Real-Time Hand Gesture Recognition Using a Color Glove

Luigi Lamberti[1] and Francesco Camastra[2,⋆]

[1] Istituto Tecnico Industriale "Enrico Medi",
via Buongiovanni 84, 80046 San Giorgio a Cremano, Italy
luigi.lamberti@istruzione.it
[2] Department of Applied Science, University of Naples Parthenope,
Centro Direzionale Isola C4, 80143 Naples, Italy
francesco.camastra@uniparthenope.it

Abstract. This paper presents a real-time hand gesture recognizer based on a color glove. The recognizer is formed by three modules. The first module, fed by the frame acquired by a webcam, identifies the hand image in the scene. The second module, a feature extractor, represents the image by a nine-dimensional feature vector. The third module, the classifier, is performed by means of *Learning Vector Quantization*. The recognizer, tested on a dataset of 907 hand gestures, has shown very high recognition rate.

1 Introduction

Gesture is one of the means that humans use to send informations. According to Kendon's gesture continuum [1] the information amount conveyed by gesture increases when the information quantity sent by the human voice decreases. Moreover, the hand gesture for some people, e.g. the disabled people, is one of main means, sometimes the most relevant, to send information.

The aim of this work is the development of a real-time hand gesture recognizer that can also run on devices that have moderate computational resources, e.g. netbooks. The real-time requirement is motivated by associating to the gesture recognition the carrying out of an action, for instance the opening of a multimedia presentation, the starting of an internet browser and other similar actions. The latter requirement, namely the recognizer can run on a netbook, is desirable in order that the system can be used extensively in classrooms of schools as tool for teachers or disabled students.

The paper presents a real-time hand gesture recognition system based on a color glove that can work on a netbook. The system is formed by three modules. The first module, fed by the frame acquired by a webcam, identifies the hand image in the scene. The second module, a feature extractor, represents the image by means of a nine-dimensional feature vector. The third module, the classifier, is performed by *Learning Vector Quantization*.

⋆ Corresponding author.

The paper is organized as follows: Section 2 describes the approach used; Section 3 gives an account of the segmentation module; the feature extraction process is discussed in Section 4; a review of Learning Vector Quantization is provided in Section 5; Section 6 reports some experimental results; in Section 7 some conclusions are drawn.

2 The Approach

Several approaches were proposed for gesture recognition [2]. Our approach was inspired by Virtual Reality [3] applications where the movements of the hands of people are tracked asking them to wear data gloves [4]. A data glove is a particular glove that has fiber-optic sensors inside that allow the track the movement of the fingers of hand. Our approach is similar to the Virtual Reality's one. We ask the person, whose gesture has to be recognized, to wear a glove or more precisely, a color glove. A color glove was recently used by Wang and Popovic [5] for the real-time hand tracking. Their color glove was formed by patches of several different colors[1]. In our system we use a wool glove[2] where three different colors are used for the parts of the glove corresponding to the palm and the fingers, whereas the rest of glove is black. One color is used to dye the palm, the remaining two to color differently adjacent fingers, as shown in figure 1. We

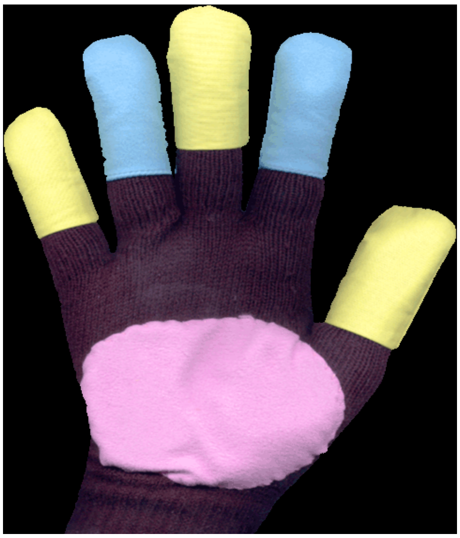

Fig. 1. The Color Glove used in our approach

[1] In the figure reported in [5] the glove seems to have at least seven different colors.
[2] Wool is not obviously compulsory, clearly cotton or other fabrics can be used.

have chosen to color the palm by magenta and the fingers by cyan and yellow. Further investigations seem to show that the abovementioned choice does not affect remarkably the performances of the recognizer.

