Handy: A Real-Time Three Color Glove-Based Gesture Recognizer with Learning Vector Quantization

Luigi Lamberti[a], Francesco Camastra[b,*]

[a]*Istituto Tecnico Industriale "Enrico Medi", via Buongiovanni 84, 80046 San Giorgio a Cremano, Italy*
[b]*Department of Applied Science, University of Naples Parthenope, Centro Direzionale Isola C4, 80143 Naples, Italy*

Abstract

This paper presents Handy, a real-time hand gesture recognizer based on a three 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.

Keywords: Gesture Recognition; Real Time; Color Glove; HSI Color Space; Learning Vector Quantization; Data Glove

1. Introduction

Gesture is one of the means that humans use to send informations. In general, the information amount conveyed by gesture increases when the information quantity sent by the human voice decreases (Kendon, 1988). Moreover, the hand gesture for some people, e.g., the disabled people, is one among main means for

*Corresponding author
Email addresses: luigi.lamberti@istruzione.it (Luigi Lamberti), camastra@ieee.org (Francesco Camastra)

sending 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 performing 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 environments, e.g., school classrooms, that usually have computers with moderate computational resources.
The paper presents *Handy*, a real-time hand gesture recognition system based on a three color glove that can work on a device with moderate computational resource as a netbook. Handy is composed of 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*.
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 (Mitra and Acharya, 2007; Frantti and Kallio, 2004; Huang et al., 2011; Tsai and Lee, 2011; Chaudhary et al., 2011). Our approach was inspired by Virtual Reality applications (Burdea and Coiffet, 2003) where the movements of the hands of people are tracked asking

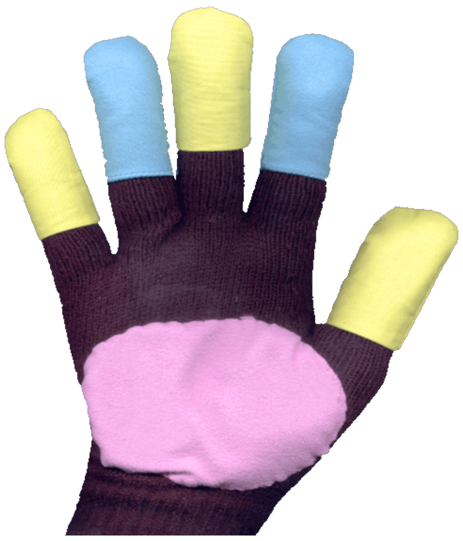


Figure 1: The Color Glove used in our approach.

them to wear data gloves (Dipietro et al., 2008). A data glove is a particular glove that has sensors, typically magnetic or in optic fiber, inside that allow to 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 *three-color glove*. A color glove was recently used by Wang and Popovic for the real-time hand tracking (Wang and Popovic, 2009). Their color glove was formed by patches of several different colors. In the figures of their manuscript the glove seems to be composed of at least seven different colors.

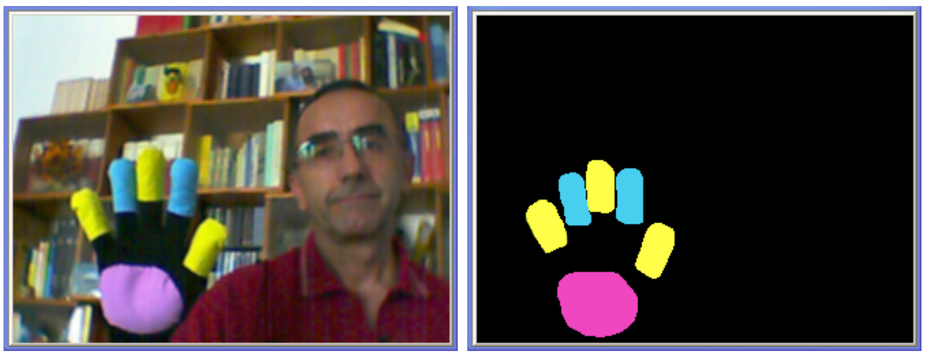
In our system we use a wool[1] three-color glove where three different colors are used for the parts of the glove corresponding to the palm and the fingers, whereas the rest of glove is black. One color is used to dye the palm, the remaining two to color differently adjacent fingers, as shown in Fig. 1. We have chosen to color the palm by magenta and the fingers by cyan and yellow. Further investigations

[1]wool is not compulsory, other fabrics may be used.

