ISSN 1870-4069

Identification of Static and Dynamic Signs of the Mexican Sign Language Alphabet for Smartphones using Deep Learning and Image Processing

Bella Martinez-Seis, Obdulia Pichardo-Lagunas, Edgar Rodriguez-Aguilar, Enrique-Ruben Saucedo-Diaz

Instituto Politecnico Nacional, Interdisciplinary Professional Unit for Engineering and Advanced Technologies (UPIITA-IPN), Mexico
{bcmartinez,opichardola}@ipn.mx

Abstract. The Mexican Sign Language (MSL) is a language with its own syntax and lexicon. It is used by the deaf people, who use it to express thoughts, ideas and emotions. However, most of hearing people are unable to understand this language. The alphabet of any Sign Language (SL) is composed of signs where each sign corresponds to a letter of the alphabet of the dominant language in the region, for example, Spanish or English. Most signs of a signed alphabet are static, that means, they are only composed by the configuration of the hands. However, there are letters that are represented by signs that include movement. The present work proposes a system that, using artificial vision techniques and image processing, identify the 27 letters -including dynamic and static signs- of the Spanish alphabet in a mobile application. To solve the problem of sign identification it was used a combination of image processing techniques and deep learning. Canny and Camshift algortihms was implemented for the recognition of edges and trajectories in signs with movement. Once the characteristics were identified, the K-means and Tensorflow algorithms were used to classify the signs. The system achieves a 92% accuracy in the alphabet sign detection.

Keywords: Mexican sign language, automatic sign detection, trajectory tracking.

1 Introduction

The translation of any oral language implies a major challenge, since not all the structures of the original language can be correctly expressed under the rules of the second language. The challenge acquires a new level of difficulty when it comes to a sign language. The automatic translation of sign language can be done based on the oral language -normally are used written texts- or starting from a set of images that can be represent words or letters.

The alphabets of the different sign languages in the world are usually based on the dominant oral language used in their geographical region.

These alphabets use a different symbol for each letter, the symbols used can be static or dynamic. The static symbols only involve the configuration of the hand, the dynamic signs also include movement. The translation of sign language is usually approached as image processing. The identification of static signs is a common task but is not the case of dynamic signs, the identification of movement implies the use of different algorithms with greater degree of difficulty.

The present work proposes the identification of the signs that compose the Mexican Sign Language alphabet including the dynamic signs. For this job it is proposed use techniques of image processing and deep learning . Unlike other works the system reduces the tasks and processing time in order to perform this process on a mobile device.

2 Computational Tools for Automatic Sign Detection

To implement the automatic identification of the alphabet of a sign language it is essential to use different types of images or videos if the identification of dynamic signs is included.

The images or videos to be used must be pre-processed to identify their characteristics and then be classified in a category. The algorithms used in this pre-processing are described below.

2.1 Image Processing

Image processing is a technique that allows to improve and identify the characteristics of the images obtained for various applications. There are different techniques in image processing that have been developed during the last decades, however, the rapid advance in hardware and software technologies has allowed a greater development in recent years.

Digital image processing generally refers to the processing of a two-dimensional image by a digital computer [1]. A digital image is represented as a matrix of real numbers represented by a finite number of bits. Image processing techniques range from image pre-processing, image enhancement, feature extraction or image classification.

The pre-processing of images is responsible for correcting errors that may be related to the geometry or the brightness values of the pixels. These errors are corrected using mathematical models in order to improve the visual characteristics of the image or to represent it in a model that suits the possible applications[2]. The process of improving the image seeks to accentuate the characteristics of the image that allow a subsequent analysis. Some of these tasks are contrast and edge identification, noise filtering or scaling. The improvement process emphasizes the specific characteristics of the image. Image segmentation is the process that subdivides an image into its constituent parts or objects. The subdivision of the image depends largely on the problem that is being solved.

