A Computer Vision Method for the Italian Finger Spelling Recognition

Vitoantonio Bevilacqua[1(✉)], Luigi Biasi[1], Antonio Pepe[1], Giuseppe Mastronardi[1], and Nicholas Caporusso[2]

[1] Dipartimento di Ingegneria Elettrica e dell'Informazione, Politecnico di Bari, Bari, Italy
vitoantonio.bevilacqua@poliba.it
[2] INTACT healthcare, Bari, Italy

Abstract. Sign Language Recognition opens to a wide research field with the aim of solving problems for the integration of deaf people in society. The goal of this research is to reduce the communication gap between hearing impaired users and other subjects, building an educational system for hearing impaired children. This project uses computer vision and machine learning algorithms to reach this objective. In this paper we analyze the image processing techniques for detecting hand gestures in video and we compare two approaches based on machine learning to achieve gesture recognition.

Keywords: Image processing · Computer vision · Machine learning · SVM · MLP · Gaussian Mixture Model · Sign language · LIS

1 Introduction

Hearing-impaired people usually communicate using the visual-gestural channel which is notably different from the vocal-acoustic one. To this end, sign language is a complete language having its own grammar, syntax, vocabulary and morphological rules. Furthermore, each community usually develops and employs its specific language. As a result, there are many different languages based on signs, such as the Italian Sign Language (LIS), the American Sign Language (ASL), or the British Sign Language. Additionally, each language has its vernacular variants and, similarly to spoken languages, a constantly evolving lexicon.

The Italian Sign Language - as other sign languages - is based on an alphabet, commonly referred to as finger spelling. Despite being not largely used in conversations, finger spelling is crucial, both for beginners and in communication. Indeed, it is employed to represent names of people or places, and to replace signs which are harder to remember. The LIS finger spelling represents all the 26 letters of the Italian alphabet, as shown in Fig. 1: some letters are associated with a static gesture, others include hand movement.

The goal of this research is to realize a system for detecting LIS gestures and for translating them into written or spoken language (e.g., with the help of text-to-speech systems), as this could be employed in communication technology or educational tools to provide hearing impaired people with enhanced interaction, simplified communication and, in general, with more opportunities of social inclusion.

DOI: 10.1007/978-3-319-22053-6_28

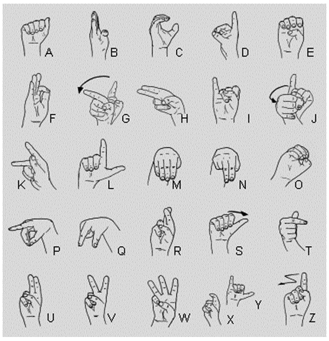

Fig. 1. LIS finger spelling

2 Related Work

In the last years, research showed that hardware/software solutions based on automatic recognition systems can play a crucial role in helping sign language users communicate more easily and efficiently. Also, several projects integrated them in multimedia platforms to address applications accessibility and to increase social inclusion of users having some degree of hearing deficiency.

Recently, promising results have been achieved with RGB-D sensors. However, they require users to be constantly connected to power supply, which is an important drawback especially when mobility is required. Moreover, they are not currently available on commercial mobile devices.

Several experimental projects have been realized in this field; nevertheless, they usually involve very complex equipment and sophisticated settings. For instance, a large number of applications utilize Microsoft Kinect, which is not portable. Others are exploring wearable solutions, such as sensor-equipped gloves [7], which require users to have both their hands busy. Conversely, we adopt an alternative approach based on cameras and low-cost devices with the aim of designing a portable solution especially dedicated to enabling the deaf use sign language in mobility.

In this work, we introduce SignInterpreter, a Sign Language detection system based on RGB sensors, as they are incorporated in the majority of nowadays available mobile devices. Also, we detail an experimental study in which we compare the performance of our solution using different image processing algorithms.

In [6, 9] two methods that provide sign recognition with image processing and machine learning method have been proposed. In [6] a Random Forest Algorithm is trained with Hu moments, in [9] a SVM is trained with Zernike moments and Hu moments too.