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 any euro, whereas the cost of effective data gloves can exceed several hundreds euro.

3. Segmentation Module

Handy has three modules. The first one is the *segmentation* module. The module receives as input the RGB color frame acquired by the webcam of the netbook and performs the segmentation process identifying the hand image. The segmentation process can be divided in five steps. In the first step the original frame is reduced to a RGB Color image of 320x240 pixels with the aim of speeding up the whole recognition process. Then, the image is represented in *Hue-Saturation-Intensity* (*HSI*) color space (Gonzales and Woods, 2002). We tried several color spaces, i.e., RGB, HSI, CIE XYZ, L*a*b*, L*u*v* and some others (DelBimbo, 1999). We chose HSI since in our experiments it was the most suitable color space to be used in the segmentation process. Several algorithms were proposed (Cheng et al., 2001) 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), "Likely Cyan Pixels" (LC), "Yellow Pixels" (Y), "Likely Yellow Pixels" (LY), "Magenta Pixels" (M), "Likely Magenta Pixels" (LM), "Black Pixels" (B). A pixel, represented by means of a triple



(a) (b)

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

$P = (H, S, I)$, is categorized as follows:

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

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

In the third step only the pixels belonging to LC, LY and LM categories are considered. Given a pixel P and denoting with $N(P)$ its neighborhood, using the

8-connectivity (Gonzales and Woods, 2002), the following rules are applied:

$$\left\{\begin{array}{l} \mathsf{If}\ P \in LC \wedge \bigvee_{Q \in N(P)} Q \in C \quad \mathsf{then}\ P \in C \quad \mathsf{else}\ P \in B \\ \mathsf{If}\ P \in LY \wedge \bigvee_{Q \in N(P)} Q \in Y \quad \mathsf{then}\ P \in Y \quad \mathsf{else}\ P \in B \\ \mathsf{If}\ P \in LM \wedge \bigvee_{Q \in N(P)} Q \in M \quad \mathsf{then}\ P \in M \quad \mathsf{else}\ P \in B \end{array}\right\}, \tag{2}$$

where $\bigvee_{i=1}^{\ell} t_i$ stands for $t_1 \vee t_2 \vee \ldots \vee t_\ell$.

In other words, pixels belonging to LC, LY and LM categories are upgraded to C, Y and M, respectively if in their neighborhood exists at least one pixel belonging to the respective superior category. 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 (see Fig. 2).

In the fourth step the connected components for the color pixels, i.e., the ones belonging to the Cyan, Yellow and Magenta categories, are computed, using the *OpenCV* library (Bradski and Kaehler, 2008). Finally, in the last step each connected component is undergone to a *morphological opening* followed by a *morphological closure* (Gonzales and Woods, 2002) using as structuring element a circle of radius of three pixels and area of 37 pixels (see Fig. 3).

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 corresponds 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 $\theta_i (i = 1, \ldots, 5)$ between the major axis of the palm

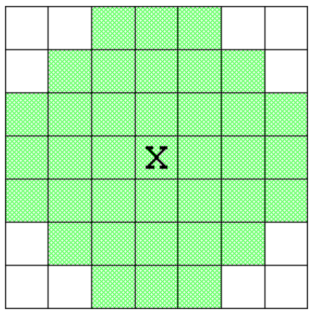


Figure 3: The structuring element, in green, used for obtaining morphological opening and closure. The cross indicates the center of the element.