Image processing is a basic tool in the identification of symbols in different sign languages. Konwar et al. identified American Sign Language (ASL) using the HSV color model to detect the shape of the hand using skin color and edge detection[3]. Another work is the one made by Adithya et all in 2013 employed ANN forward backward algorithm to automatically recognize alphabets and numbers of Indian Sign Language with 91.1% of accuracy [4].

Image Segmentation The goal of segmentation is to identify the objects of interest in an image. Image threshold techniques are used for image segmentation [5]. The best threshold is the one that selects all the pixels of the object and assigns them to "black". The threshold can be defined as the gray scale assignment in the binary set {*0, 1*}:

$$S(x,y) = \begin{Bmatrix} 0 \text{ if } g(x,y) < T(x,y) \\ 1 \text{ if } g(x,y) \geq T(x,y) \end{Bmatrix}, \tag{1}$$

where *S(x, y)* is the value of the segmented image, *g(x, y)* is the gray level of the pixel *(x, y)* and *T(x, y)* is the threshold value at the coordinates *(x, y)*. In the simplest case *T(x, y)* is coordinate independent and a constant for the whole image. It can be selected, for instance, on the basis of the gray level histogram. When the histogram has two pronounced maxim, which reflect gray levels of object(s) and background, it is possible to select a single threshold for the entire image[6].

Segmentation of images involves sometimes not only the discrimination between objects and the background, but also separation between different regions. The segmentation of images is of vital importance in the recognition of signs. Identifying the components in an image will later allow to evaluate characteristics such as manual configuration.

Feature Extraction Feature extraction techniques are developed to extract characteristics that allow the classification of objects. The characteristics are the specific elements that describe an object can be: color, size, shape, location, etc. Segmentation techniques are used to isolate the objects of a plane to be later grouped according to their characteristics[7]. Once the pre-processing and segmentation has been carried out, some feature extraction technique is applied to the segments to subsequently apply classification techniques. The phase of extraction of characteristics is one of the most relevant processes in the recognition systems and has an important impact on the efficiency of the systems. The selection of techniques for extracting characteristics must be made taking into account various factors[8].

Canny Edge Detection To be able to detect the edges with the Canny method, the image P is first blurred and then it is convolved with a pair of orthogonal filters, like the Prewitt, to create two images H and V that contain the horizontal and vertical derived directions. For a pixel *(i,j)* with orientation θ_{ij} and magnitude i_j the gradient is calculated as follows[9]:

$$\theta_{ij} = \arctan\left[\frac{v_{ij}}{h_{ij}}\right], \tag{2}$$

$$a_{ij} = \sqrt{h_{ij}^2 + v_{ij}^2}. \tag{3}$$

Track Tracking There are different ways to calculate or follow the trajectory of an object or several. There are several algorithms that rely exclusively on artificial vision, using the camera of a device as the only source of information. This set of algorithms is known as video tracking. To carry out video tracking, an algorithm analyzes sequential video frames and determines the movement of the targets between the frames. Each of the algorithms has its strengths and weaknesses. The use or application sought must be considered before choosing the appropriate algorithm.

One of this options is the Meanshift Algorithm, the objective of this algorithm is to select a region of interest and evaluate it in a window where we look for the average value and move the center of our region of interest to this point. The algorithm continues this process until it finds a convergence of the area of interest. It must be assumed that every pixel that directs us to the same local maximum of convergence is part of the same region. That is why we do the same process in parallel, sweeping the image and finding how many local maxim exist. From this we label the image according to the maximum to which each pixel belongs.

Other option is the Camshift Algorithm that corrects the errors that may occur with the previous method. In the Meanshift algorithm the window always keeps the same size, even when the object moves away from the camera, which sometimes causes an inconsistent result. The size of the window needs to adapt to the size and rotation of the lens, CAMshift (Continuously Adaptive Meanshift) is responsible for addressing the problem. The CAMshift algorithm first applies Meanshift. Once Meanshift converges, update the size of the window and calculate the orientation of the best ellipse that fits in it. Again apply Meanshift with the new window and the location of the previous window. The process continues until the required accuracy is achieved[10].