3 SignInterpreter

3.1 System Architecture

SignInterpreter is a Sign Language recognition system designed to work with standard RGB cameras, such as webcams, and cameras mounted on mobile and embedded devices: users communicate by realizing gestures in front of the camera. The system acquires the video stream and processes the captured frame in order to extract features that are converted into digitized text. This, in turn, can be utilized to represent messages, or to control applications. The system architecture is shown in Fig. 2.

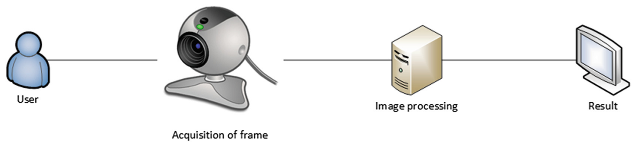

Fig. 2. System architecture

In addition to the recognition hardware, SignInterpreter comprises a client-server architecture (Fig. 3) specifically designed to increase the performances of the system on mobile devices: video streams and image features are acquired on the client; image processing and machine learning tasks are executed on the server, in real-time. In addition to enabling a larger dataset to be collected and used, this improves the overall accuracy of the system, regardless of the computational power of the client.

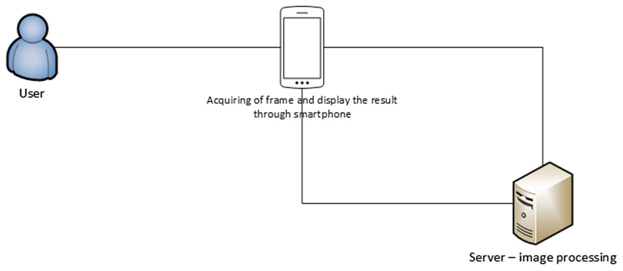
User
Acquiring of frame and display the result through smartphone
Server – image processing

Fig. 3. Distributed architecture

3.2 Software Architecture

The software operates in three main phases: (1) *background acquisition and modeling*; (2) *calibration* to get the user's skin color; (3) *hand detection and gesture recognition*. The first step removes the noise produced by background objects. During calibration, the system collects several color samples of the hand and utilizes them to obtain a precise model of the skin color of user's hand. In the last phase, foreground is extracted from a generic frame: this is realized using information from the background model acquired in the first phase, and the hand model. Then, a segmentation process is executed on the resulting image. Specifically, the algorithm discards all colors considered as different from the color model of user's skin. Subsequently, the Canny edge detection algorithm is employed to extract the contour of the hands. This is the input to a classifier that is trained to recognize gestures. The system workflow is shown in Fig. 4.

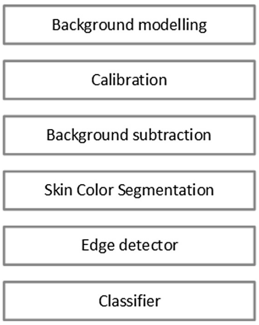

Fig. 4. Workflow

4 Hand Detection

This stage consists in extracting foreground objects from the scene, so that segmentation can be realized [1]. Background subtraction is crucial in order to reduce the number of false negatives in the segmentation phase.

4.1 Background Subtraction

In [4], several background subtraction algorithms are discussed. In our experimental study, we employed the Gaussian Mixture Model (GMM) algorithm [3] to discriminate background. The GMM algorithm models the value of each pixel with a mixture of *K* Gaussian distributions and it detects the pixel intensity that most likely represents the background, using a heuristic method. If a pixel does not match the intensity value, it is recognized as a foreground pixel.

The likelihood to observe an intensity pixel value x (x_R, x_G, x_B) at the time t is expressed by:

$$p(x_t) = \sum w_{i,t}\, \eta(x_t;\ \mu_{i,t},\ \Sigma_{i,t})$$

Where w is the weight vector, $\eta_{i,t}$ is the Gaussian distribution with an average μ and a covariance matrix Σ.

In order to remove background, we define a threshold discriminates background pixels from foreground pixels.