Finally, we conclude the section with a cost analysis. In terms of costs, our glove compares favourably with data gloves. Our glove costs some euro, whereas the cost of effective data gloves can exceed several hundreds euro.

3 Segmentation Module

The gesture recognizer has three modules. The first one is the *segmentation* module. The module receives as input the RGB color frame acquired by the webcam and performs the segmentation process identifying the hand image. The segmentation process can be divided in four steps. The first step consists in representing the frame in *Hue-Saturation-Intensity* (*HSI*) color space [6]. We tested experimentally several color spaces, i.e. RGB, HSI, CIE XYZ, L*ab, L*uv and some others [7]. We chose HSI since it was the most suitable color space to be used in our segmentation process. Several algorithms were proposed [8] to segment color images. Our choice was to use the least expensive computationally segmentation strategy, i.e. a thresholding-based method. During the second step, the pixels of the image are divided in seven categories: "Cyan Pixels" (C), "Probable Cyan Pixels" (PC), "Yellow Pixels" (Y), "Probable Yellow Pixels" (PY), "Magenta Pixels" (M), "Probable Magenta Pixels" (PM), "Black Pixels" (B). A pixel, represented as a triple P=(H,S,I), is classified as follows:

$$\left\{\begin{array}{ll} P \in C & \text{if } H \in [\Theta_1, \Theta_2] \wedge S > \Theta_3 \wedge I > \Theta_4 \\ P \in PC & \text{if } H \in [\Theta_{1r}, \Theta_{2r}] \wedge S > \Theta_{3r} \wedge I > \Theta_{4r} \\ P \in Y & \text{if } H \in [\Theta_5, \Theta_6] \wedge S > \Theta_7 \wedge I > \Theta_8 \\ P \in PY & \text{if } H \in [\Theta_{5r}, \Theta_{6r}] \wedge S > \Theta_{7r} \wedge I > \Theta_{8r} \\ P \in M & \text{if } H \in [\Theta_9, \Theta_{10}] \wedge S > \Theta_{11} \wedge I > \Theta_{12} \\ P \in PM & \text{if } H \in [\Theta_{9r}, \Theta_{10r}] \wedge S > \Theta_{11r} \wedge I > \Theta_{12r} \\ P \in B & \text{otherwise} \end{array}\right\}, \qquad (1)$$

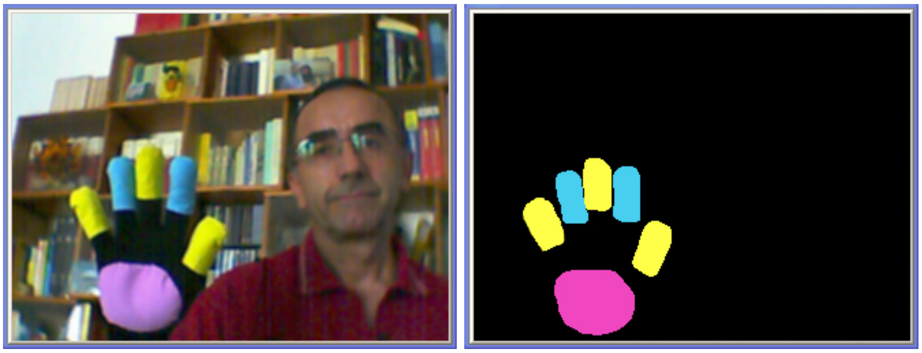
(a) (b)

Fig. 2. The original image (a). The image after the segmentation process (b).

where Θ_{ir} is a relaxed value of the respective threshold Θ_i and $\Theta_i, (i = 1, \dots, 12)$ are thresholds that were set up in a proper way.