Image Classification Image classification refers to the process of computer vision that allows you to group images according to their content. An image classification algorithm could indicate if an image contains the shape of a particular object. The robust classification of images is a challenge in the applications of artificial vision. The techniques used in this field will be described more precisely in the next section of this title.

2.2 Machine Learning

The techniques of Machine Learning are responsible for studying and computationally modeling learning processes in their various manifestations. It seeks to build a program that automatically improves with experience.

The Artificial Neural Networks (ANN), are inspired by the biological neural networks of the human brain. The minimum unit of RNA is an element that behaves, or tries to behave, similar to the biological neuron.

The ANN learn from experience and abstract the characteristics of a series of data [11]. Neural networks are composed of nodes connected through directed connections. A connection from unit j to unit i serves to propagate the activation a_j from j to i. In addition, each connection has a numeric weight W_j, i associated, which determines the strength and the sign of the connection. Each unit i first calculates a weighted sum of its inputs:

$$in_i = \sum_{j=0}^{n} W_{j,i} a_i, \tag{4}$$

finally applies an activation function g to this sum to produce the output:

$$a_i = g(in_i) = \left(\sum_{j=0}^{n} W_{j,i} a_i \right). \tag{5}$$

Much of the image classification tasks are performed using automatic learning techniques. The most commonly used are conventional neural networks, deep learning networks and convolutional neuronal networks. Traditional machine learning is based on networks composed of one input and one output layer with a hidden layer at most.

Deep learning networks are distinguished from common neural networks by their depth; that is, the number of node layers through which the data passes in a pattern recognition process. More than three layers (including entry and exit) qualify as learning.

Tensorflow. $TensorFlow^{TM}$is an open source software library for numerical calculation that uses data flow graphs. The nodes in the graph represent mathematical operations, while the edges of the graph represent arrays of multidimensional data (tensors) communicated with each other. Its flexible architecture allows to implement calculations in one or more CPUs or GPUs in a desktop, server or mobile device with a single API. TensorFlow was developed by Google Brain Team researchers and engineers within the Google Artificial Intelligence research organization to perform machine learning and deep neural network research, but the system is general enough to apply to a wide variety of other domains.

3 Sign Detection Model in Smartphones

We propose two parallels solutions to alphabet sign detection that involves hand gesture: the first one is focused on the signs that are static and the second one in the ones that are dynamic. In both of them we have some common stages as we can see in Fig. 1.

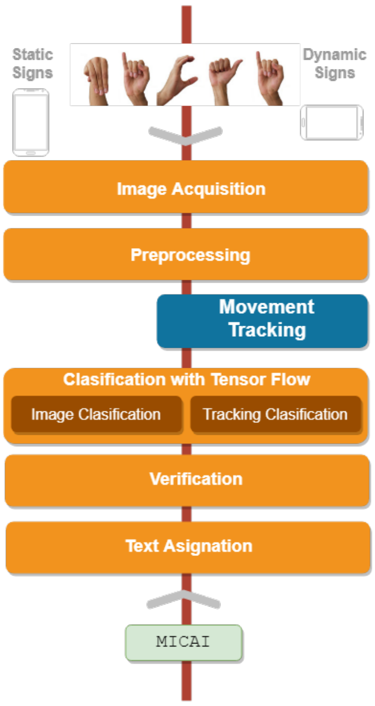

Fig. 1. Process for sign detection in smartphones.

First we do the image acquisition throw the smartphone and the image is prepossessed. Then, it follows the classification stage that depends on the type of sign (static or dynamic), for the static signs the image classification using tensor flow is the most important part, while in the dynamic signs the classification is simpler but it requires previous stages. Finally, there is a verification and text assignation stage to show the corresponding letter of the alphabet in an mobile application.