4.2 Skin Color Detection

In our system, skin color detection plays an essential role. The skin color model proposed in [5] did not show significant results as the authors refer to African-Americans, only. For the purpose of our study, two skin color detection algorithms have been implemented using face color and hand color. The former extracts the color model from the color of user’s face. In particular, we define a region of interest, which is located between the eyes and the nose. This region is calculated by extracting the face, by Viola-Jones algorithm, and on it we select a rectangle situated roughly in the middle of face. The choice of this region is due to the fact that we are sure that will contain skin. Extracted color is represented in the HSV color space (Fig. 5).

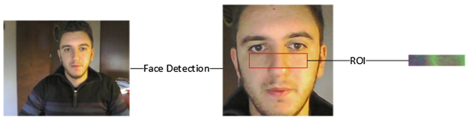

Fig. 5. Skin color detection: approach 1 (Color figure online)

However, we experienced that this method is not accurate, because face color and hand color can be very different in some individuals.

Therefore, we designed a second algorithm that detects skin color from several points of the hand. In order to do so, we provide users with a window showing the video being captured with overlay markers. By asking the user to place their hands over the markers, the algorithm can evaluate the color model of the skin. This process is shown in Fig. 6.

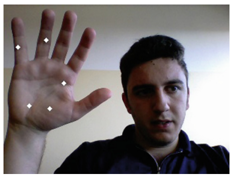

Fig. 6. Skin color detection: approach 2 (Color figure online)

Once the skin color has been extracted, it is then used as a threshold to segment the foreground image, and a second filter is used to delete the face and all regions with an area bigger than the face area and the regions with a too small area. In this way, the remaining regions are very likely to be the hands. The workflow is shown in Fig. 7.

5 Gesture Recognition

Gesture recognition has been implemented using a supervised learning algorithm. Specifically, we employed *Support Vector Machines*.

With the hypothesis of having a linear binary classifier problem, the SVM algorithm finds the separation level that maximizes the margin between two classes, and maximizes the empty area included between them. The amplitude of this value is

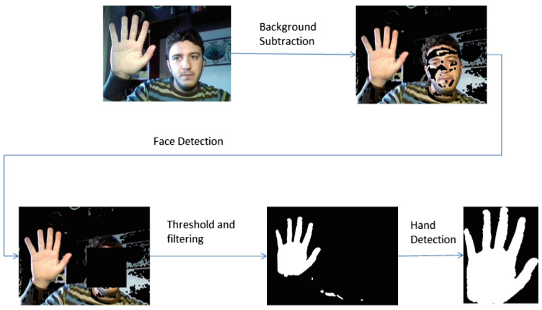

Fig. 7. Hand detection

defined by the distance between the hyperplane that splits the two classes and the samples around that area, i.e., the *support vector* (Fig. 8).

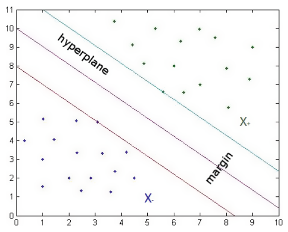

Fig. 8. SVM

In order to train the algorithm, we acquired a training set of 2160 pictures representing all the 26 signs of LIS (80 pictures per sign) (Fig. 9).

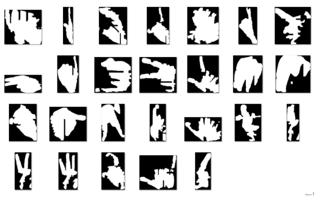

Fig. 9. Dataset

We defined four main classes of signs based on their aspect ratio, in order to improve the recognition performances on letters having similar sign representations. Then, we trained one SVM classifier per group. Particularly, each classifier has been trained using edges extracted from the pictures. All the pictures belonging to the same class have the same dimensions.

- First class: signs {B, R, U, V, Z}, height = 126 px, width = 104 px;
- Second class: signs {A, E, M, N, O, P, Q, T, X}, height = 106 px, width = 141 px;
- Third class: signs {C, D, F, I, L, Y}, height = 179 px, width = 141 px;
- Fourth class: signs {G, H, J, K, S}, height = 159 px, width = 70 px;

The entire image is given as an input to the classifier. As a result, the features vector for each classifier has *h x w* components, where *w* and *h* represent image width and height, respectively.