In the third step only the pixels belonging to PC, PY and PM categories are considered. Given a pixel P and denoting with $N(P)$ its neighborhood, using the 8-connectivity [6], the following rules are applied:

$$\left\{\begin{array}{l} \textsf{If } P \in PC \wedge \exists_{Q \in N(P)} Q \in C \textsf{ then } P \in C \textsf{ else } P \in B \\ \textsf{If } P \in PY \wedge \exists_{Q \in N(P)} Q \in Y \textsf{ then } P \in Y \textsf{ else } P \in B \\ \textsf{If } P \in PM \wedge \exists_{Q \in N(P)} Q \in M \textsf{ then } P \in M \textsf{ else } P \in B \end{array}\right\}, \qquad (2)$$

In a nutshell, pixels belonging to PC, PY and PM categories are upgraded respectively to C,Y and M, respectively if in their neighborhood exists at least one pixel belonging to the respective superior class. The remaining pixels are degraded to black pixels. At the end of this phase only four categories, i.e. C, Y, M, and B, remain.

In the last step the connected components for the color pixels, i.e. the one belonging to the Cyan, Yellow and Magenta categories, are computed. Finally, each connected component is undergone to a *morphological opening* followed by a *morphological closure* [6]. In both steps the structuring element is a circle of radius of three pixels.

4 Feature Extraction

After the segmentation process, the image of the hand is represented by a vector of nine numerical features. The feature extraction process has the following steps. The first step consists in individuating the region formed by magenta pixels, that corrisponds to the palm of the hand. Then it is computed the centroid and the major axis of the region. In the second step the five centroids of yellow and cyan regions, corresponding to the fingers are individuated. Then, for each of the five regions, the angle $\theta_i (i = 1, \dots, 5)$ between the main axe of the palm and the line connecting the centroids of the palm and the finger, is computed (see figure 3). In the last step the hand image is represented by a vector of nine normalized numerical features. As shown in figure 3, the feature vector is formed by nine numerical values that represent five distances $d_i (i = 1, \dots, 5)$ and four angles $\beta_i (i = 1, \dots, 4)$, respectively. Each distance measures the Euclidean distance between the centroid of the palm and the respective finger. The four angles between the fingers are easily computed by subtraction having computed before the angles θ_i. The extracted features are invariant, by construction, w.r.t. rotation and translation in the plane of the image.

Finally, all features are normalized. The distances are normalized, dividing them by the maximum value that they can assume [9]. The angles are normalized, dividing them by $\frac{\pi}{2}$ radians, assuming that it is the maximum angle that can be measured by the fingers.

In the description above we have implicitly supposed that exists only one region for the palm and five fingers. If the regions for the palm and the finger are not unique, the system uses a different strategy depending on it is already