In the following sections, we will explain the stages that depend on the type of sign, such as classification and the ones for dynamic signs. Then we will show the integration of those stages to the model and to the smartphone.

3.1 Static Signs Detection with Deep Learning

In static sign detection, even when the signs are statics, there is a continuous image acquisition. The systems has to select when do we have a new sign to translate. To do so, we process continuously the images, and we selected them through the verification phase.

For each image, the system has to classify it into one of the 21 possible letters that correspond to static signs. We use TensorFlow because it offers first level solutions for computational learning problems, it helps us to recognize the signs of the MSL alphabet. Two of the main models are Mobilenet and Inception V3, the first one uses depthwise separable convolution while Inception V3 uses standard convolution. Then, Mobilnet requires less parameters than Inception V3 but also there is a slight decrease in the performance. Nevertheless, Inception V3 is optimized to be precise, while the MobileNet is optimized to be small and efficient (as expected in a classifier that operates on a mobile device) at a minimum cost of accuracy, so we selected MobilNet [12]. Considering the decrease of performance, with Mobilnet, a previous stage was used in the images as we can see in Fig. 2.

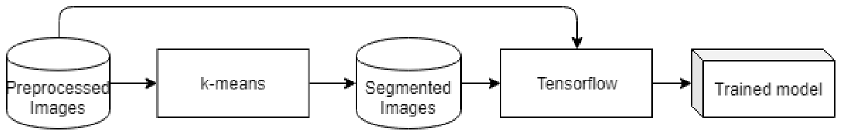

Fig. 2. Training data process for static sign classification.

In the training phase, with Tensorflow algorithm, the algorithm uses original images and segmented images. Segmentation is performed by k-means algorithm, it allows us to get two segments: one for the background and clothes and the other one for the hand. The Figure 3 shows an example of the two detected segments. For training phase, we use the ones that contain our Region of Interest (RoI) that involves the hand and the other one was discarded.

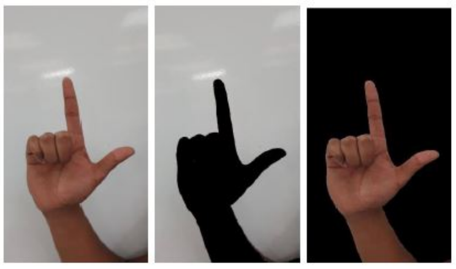

Fig. 3. Original image (left); Discarded image (in the middle); Used segment (right).

3.2 Dynamic Signs Detection with Deep Learning

There are six dynamic signs corresponding to letters; "J", "K", "N", "Q", "X" and "Z". The dynamic sign detection requires an image processing of a series of images in a time window of three seconds... For dynamic signs, we need algorithms capable of identifying an object and tracking it, then the process we follow is shown in Figure 4. First, a RGBA Selection is done in order to identify the hand through its colors, and then we convert the image to gray-scale. Second, the hand tracking is performed using the CamShift algorithm. Third, the tracking generates a trajectory that is drawn in a canvas, its pixels are used for the classification process. Finally, the Bitmap of the canvas pixels is generated because it is used for the process that recognize the image getting the translation of the signs.

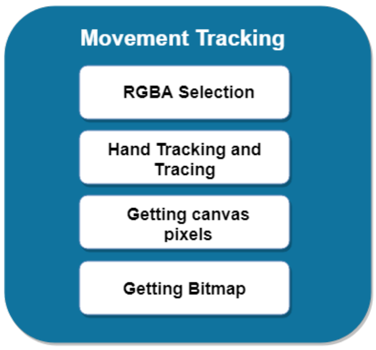

Fig. 4. Movement tracking process.

All tracking algorithms, basically, have the same principle of operation: segmentation, edge detection, create binary masks and enclose the object in a tracking window. Some steps can be omitted or used in a different order, but the method is generally the same.

