now make some considerations on the recognition of the letter. Initially, the use of Tesseract was attempted for this purpose. The Tesseract engine was originally developed as proprietary software at Hewlett Packard labs in Bristol, England and Greeley, Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows, and some migration from C to C ++ in 1998. The use of this engine is frequent in many application areas such as the recognition of car number plates or in reading the address in ordinary mail. Therefore, we expected that even in this area it could provide good results. Unfortunately, we have instead experienced a non-optimal behaviour, finding a high failure rate. We have not investigated the reasons for these results. We preferred to focus the research on creating a specific algorithm that was able to recognize the letters with a high success rate. The tracking algorithm was then used to generate and save samples for each letter (A, B, C, D, E ...). We repeat the process of writing the letters so as to have an huge number of samples for each letter. For example, 100 samples for the letter A, 100 samples for the B, etc. ... In this way we are preparing our training and testing images set. We therefore focused on the first five letters of the alphabet to verify the response of our recognizer.

2. Resizing and Linearization of training and testing images

Before proceeding with the recognition of the letter using our correlation algorithm, we must implement a further step to reduce the incidence of computational effort. We then implement a technique, commonly used in the field of image recognition, resizing. That is, starting from the image of the letter of size 256x256 pixels, we proceed to identify the only area of interest or rather that occupied by the letter. This will be nothing more than a rectangle having the width and height of the letter as its dimensions. So we get a cropped image that we are going to resize in a picture of resolution 28x28 (=784) pixels. We apply this methodology to all the images of our training and testing set.

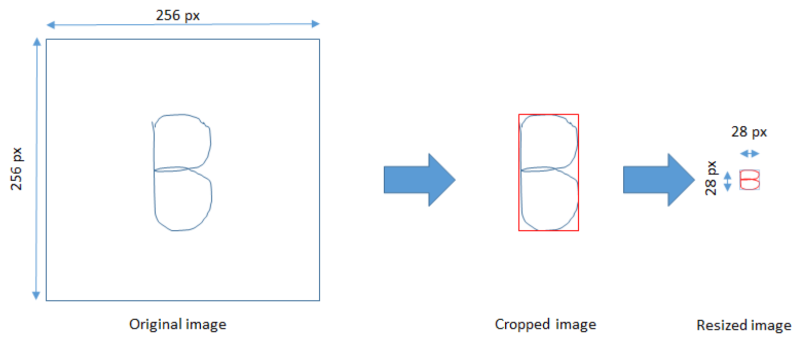


Figure 5 Image Resizing

Following the resizing of the image, see Figure 5, we proceed with its linearization. In this way we can treat the image on a numerical level as if it were a vector of dimension 1 row x 784 columns. Let us therefore consider the image 28x28 pixels, we take the grayscale version of it, thus obtaining an initial matrix composed of values between 0 and 255 and we proceed to its normalization. At this point we linearize it by transforming the first column into a row of 28 elements to which we are going to hang the second column and so on. We therefore pass by having initial files with *jpg* extension to final files with *npy* extension. In this way every image generated with our tracking algorithm has its correspondence in a numerical vector.

3. Correlation Analysis and Pearson Index

At the end of the linearization process we have a set of training images and a set of testing images with which we will verify the goodness of our interpreter. Consider keeping 80% of the images in the training set and 20% in the testing set. At the base of our interpreter the fundamental element is the analysis of the correlation between linearized images. Let's assume we introduce an image of the letter A into our algorithm, the code will calculate the Pearson correlation index for all the images in the training set and extract the letter that will obtain the highest value. Note that in statistics, Pearson's index is a measure of the linear correlation between two variables. According to the Cauchy – Schwarz inequality it has a value between +1 and −1, where 1 is total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation. Given two statistical variables X and Y, the Pearson correlation index (7), is defined as their covariance divided by the product of the standard deviations of the two variables:

$$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} \tag{7}$$

Where σ_{XY} is the covariance between X and Y, and σ_X and σ_Y are the two standard deviations. What we did was calculate the Pearson index for each pixel of the image to be recognized (linearized testing image) with the corresponding pixel of each image present in the linearized training set, see Figure 6. If we want to extend the correlation index to two vectors $\overline{X}$ and $\overline{Y}$, we get the Eq.8:

$$\rho_{\bar{X}\bar{Y}} = \frac{\sigma_{\overline{XY}}}{\sigma_{\bar{X}}\sigma_{\bar{Y}}} \tag{8}$$