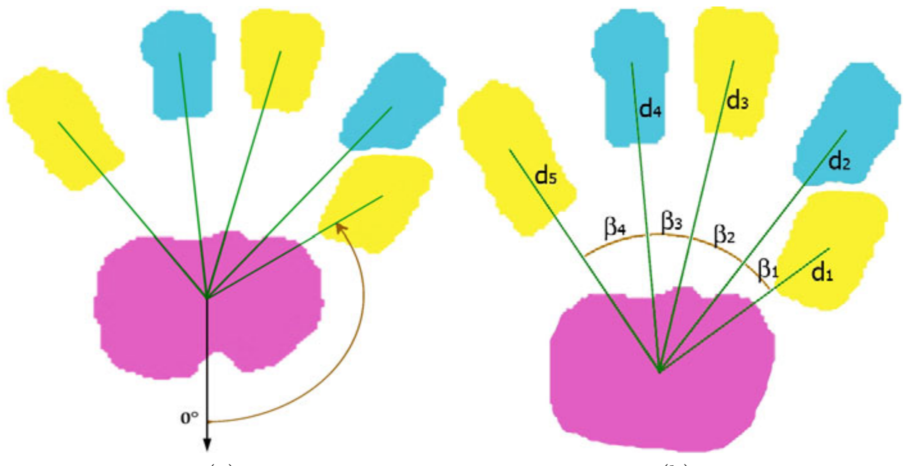

Fig. 3. (a) The angles Θ between the main axe of the palm and the line connecting the centroids of the palm and each finger, are computed. (b) The feature vector is formed by five distances $d_i (i = 1, \ldots, 5)$ and four angles $\beta_i (i = 1, \ldots, 4)$, obtained by subtraction, from angles Θ.

trained. If the system is not trained yet, i.e. it is in training the system takes for the palm and for each finger the largest region, in terms of area. If the system is trained it selects for each finger up to the top three largest regions, if they exist; whereas for the palm up the top two largest regions are picked. The chosen regions are combined in all possible ways yielding different possible hypotheses for the hand. Finally, the system selects the hypothesis whose feature vector is evaluated with the highest score by the classifier.

5 Learning Vector Quantization

We use *Learning Vector Quantization* (*LVQ*) as classifier since LVQ requires moderate computational resources compared with other machine learning classifiers (e.g. SVM [10]). We pass to describe LVQ. We first fix the notation; let $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{\ell}$ be a data set with $\boldsymbol{x}_i \in \mathbb{R}^N$. We call *codebook* the set $W = \{w_k\}_{k=1}^{K}$ with $w_k \in \mathbb{R}^N$ and $K \ll \ell$. Vector quantization aims to yield codebooks that represent as much as possible the input data D. LVQ is a supervised version of vector quantization and generates codebook vectors (*codevectors*) to produce *near-optimal decision boundaries* [11]. LVQ consists of the application of a few different learning techniques, namely LVQ1, LVQ2 and LVQ3. LVQ1 uses for classification the nearest-neighbour decision rule; it chooses the class of the nearest codebook vector. LVQ1 learning is performed in the following way: if $\boldsymbol{m}_t^c$ [3] is the nearest codevector to the input vector $\boldsymbol{x}$, then

[3] $\boldsymbol{m}_t^c$ stands for the value of $\boldsymbol{m}^c$ at time t.

$$\begin{array}{ll} \boldsymbol{m}^c_{t+1} = \boldsymbol{m}^c_t + \alpha_t[\boldsymbol{x} - \boldsymbol{m}^c_t] & \text{if } \boldsymbol{x} \text{ is classified correctly} \\ \boldsymbol{m}^c_{t+1} = \boldsymbol{m}^c_t - \alpha_t[\boldsymbol{x} - \boldsymbol{m}^c_t] & \text{if } \boldsymbol{x} \text{ is classified incorrectly} \\ \boldsymbol{m}^i_{t+1} = \boldsymbol{m}^i_t & i \neq c \end{array} \quad (3)$$

where α_t is the learning rate at time t.

In our experiments, we used a particular version of LVQ1, that is *Optimized Learning Vector Quantization (OLVQ1)* [11], a version of the model that provides a different learning rate for each codebook vector. Since LVQ1 tends to push codevectors away from the decision surfaces of the *Bayes rule* [12], it is necessary to apply to the codebook generated a successive learning technique called LVQ2. LVQ2 tries harder to approximate the Bayes rule by pairwise adjustments of codevectors belonging to adjacent classes. If $\boldsymbol{m}^s$ and $\boldsymbol{m}^p$ are nearest neighbours of different classes and the input vector $\boldsymbol{x}$, belonging to the $\boldsymbol{m}^s$ class, is closer to $\boldsymbol{m}^p$ and falls into a zone of values called *window* [4], the following rule is applied:

$$\begin{array}{l} \boldsymbol{m}^s_{t+1} = \boldsymbol{m}^s_t + \alpha_t[\boldsymbol{x} - \boldsymbol{m}^s_t] \\ \boldsymbol{m}^p_{t+1} = \boldsymbol{m}^p_t - \alpha_t[\boldsymbol{x} - \boldsymbol{m}^p_t] \end{array} \quad (4)$$

It can be shown [13] that the LVQ2 rule produces an instable dynamics. To prevent this behavior as far as possible, the window w within the adaptation rule takes place must be chosen carefully.