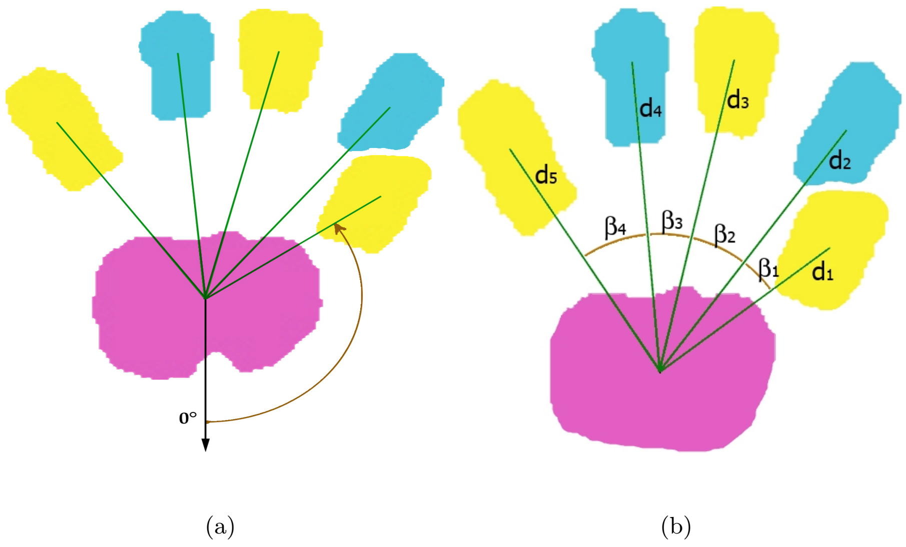


Figure 4: (a) The angles θ_i between the major axis of the palm and the line connecting the centroids of the palm and each finger, are computed. (b) The feature vector is formed by five distances $d_i (i = 1, \ldots, 5)$ and four angles $\beta_i (i = 1, \ldots, 4)$, obtained by subtraction, from angles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

and the line connecting the centroids of the palm and the finger, is computed (see Fig. 4). In the last step the hand image is represented by a vector of nine normalized numerical features. As shown in Fig. 4, the feature vector is formed by nine numerical values that represent five distances $d_i(i = 1, \ldots, 5)$ and four angles $\beta_i(i = 1, \ldots, 4)$, 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 rotation and translation in the plane of the camera.

Finally, all features are normalized. The distances are normalized, dividing them by the maximum value that they can assume (Lamberti, 2010). 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. As a general comment, Handy feature extraction process described above, compares favourably, in terms of minor computational complexity, with the one employed in the Real-Time Hand Gesture Recognizer by Ren and Gu (2010), based on the computation of *Normalized Moment of Inertia* (DelBimbo, 1999) and *Hu invariant moments* (Hu, 1962).

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 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. The Classifier

In recent years *Support Vector Machine* (*SVM*) (Cortes and Vapnik, 1995; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004) has been one of the most effective classification algorithm. Since SVM is a binary classifier, if we want to use SVM when the number of classes K is larger than two it is necessary to use specific strategies. The simplest strategy, *one-versus-all method* (Herbrich, 2004), requires an ensemble of K SVM classifiers, i.e., one classifier for each class against all the other classes. If we use other strategies the number of classifiers increases to K(K-1)/2. Therefore, SVMs require computational resources that are not compatible with a real-time recognizer.

5.1. Learning Vector Quantization

Having said that, we have chosen *Learning Vector Quantization* (*LVQ*) (Kohonen, 1995) as classifier in the gesture recognizer since it requires moderate computational resources. In last years LVQ was successfully applied in many different fields such as handwritten digit (Ho, 1993) and cursive character recognition (Camastra and Vinciarelli, 2001), the identification of the Head-and-Shoulders patterns in the financial time series (Zapranis and Tsinaslanidis, 2010), the analysis of sensor array data (Ciosek and Wróblewski, 2006) and automatic recognition of whale calls for real-time monitoring (Mouy et al., 2009).
We pass to describe LVQ and first fix the notation. Let $\mathcal{D} = \{\vec{x}_i\}_{i=1}^{\ell}$ be a data set with $\vec{x}_i \in \mathbb{R}^N$. 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* (Kohonen, 1997). 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. We denote with $\vec{m}_t^c$ the value of $\vec{m}^c$ at time t and with $\mathcal{C}(\vec{x})$ and $\mathcal{C}(\vec{m}_t^c)$ the class of $\vec{x}$ and $\vec{m}_t^c$, respectively. If $\vec{m}_t^c$ is the nearest codevector to the input vector $\vec{x}$, then