By comparing the different trajectory tracking algorithms, with CamShift its window adapts to the distance and rotation of the object with respect to the camera, while in Meanshift, the window remains the same regardless of the distance from the objective. Camshift first applies Meanshift, but it adjusts the window according to the total movement of the objective despite the processing consumption.

Tracking by color-based methods it is possible to find the coordinates of specific points in the hand, which are stored as soon as a dynamic signal is executed (limited by a time window). After performing a series of operations on these coordinates, it is possible to construct an image like the one in Figure 5 with an artificial equivalent.

With hundreds of trajectories used as training information, the model generated is able to identify patterns and associate them with a trace obtained by the smartphone application of the system. The inference in the mobile is then made with a second model, in a process totally independent of the recognition of the static signs.

The algorithm of Tensorflow receives the image as a Bitmap, it obtains the pixels in an integer array, copies the input information to the algorithm,

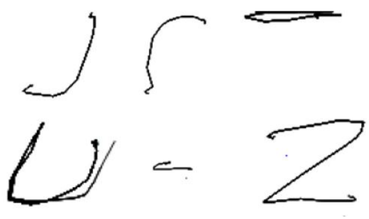

Fig. 5. Track of letters "J", "K", "N", "Q", "X" and "Z".

and proceeds to read the results thrown by the model, finally, by means of ordering methods, create a list of probabilities finding the better classification. The classifier is independent of the static signs and sends the Bitmap generated by the description of the trajectory of the hand.

3.3 Integration of the Classification to Smartphone

In order to classify we need two previous stages: Image Acquisistion and Verification. And once we have the classification we have two final stages. All of them are explained in the following section. Once we have define those stages the complete model integrated to the smartphone is presented.

Common Stages for both Types of Signs. The commons stages are image acquisition, pre-processing, verification and text assignation, those work as follows.

Image Acquisition. This stage has the purpose of capturing continuous images of the sign that is going to be translated using the camera of the device. Because the aim of the system is the translation of the entire MSL alphabet, it was required to implement two *views* in different *layouts*. It mainly depends on the smartphone orientation. For the statics signs the system uses the natural device orientation (vertical) and it gets continuous images. For the dynamic sign, the images are capture, the tracking process gives a trajectory that was process to get the corresponding bitmap that correspond to the pixels of the path that the hand follows, this bitmap is used to classify the letter. For the static signs, the smartphone captures the images from the camera of the device and uses a *TextureView* to show it inside the application. For the dynamic signs, the application uses an OpenCV view integrated with a canvas to the program, so that the user see the camera, the trajectory of the hand and the app interface.

Preprocessing. The images for the training phase and the pictures that are taken from the smartphone have a preprocessing phase. This consist of a transformation of the image to grayscale and then an edge detection phase using the Canny Algorithm with double thresholding as we can see in Figure 6.

Verification. Because of the continuous reception of the images to be process, the system does previous verification:

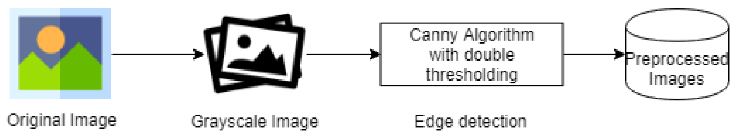

Fig. 6. Pre-processing the image to get the training set.

– That the previous letter is different, except in the case of the first written letter. This rules out the existence of results such as "aa", "bb", and so on that are not used in spanish.
– The probability thrown by the classifier model must be greater than 0.9.
– There must be 3 readings (of the same symbol) with a probability greater than 0.9 before displaying the translation.

Text Assignation. Once the verification is done, the letters are display to form words.

Integration to Smartphone. The training phases of the proposed model were processed in a computer, the results of this phase is used by the application of the smarthphone. The system uses three main elements of the device: the camera, the display and the processor. The camera is used to continuously take pictures. Then the device preprocesses the images and used them on the classifiers using resources of the smartphone. During the use of the application the shown images correspond to the ones taken by the camera, and for the dynamic signs, over the images, the application displays the trajectory of the sign.