In order to overcome the LVQ2 stability problems, Kohonen proposed a further algorithm (LVQ3). If $\boldsymbol{m}^i$ and $\boldsymbol{m}^j$ are the two closest codevectors to input $\boldsymbol{x}$ and $\boldsymbol{x}$ falls in the window, the following rule is applied[5]:

$$\left\{\begin{array}{ll} \boldsymbol{m}^i_{t+1} = \boldsymbol{m}^i_t & \text{if } C(\boldsymbol{m}^i) \neq C(\boldsymbol{x}) \wedge C(\boldsymbol{m}^j) \neq C(\boldsymbol{x}) \\ \boldsymbol{m}^j_{t+1} = \boldsymbol{m}^j_t & \text{if } C(\boldsymbol{m}^i) \neq C(\boldsymbol{x}) \wedge C(\boldsymbol{m}^j) \neq C(\boldsymbol{x}) \\ \boldsymbol{m}^i_{t+1} = \boldsymbol{m}^i_t - \alpha_t[\boldsymbol{x}_t - \boldsymbol{m}^i_t] & \text{if } C(\boldsymbol{m}^i) \neq C(\boldsymbol{x}) \wedge C(\boldsymbol{m}^j) = C(\boldsymbol{x}) \\ \boldsymbol{m}^j_{t+1} = \boldsymbol{m}^j_t + \alpha_t[\boldsymbol{x}_t - \boldsymbol{m}^j_t] & \text{if } C(\boldsymbol{m}^i) \neq C(\boldsymbol{x}) \wedge C(\boldsymbol{m}^j) = C(\boldsymbol{x}) \\ \boldsymbol{m}^i_{t+1} = \boldsymbol{m}^i_t + \alpha_t[\boldsymbol{x}_t - \boldsymbol{m}^i_t] & \text{if } C(\boldsymbol{m}^i) = C(\boldsymbol{x}) \wedge C(\boldsymbol{m}^j) \neq C(\boldsymbol{x}) \\ \boldsymbol{m}^j_{t+1} = \boldsymbol{m}^j_t - \alpha_t[\boldsymbol{x}_t - \boldsymbol{m}^j_t] & \text{if } C(\boldsymbol{m}^i) = C(\boldsymbol{x}) \wedge C(\boldsymbol{m}^j) \neq C(\boldsymbol{x}) \\ \boldsymbol{m}^i_{t+1} = \boldsymbol{m}^i_t + \epsilon\alpha_t[\boldsymbol{x}_t - \boldsymbol{m}^i_t] & \text{if } C(\boldsymbol{m}^i) = C(\boldsymbol{m}^j) = C(\boldsymbol{x}) \\ \boldsymbol{m}^j_{t+1} = \boldsymbol{m}^j_t + \epsilon\alpha_t[\boldsymbol{x}_t - \boldsymbol{m}^j_t] & \text{if } C(\boldsymbol{m}^i) = C(\boldsymbol{m}^j) = C(\boldsymbol{x}) \end{array}\right\}, \quad (5)$$

where $\epsilon \in [0, 1]$ is a fixed parameter.

6 Experimental Result

To validate the recognizer we selected 13 gestures, invariant by rotation and translation. We associated to each gesture a symbol, a letter or a digit, as shown in figure 4. We collected a database of 1541 gestures, performed by people of different gender and physique. The database was splitted with a random process into training and test set containing respectively 634 and 907 gesture. The number of classes used in the experiments was 13, namely the number of the different gestures in our database. In our experiments the three learning techniques, i.e.