$$\begin{aligned}
\vec{m}_{t+1}^c &= \vec{m}_t^c + \alpha_t[\vec{x} - \vec{m}_t^c] && \text{if } \mathcal{C}(\vec{x}) = \mathcal{C}(\vec{m}_t^c) \\
\vec{m}_{t+1}^c &= \vec{m}_t^c - \alpha_t[\vec{x} - \vec{m}_t^c] && \text{if } \mathcal{C}(\vec{x}) \neq \mathcal{C}(\vec{m}_t^c) \\
\vec{m}_{t+1}^i &= \vec{m}_t^i && i \neq c
\end{aligned} \tag{3}$$

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)* (Kohonen, 1997), 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* (Duda et al., 2001), 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 $\vec{m}^s$ and $\vec{m}^p$ are nearest neighbours of different classes and the input vector $\vec{x}$, belonging to the $\vec{m}^s$ class, is closer to $\vec{m}^p$ and falls into a zone of values called *window* (that is defined around the midplane of $\vec{m}^s$ and $\vec{m}^p$), the following rule is applied:

$$\begin{aligned}
\vec{m}_{t+1}^s &= \vec{m}_t^s + \alpha_t[\vec{x} - \vec{m}_t^s] \\
\vec{m}_{t+1}^p &= \vec{m}_t^p - \alpha_t[\vec{x} - \vec{m}_t^p]
\end{aligned} \tag{4}$$

It can be shown (Kohonen, 1995) 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. Moreover, the hypothesis margin of the classifier (Crammer et al., 2002) is given by:

$$\frac{\|\vec{x} - \vec{m}^s\| - \|\vec{x} - \vec{m}^p\|}{2}.$$

Hence LVQ2 can be seen as a classifier which aims at structural risk minimization during training, comparable to *Support Vector Machines* (Vapnik, 1998). Therefore a very good generalization ability of LVQ2 can be expected also for high dimensional data.

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

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

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

5.2. Classification with Rejection

In practical applications, it is usually associated to the recognition of a gesture the performing of a given action, e.g., the starting of a multimedia performance. In this applicative scenario, it is desiderable that the classifier recognizes a gesture, i.e., classifies, only when the probability of making a mistake is negligible. When the probability of making a mistake is not negligible, the classifier has to *reject* the gesture, i.e., it does not classify. We implemented a rejection scheme in the classifier in the following way. Let d be the Euclidean distance between the input $\vec{x}$ and the closest codevector $\vec{m}^c$, the following rule is applied:

$$\textsf{If } d \leq \rho \quad \textsf{then classify} \quad \textsf{else reject}$$

where ρ is a parameter that manages the trade-off between error and rejection.

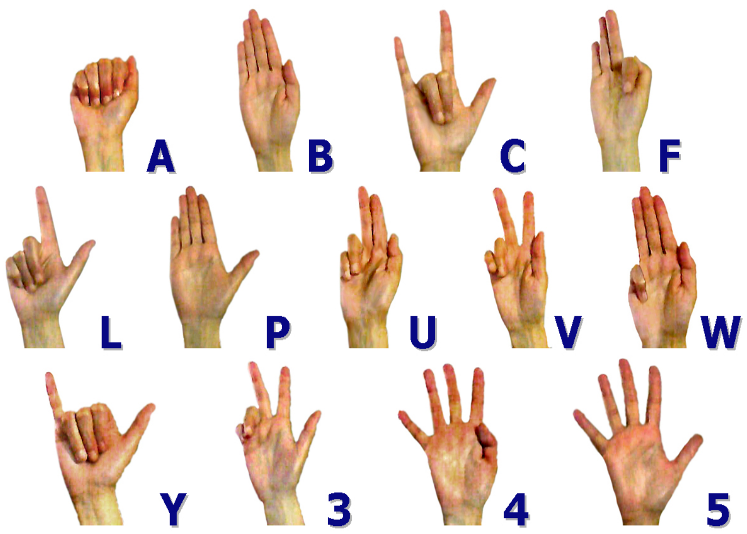


Figure 5: Gestures represented in the database.