The application detects the orientation of the smartphone, with the vertical position the application waits for a static sign and in the horizontal position it waits for a dynamic sign.

4 Experiments and Results

4.1 Tensorflow Validation for Sign Classification

The resolution of the input images were 128, 160, 192, or 224 pixels. Which is enough considering that images are finally taken from smartphones. We compare two of the main algorithms of Tensorflow: Inception 3 and MobileNet. In the tests carried out, the Inception V3 model uses approximately 85 MB of space, even after performing compression operations; MobileNet models, on the other hand, uses 5 423, 10 343 and 16 797 KB of space, depending on the selected architecture. In short, the Inception V3 model is five times heavier than the more complex MobileNet, considerably slower, and it did not offer a significant advantage in terms of accuracy. The configuration of all the tests remained fixed in terms of the resolution of the input images: 224x224 pixels. The relative size of the network was varied throughout the tests, finding that 1.0 and 0.75 of the largest MobileNet obtained the best results.

We worked with transfer learning, which means that we started with a model that had been previously trained to solve another problem. A model trained in the ImageNet Large Visual Recognition Challenge data set was used and re-trained to differentiate between a small number of classes (21 signs).

4.2 Training Model for Static Signs

The used Mobilnet has a learning rate of 0.015 with 4000 training epochs with 795 images per each category, so that the size is up to 16797 KB. For the recognition of the static signs, more than 30 models were trained with different parameters and using different sets of photographs. Table 1 summarizes the parameters and results of some of the most relevant models, that is, those that presented significant improvements with respect to the previous batch of tests. The % accuracy depends on 15 attempts for each static sign.

Table 1. Classifiers for static signs.

	Learning loop	Epochs	MobileNet Section	Images by category	k-means Images	Accuracy (%)	Failure
1	0.5	500	0.5	100	0	47.60	d,e,f,i,r,s,t,v,w,y
5	0.0001	4 000	0.75	400	0	69.04	d,i,l,p,t,y
9	0.001	4 000	0.75	500	0	77.38	d,i,t
14	0.0001	4 000	1	550	0	52.80	d,e,i,m,n,r
15	0.001	4 000	1	550	0	78.57	i,s,t
17	0.001	4 000	1	600	70	75	d,p,t
K	0.001	4 000	1	70	70	52.38	a,b,d,i,l,m,n,s,t,u
27	0.01	4 000	1	630	100	83.33	p,s,t
29	0.01	4 000	1	630	150	85.71	i,t
30	0.04	4 000	1	795	150	88.09	s,t
31	0.02	4 000	1	795	150	90.47	t
32	0.015	4 000	1	795	150	95.23	-

We performed experiments with different backgrounds and similar light conditions, Table 2 shows the results of those experiments where the background not only changes on color but also on texture. When we have a white background the contrast increases so the accuracy is high, up to 93%. The wort case is with not smooth surface because the shadows of the texture of the background make noise to the algorithm of border detection and the classification fails. In that case, the signs that are difficult to classify are the ones that look like the fist, or that only have one finger up.

Table 2. Test with different backgrounds.

Background	RGB	HSV	Accuracy (%)	Failure
Random background, not smooth	-	-	52	c, i, l, n, p, s, t
Dark red background	#B62906	(12°, 97%, 71%)	60	c, i, l, n, p, s
Black background	#191114	(330°, 25%, 7%)	74	c, i, l, p, t
Light gray background, rough surface	#BDB6B3	(18°, 5%, 74%)	76.1	c, i, p
Light gray background	#BDB6B3	(18°, 5%, 74%)	85.9	p, t, r
Light gray / light blue background	#BAB6BB	(288°, 3%, 73%)	89.71	p, t
White background	#F0FBFF	(196%, 6%, 100%)	93.1	s

4.3 Training Model for Dynamic Signs

The used Mobilnet has a learning rate of 0.02 with 4000 training epochs with 110 images per each category (a seventh amount of static images). For the recognition of the dynamic signs, we tested it with the best results if static signs. Table 3 summarizes the parameters and results of some of the most relevant models, that is, those that presented significant improvements with respect to the previous batch of tests. The % accuracy depends on 15 attempts for each static sign.