[4] The window is defined around the midplane of $\boldsymbol{m}^s$ and $\boldsymbol{m}^p$.

[5] $C(\boldsymbol{q})$ stands for the class of $\boldsymbol{q}$.

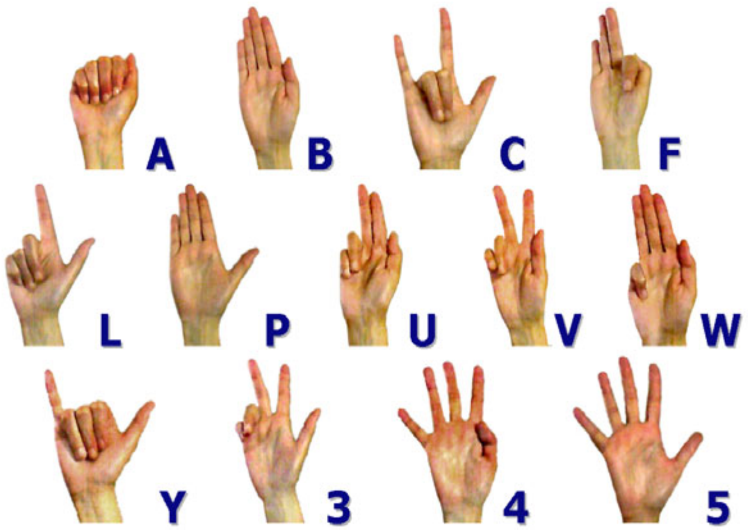

Fig. 4. Gestures represented in the database

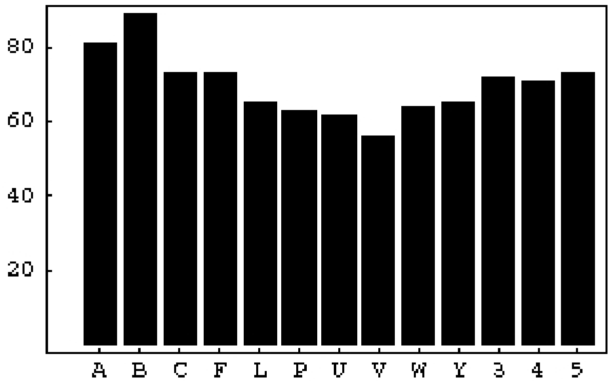

Fig. 5. Gesture distribution in the test set

LVQ1, LVQ2 and LVQ3, were applied. We trained several LVQ nets by specifying different combinations of learning parameters, i.e. different learning rates for LVQ1, LVQ2, LVQ3 and various total number of codevectors. The best LVQ net was selected by means of *crossvalidation* [14]. LVQ trials were performed using *LVQ-pak* [15] software package. Figure 5 shows the gesture distribution in the test set. In Table 1, for different classifiers, the performances on the test set, measured in terms of recognition rate in absence of rejection, are reported.

Our best result in terms of recognition rate is *97.79 %*. The recognition rates for each class are shown in figure 6.

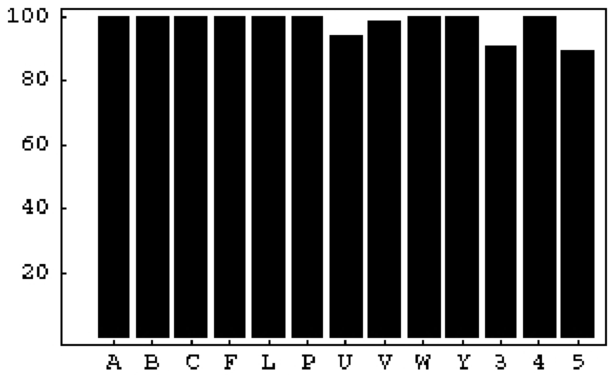

Fig. 6. Recognition rates, for each class, in the test set

Table 1. Recognition rates on the test set, in absence of rejection, for several LVQ classifiers