A different approach is described in [10]: it consists in training a classifier with a pixel coordinate transformation; specifically, Cartesian coordinates are transformed into polar coordinates, i.e. (ρ, ϑ).

Contour pixels are sampled, and only 50 pixels are taken. The coordinates of each pixel are transformed into polar coordinates, where ρ is calculated from the mass center of the picture.

A multilayer perceptron (MLP) was trained with polar coordinates of 50 pixels, leading to a features vector consisting of 100 components.

6 Experimental Results

In this section the results obtained from both the classifiers are presented. The SVM classifier uses a linear kernel, the maximum number of iterations is set to 100 and the error threshold is set to $\varepsilon = 0{,}000001$.

The MLP topology has a single hidden layer. The learning rate is set to 0.3 and the number of epochs is set to 500.

The test set consists of 20 pictures per sign. The following confusion matrices show the results for each sign obtained from both classifiers.

Table 1 reports the confusion matrix relative to class 1. The entry "*Neg*" specifies the negative samples, which don't belong to this class (Tables 2, 3 and 4).

Table 1. Confusion matrix for class 1

	classifier	B	R	U	V	W	Z	Neg
B	SVM	80 %	10 %	0 %	5 %	0 %	5 %	0 %
	MLP	85 %	15 %	0 %	0 %	0 %	0 %	0 %
R	SVM	5 %	80 %	5 %	0 %	0 %	10 %	0 %
	MLP	0 %	80 %	15 %	5 %	0 %	0 %	0 %
U	SVM	0 %	0 %	95 %	0 %	0 %	0 %	5 %
	MLP	5 %	5 %	75 %	10 %	0 %	5 %	0 %
V	SVM	0 %	0 %	0 %	95 %	5 %	0 %	0 %
	MLP	0 %	0 %	5 %	85 %	0 %	0 %	10 %
W	SVM	0 %	5 %	0 %	0 %	95 %	0 %	0 %
	MLP	0 %	0 %	0 %	10 %	80 %	0 %	10 %
Z	SVM	0 %	5 %	0 %	0 %	0 %	95 %	0 %
	MLP	0 %	0 %	5 %	0 %	5 %	90 %	0 %
Neg	SVM	0 %	0 %	0 %	0 %	0 %	0 %	100 %
	MLP	0 %	0.8 %	0 %	0 %	1.7 %	0 %	97.5 %

For the first class the accuracy of the SVM classifier is ca. 95 %, while the accuracy of the MLP classifier is ca. 89 %. For the second class the accuracy of the SVM classifier is ca. 95 %, while the accuracy of the MLP classifier is 87 %. For the third class the accuracy of the SVM classifier is ca. 95 %, and the accuracy of the MLP

Table 2. Confusion matrix for class 2

	classifier	A	E	M	N	O	P	Q	T	X	Neg
A	SVM	95 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	5 %
	MLP	90 %	0 %	0 %	5 %	0 %	0 %	0 %	0 %	0 %	0 %
E	SVM	0 %	100 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
	MLP	0 %	90 %	0 %	0 %	5 %	0 %	0 %	5 %	0 %	0 %
M	SVM	0 %	0 %	100 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
	MLP	0 %	0 %	70 %	5 %	0 %	15 %	10 %	0 %	0 %	0 %
N	SVM	0 %	0 %	10 %	90 %	0 %	0 %	0 %	0 %	0 %	0 %
	MLP	0 %	5 %	10 %	85 %	0 %	0 %	0 %	0 %	0 %	0 %
O	SVM	0 %	0 %	0 %	0 %	100 %	0 %	0 %	0 %	0 %	0 %
	MLP	0 %	5 %	0 %	0 %	70 %	0 %	0 %	5 %	0 %	20 %
P	SVM	0 %	0 %	0 %	0 %	0 %	100 %	0 %	0 %	0 %	0 %
	MLP	5 %	0 %	0 %	5 %	0 %	70 %	5 %	0 %	0 %	15 %
Q	SVM	0 %	0 %	0 %	0 %	0 %	0 %	100 %	0 %	0 %	0 %
	MLP	0 %	0 %	5 %	0 %	0 %	15 %	75 %	0 %	0 %	5 %
T	SVM	0 %	0 %	0 %	0 %	0 %	0 %	0 %	75 %	0 %	25 %
	MLP	0 %	0 %	0 %	0 %	15 %	0 %	0 %	85 %	0 %	0 %
X	SVM	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	90 %	10 %
	MLP	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	90 %	10 %
Neg	SVM	0 %	0 %	0 %	0 %	0.5 %	0 %	0 %	0 %	0 %	99.5 %
	MLP	0.5 %	0 %	0.5 %	0 %	1.7 %	0 %	0 %	1.7 %	1.1 %	94.5 %