Table 3. Classifiers for dynamic signs.

	Learning loop	Epochs	MobileNet Section	Images by category	Accuracy (%)	Failure
1	0.02	4 000	1	15	$\approx$ 66.66	q, k
3	0.02	4 000	1	50	$\approx$ 83.33	q
4	0.02	4 000	1	110	$\approx$ 92	

Once the trajectory of the sign is read properly, the classification process is simpler, since there are only 6 dynamic signs, which facilitates the work of the classifier model and improves its accuracy. In addition, each of the trajectories has particular characteristics and is easily distinguishable. Even in the second of the tests, which had a training corpus of only 50 images per category, an acceptable level of accuracy was achieved.

5 Conclusions

We proposed a two phases training model, the first one for image segmentation using k-means, and the second one for image recognition with tensor flow. This process increases the accurancy of sign detection because the k-means method allowed to improve the extraction of particular traits in some signs, identifying false positives.

The implementation of trajectory tracking algorithms allowed to identify signs with movement grouping them according to their trajectory with a success of at least 90%.

The need of using a glove of contrasting color was discarded because the precision achieved with the natural color already exceeded 90%.

Separating the identification stages by performing the training in an external device allowed the process to be efficient, leaving the smartphone only with the task of classification.

References

1. Young, I.T., Gerbrands, J.J., Van Vliet, L.J.: Fundamentals of Image Processing, 2nd ed., pp. 2–32 (2007)
2. Chitradevi, B., Srimathi, P.: An Overview on Image Processing Techniques. International Journal of Innovative Research in Computer and Communication Engineering IJIRCCE 2(11) (2014)
3. Pansare, J., Ingle, M.: Vision-based approach for American Sign Language recognition using Edge Orientation Histogram. In: International Conference in Image, Vision and Computing. IEEE (2016)
4. Adithya, V., Vinod, P.R., Gopalakrishnan, U.: Artificial Neural Network Based Method for Indian Sign Language Recognition. IEEE (2013)
5. Nixon, M., Aguado, A.: Feature extraction and image processing for computer vision. 1st ed. Elsevier, Academic Press, Amsterdam (2013)
6. Monga, P., Ghogare, S.: Scrutiny on Image Processing. International Journal of Advanced Research in Computer Science and Software Engineering 5(1) (2015)
7. Rao, R.M., Arora, M.K.: Overview of Image Processing. In: Varshney, P.K., Arora, M.K. (eds.) Advanced Image Processing Techniques for Remotely Sensed Hyperspectral Data, pp- 51–85. Springer, Berlin, Heidelberg (2004)
8. Kumar, G., Bhatia, P.: A Detailed Review of Feature Extraction in Image Processing Systems. In: 2014 Fourth International Conference on Advanced Computing and Communication Technologies, IEEE (2014)
9. Canny, J.: A Computational Approach to Edge Detection. IEEE Trans. Pattern Analysis and Machine Intelligence 8(6), 679-–698 (1986)
10. OpenCV: Meanshift and Camshift. Docs.opencv.org. (2015) `http://docs.opencv.org/3.1.0/db/df8/tutorial_py_meanshift.html`. Último acceso: 01/04/2018
11. Dreiseitl, S., Ohno-Machado, L.: Logistic regression and artificial neural network classification models: a methodology review. J Biomed Inform 35(5–6), 352–359 (2002)
12. Codelabs.developers.google.com. (2017). TensorFlow For Poets. [En línea] https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/. Último acceso: 16/11/2017.