Algorithm	Correct Classification Rate
knn	85.67 %
LVQ1	96.36 %
LVQ1 + LVQ2	97.57 %
LVQ1 + LVQ3	**97.79** %

The system, implemented in C++ under Windows XP Microsoft and .NET Framework 3.5 on a Netbook with 32bit "Atom N280" 1.66 Ghz, Front Side Bus a 667 Mhz and 1 GB di RAM, requires 140 CPU msec to recognize a single gesture.

Finally, a version of the gesture recognition system where the recognition of a given gesture is associated the carrying out of a Powerpoint[6] command is actually in use at Istituto Tecnico Industriale "Enrico Medi" helping the first author in his teaching activity.

7 Conclusions

In this paper we have described real-time hand gesture recognition system based a color Glove. The system is formed by three modules. The first module identifies the hand image in the scene. The second module performs the feature extraction and represents the image by a nine-dimensional feature vector. The third module, the classifier, is performed by means of Learning Vector Quantization. The recognition system, tested on a dataset of 907 hand gestures, has shown a

[6] Powerpoint, .NET and Windows XP are registered trademarks by Microsoft Corp.

recognition rate close to 98%. The system implemented on a netbook requires an average time of 140 CPU msec to recognize a hand gesture. A version of the system where the recognition of a given gesture is associated the carrying out of a multimedia presentation command is actually used by the first author in his teaching duties. Since our system compares favourably, in terms of costs, with data glove, in the next future we plan to investigate its usage in virtual reality applications.

Acknowledgments. First the author wish to thank the anonymous reviewer for his valuable comments. The work was developed by L. Lamberti, under the supervision of F. Camastra, as final dissertation project for B. Sc. in Computer Science at University of Naples Parthenope.

References

1. Kendon, A.: How gestures can become like words. In: Crosscultural Perspectives in Nonverbal Communication, Toronto, Hogrefe, pp. 131–141 (1988)
2. Mitra, S., Acharya, T.: Gesture recognition: A survey. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 37(3), 311–324 (2007)
3. Burdea, G.C., Coiffet, P.: Virtual Reality Technology. John-Wiley & Sons, New York (2003)
4. Dipietro, L., Sabatini, A.M., Dario, P.: A survey of glove-based systems and their applications. IEEE Transactions on Systems, Man and Cybernetics 38(4), 461–482 (2008)
5. Wang, R.Y., Popovic, J.: Real-time hand-tracking with a color glove. ACM Transactions on Graphics 28(3), 461–482 (2009)
6. Gonzales, R.C., Woods, R.E.: Digital Image Processing. Prentice-Hall, Upper Saddle River (2002)
7. DelBimbo, A.: Visual Information Processing. Morgan Kaufmann Publishers, San Francisco (1999)
8. Cheng, H.D., Jiang, X.H., Sun, Y., Wang, J.: Color image segmentation: advances and prospects. Pattern Recognition 34(12), 2259–2281 (2001)
9. Lamberti, L.: Handy: Riconoscimento di semplici gesti mediante webcam. B. Sc. Dissertation (in Italian), University of Naples "Parthenope" (2010)
10. Shawe-Taylor, J., Cristianini, N.: Kernels Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
11. Kohonen, T.: Self-Organizing Maps. Springer, Berlin (1997)
12. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John-Wiley & Sons, New York (2001)
13. Kohonen, T.: Learning vector quantization. In: The Handbook of Brain Theory and Neural Networks, pp. 537–540. MIT Press, Cambridge (1995)
14. Stone, M.: Cross-validatory choice and assessment of statistical prediction. Journal of the Royal Statistical Society 36(1), 111–147 (1974)
15. Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J., Torkkola, K.: Lvq-pak: The learning vector quantization program package. Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science (1996)