Table 3. Confusion matrix for class 3

	classifier	C	D	F	I	L	Y	Neg
C	SVM	90 %	0 %	5 %	0 %	5 %	0 %	0 %
	MLP	80 %	0 %	0 %	0 %	5 %	0 %	15 %
D	SVM	0 %	100 %	0 %	0 %	0 %	0 %	0 %
	MLP	0 %	95 %	5 %	0 %	0 %	0 %	0 %
F	SVM	0 %	20 %	80 %	0 %	0 %	0 %	0 %
	MLP	0 %	0 %	85 %	0 %	0 %	0 %	15 %
I	SVM	0 %	0 %	0 %	100 %	0 %	0 %	0 %
	MLP	0 %	0 %	0 %	95 %	0 %	0 %	5 %
L	SVM	0 %	5 %	10 %	0 %	85 %	0 %	0 %
	MLP	0 %	0 %	0 %	0 %	100 %	0 %	0 %
Y	SVM	0 %	0 %	0 %	0 %	0 %	100 %	0 %
	MLP	0 %	0 %	0 %	15 %	0 %	85 %	0 %
Neg	SVM	0 %	0 %	0 %	0 %	0 %	0 %	100 %
	MLP	0.7 %	0 %	0.7 %	0.7 %	0.7 %	0 %	97.2 %

Table 4. Confusion matrix for class 4

	classifier	G	H	J	K	S	Neg
G	SVM	95 %	5 %	0 %	0 %	0 %	0 %
	MLP	100 %	0 %	0 %	0 %	0 %	0 %
H	SVM	0 %	100 %	0 %	0 %	0 %	0 %
	MLP	10 %	70 %	0 %	0 %	10 %	10 %
J	SVM	0 %	0 %	100 %	0 %	0 %	0 %
	MLP	0 %	0 %	95 %	5 %	0 %	0 %
K	SVM	0 %	0 %	0 %	100 %	0 %	0 %
	MLP	0 %	5 %	0 %	85 %	0 %	10 %
S	SVM	5 %	0 %	0 %	0 %	95 %	0 %
	MLP	0 %	0 %	0 %	0 %	85 %	15 %
Neg	SVM	0 %	0 %	0 %	0 %	0 %	100 %
	MLP	2.1 %	1 %	1 %	0 %	1 %	94.9 %

classifier is ca. 93 %. For the fourth class, the accuracy of the SVM classifier accuracy is ca. 97 %, while the accuracy of the MLP classifier is ca. 90 %.

The signs *B, C, F, Q, R, S,* and *T* show unreliable results, indeed their accuracy is less than 80 %. Figure 10 shows a screenshot of the software implemented in the experimental study; specifically, the *L* sign is being recognized by our system.

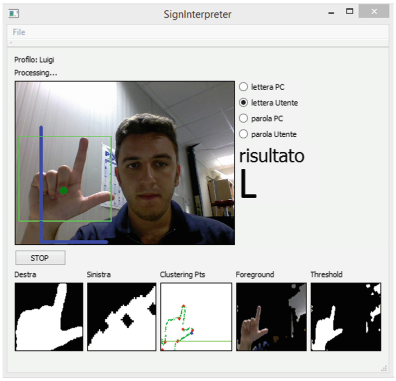

Fig. 10. Test case

7 Conclusion and Future Works

We proposed two methods, the first one trains an SVM classifier with the hand contours, and the second one uses the polar coordinates to train a NN. Even if both the approaches offer good results, which are much higher than 50 % (random classifier), we can conclude that the first approach performs better.