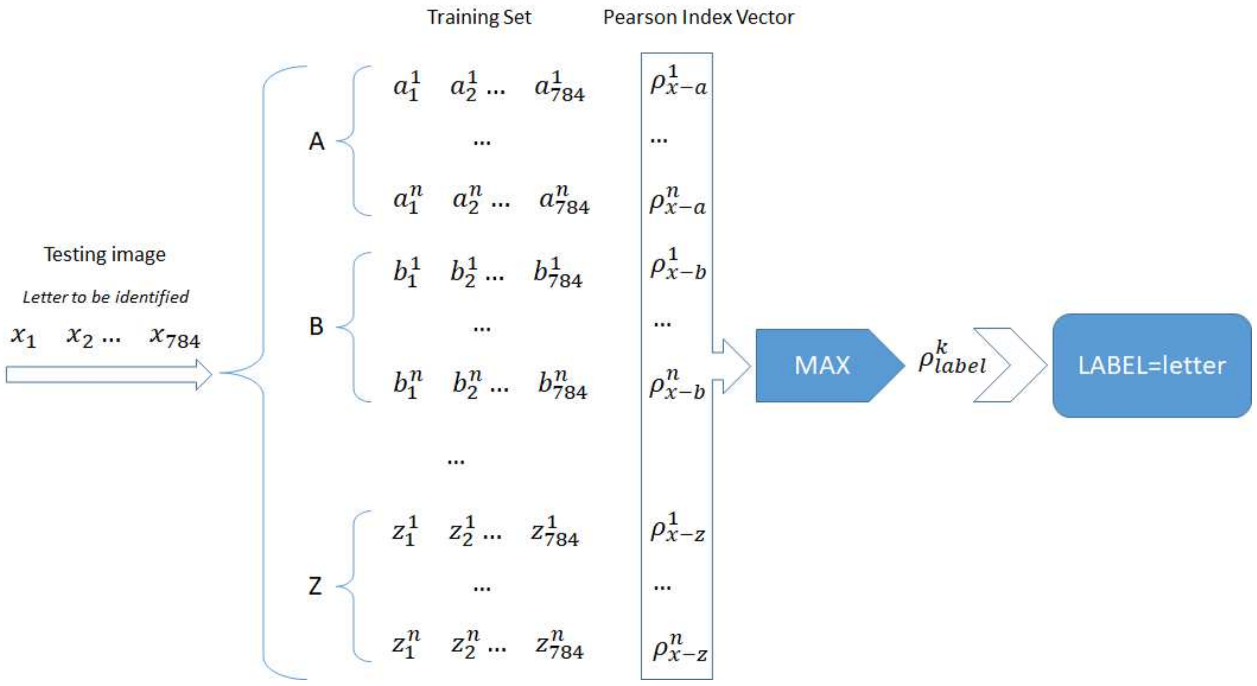

Training Set
Pearson Index Vector
Testing image
Letter to be identified
x_1 x_2 ... x_{784}
A
a_1^1 a_2^1 ... a_{784}^1
a_1^n a_2^n ... a_{784}^n
B
b_1^1 b_2^1 ... b_{784}^1
b_1^n b_2^n ... b_{784}^n
Z
z_1^1 z_2^1 ... z_{784}^1
z_1^n z_2^n ... z_{784}^n
ρ_{x-a}^1
ρ_{x-a}^n
ρ_{x-b}^1
ρ_{x-b}^n
ρ_{x-z}^1
ρ_{x-z}^n
MAX
ρ_{label}^k
LABEL=letter

Figure 6 Algorithm for Letter identification

4. Results and Conclusions

The results thus obtained were encouraging. In fact, although this is an area in which the identification of the letter is particularly complex due to various factors, such as the non-linearity of the tracking, a noisy environment in which the light intensity can undergo significant oscillations, etc… the percentage of false positives and false negatives is very low compared to the correct prediction, this both in case the average value of the Pearson index is used and in case the maximum value is used. From table 1 we can see how in the case of using the average value of the Pearson index the correct recognition of the letter occurs in the totality of the analysed cases, with a very low probability of error related to the standard deviation of the average value which percentage does not exceed 6%.

Table 1. Average of Pearson Index

		INPUT				
		A	B	C	D	E
output	A	0.411668	0.016395	-0.01273	0.044122	0.214219
	B	0.069302	0.252267	0.006358	0.164954	0.180179
	C	0.090665	-0.09439	0.553844	-0.05738	0.062453
	D	0.047107	0.084583	0.039731	0.294785	0.023598
	E	0.135556	0.102676	0.18445	0.026972	0.333754

Figure 7 clearly shows the behaviour just described. For each input letter, the values of the Pearson index in output prove to be consistent with the expected values. In the case of an incoming "*C*" for example, the result is a value of 0.55 greater than any other value obtained. If we even wanted to use the maximum value of the correlation coefficient, we would have a value of 0.73 (see Table 1) and concurrent values less than 0.44.

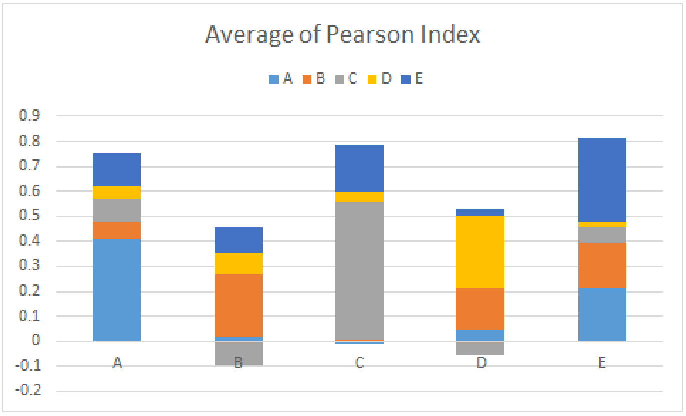


Figure 7 Average of Pearson Index

To try to give a clearer idea of the level of precision of the interpreter, we added **Errore. L'origine riferimento non è stata trovata.**, which shows the values of deviation from the mean value.