This work is limited to the recognition of the Italian finger spelling only. The segmentation based on the skin color is the hardest problem faced during this study, because it is required that the system is able to work in uncontrolled environments (users within any scene) but can be improved by using in future works the algorithm developed from authors in [12]. SVM and MLP performance could be improved by using several techniques shown in [13] for SVM performance evaluation, or in [14] for MLP pre-processing or in [15] for MLP topology optimization.

A future work could see the system equipped with a most suitable sensor able to recognize hands and fingers, as the Leap Motion, which is a technology equipped with infrared sensor. With the combination of Leap Motion and a RGB sensor, the system could be extended to identify more signs, without having to renounce to portability.

References

1. Alvarez, S., Llorca, D.F.: Spatial Hand Segmentation Using Skin Colour and Background Subtraction, Robesafe Research Group, Department of Automatics, Universidad de Alcala, Madrid, Spain
2. Comparative Study of Statistical Skin Detection Algorithms for Sub-Continental Human Images, Institute of Information Technology, Department of Statistics, Biostatistics and Informatics, University of Dhaka, Bangladesh
3. KaewTraKulPong, P., Bowden, R.: An Improved Adaptive Background Mixture Model for Real-Time Tracking with Shadow Detection, Vision and Virtual Reality group, Department of Systems Engineering, Brunel University
4. Piccardi, M.: Background subtraction techniques: a review, Computer Vision Group, Faculty of Information Technology, University of Technology, Sydney (UTS), Australia
5. Phung, S.L., Bouzerdoum, A., Chai, D.: Skin segmentation using color pixel classification: analysis and comparison. IEEE Trans. Pattern Anal. Mach. Intell. **27**, 148–154 (2005)
6. Gebre, B.G., Wittenburg, P., Heskes, T.: Automatic Sign Language Identification, Max Planck Institute for Psycholinguistics, Nijmegen, Radboud University, Nijmegen
7. Sign Language in the Intelligent Sensory, Budapest University of Technology and Economics, Department of Automation and Applied Informatics, Budapest, Hungary, Department of Mechatronics, Optics and Instrumentation Technology
8. Sonka, M., Hlavac, V., Boyle, R.: Image Processing, Analysis, and Machine Vision. Thomson Learning, Toronto (2008)
9. Otiniano-Rodriguez, K.C., Càmara-Chavez, G., Menotti, D.: Hu and Zernike Moments for Sign Language Recognition, Computing Department, Federal University of Ouro Preto, Brazil
10. Yang, M., Kpalma, K., Ronsin, J.: A Survey of Shape Feature Extraction Techniques, IETR-INSA, UMR-CNRS 6164, 35043 Rennes, Shandong University, 250100, Jinan, France, China
11. Stokoe, W.C.: Sign language structure: an outline of the visual communication systems of the American deaf
12. Bevilacqua, V., Filograno, G., Mastronardi, G.: Face detection by means of skin detection. In: Huang, D.-S., Wunsch II, D.C., Levine, D.S., Jo, K.-H. (eds.) ICIC 2008. LNCS (LNAI), vol. 5227, pp. 1210–1220. Springer, Heidelberg (2008)

13. Bevilacqua, V., Pannarale, P., Abbrescia, M., Cava, C., Paradiso, A., Tommasi, S.: Comparison of data-merging methods with SVM attribute selection and classification in breast cancer gene expression. BMC Bioinform. **13**(Suppl. 7), S9 (2012)
14. Bevilacqua, V.: Three-dimensional virtual colonoscopy for automatic polyps detection by artificial neural network approach: new tests on an enlarged cohort of polyps. Neurocomputing (2013). doi:10.1016/j.neucom.2012.03.026. ISSN: 0925-2312 (2012)
15. Bevilacqua, V., Mastronardi, G., Menolascina, F., Pannarale, P., Pedone, A.: A novel multi-objective genetic algorithm approach to artificial neural network topology optimisation: the breast cancer classification problem. In: International Joint Conference on Neural Networks, IJCNN 2006, pp. 1958–1965 (2006)