Table 2. Average Deviation of Pearson Index

		INPUT				
		A	B	C	D	E
output	A	0.092966	0.095973	0.058742	0.066742	0.096639
	B	0.070507	0.117179	0.048796	0.084052	0.101368
	C	0.076161	0.04425	0.090855	0.03732	0.085965
	D	0.076881	0.125203	0.074542	0.14419	0.092169
	E	0.081696	0.095591	0.078447	0.069434	0.09502

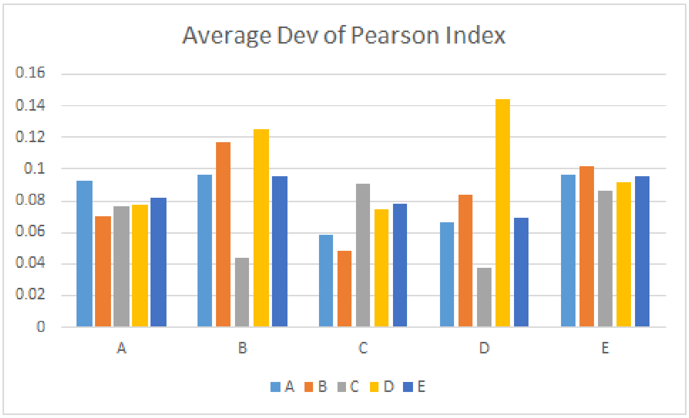
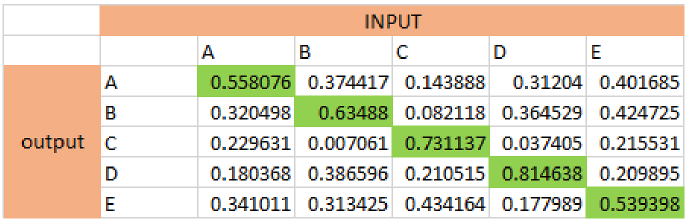


Figure 8 Average Deviation of Pearson Index



		INPUT				
		A	B	C	D	E
output	A	0.558076	0.374417	0.143888	0.31204	0.401685
	B	0.320498	0.63488	0.082118	0.364529	0.424725
	C	0.229631	0.007061	0.731137	0.037405	0.215531
	D	0.180368	0.386596	0.210515	0.814638	0.209895
	E	0.341011	0.313425	0.434164	0.177989	0.539398

Table 1 Maximum values of Pearson Index

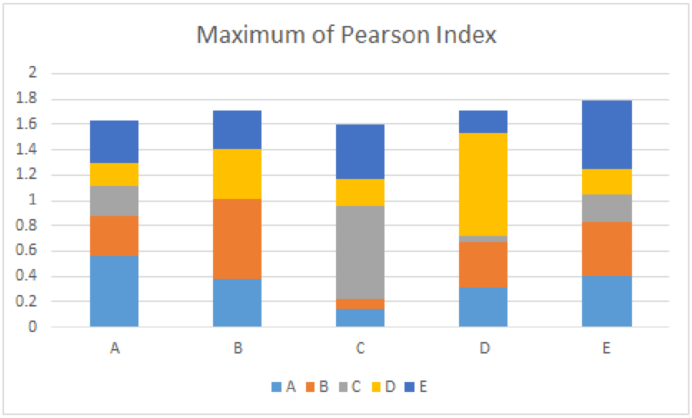


Figure 9 Maximum of Pearson Index

Concluding the study, we can say that the algorithm created for the intended purpose returns excellent results. In particular, using the maximum value of the Pearson index we note that the correlation is evident. The next step is to modify the algorithm so as to transform it into a Multilayer Perceptron, starting with the correlation index as an activation function. This should lead to having an algorithm more easily adaptable to different types of contexts and not limited to the one considered in the present study.

References

Bristow, H., & Lucey, S. (2012). V1-Inspired features induce a weighted margin in SVMs. European Conference on Computer Vision (ECCV), (pp. 1,2,5).

Bristow, H., & Lucey, S. (2014). Why do linear SVMs trained on HOG features perform so well? arXiv, 8.

Hyvarinen, A., & Koester, U. (2007). Complex cell pooling and the statistics of natural images. Network, 1.

Piroddi, A. L., Borgna, C., & Landoni, R. (2015). Facial Image Recognition And Motion Tracking Haar Technique. International Journal of Advancements in Digital Signal Processing, 1-5.

Soukupova, T., & Cech, J. (2016). Real-Time Eye Blink Detection using Facial Landmarks. 21st Computer Vision Winter Workshop. Rimske Toplice, Slovenia.