6. Experimental Results

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 Fig. 5. 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 gestures. 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., 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 *10-fold crossvalidation* (Stone, 1974; Hastie et al., 2001). LVQ trials were performed using *LVQ-pak* software package (Kohonen et al., 1996). Fig. 6 shows the gesture distribution in the test set. In Table 1,

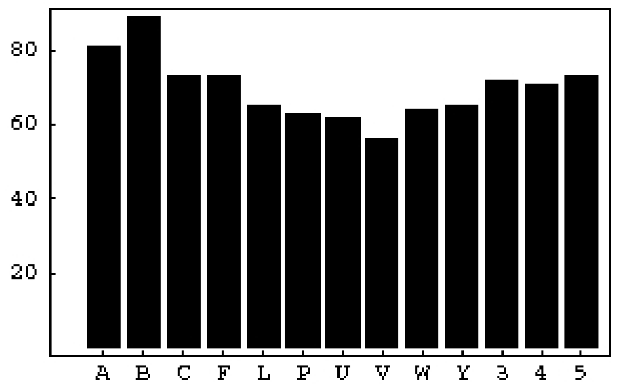


Figure 6: Gesture distribution in the test set.

for different classifiers, the performances on the test set, measured in terms of recognition rate in absence of rejection, are reported. The best classifier on the

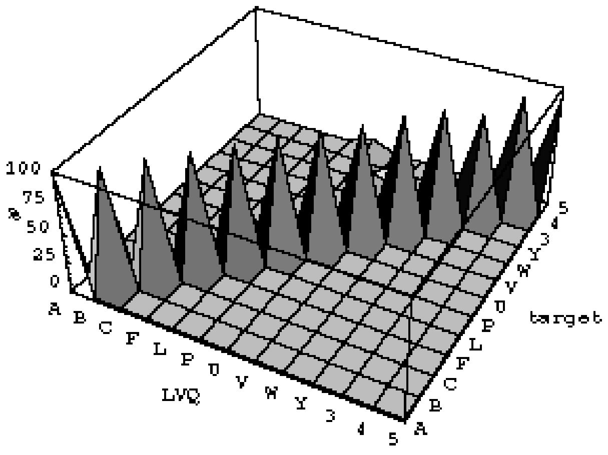


Figure 7: The confusion matrix for the best LVQ classifier on the test set.

test set, in absence of rejection, has a correct recognition rate of *98.46%*. The confusion matrix for the best classifier, on the test set, is shown in Fig. 7.
To our best classifier, i.e., LVQ1+LVQ2, the rejection rule, described in Section 5.2, was applied. The results obtained for different values of the rejection threshold ρ are shown in Table 2. Asking to the recognizer a negligible error, e.g., less than 0.6%, the recognizer can still guarantee a high correct recognition rate, i.e., close to the 98.0%.
A version of Handy[2] where to the recognition of a gesture is associated the performing of a Powerpoint[3] command is actually in use at Istituto Tecnico Industriale "Enrico Medi" helping the first author in his teaching duties. Handy, implemented in C++ under Windows XP Microsoft and .NET Framework 3.5 on

[2]A demo of Handy can be downloaded from http://www.luigilamberti.it/Autore/Handy.mpg.

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

Algorithm	Correct Classification Rate
knn	87.54 %
LVQ1	96.58 %
LVQ1 + LVQ2	**98.46** %
LVQ1 + LVQ3	98.24 %

Table 1: Recognition rates on the test set, in absence of rejection, for several LVQ classifiers. In bold it is indicated the classifier with the highest correct classification rate.

ρ	Correct	Error	Reject
0.42	98.46 %	1.54 %	0.00 %
0.33	98.02 %	1.21 %	0.77 %
0.30	97.68 %	0.55 %	1.77 %
0.26	95.26 %	0.22 %	4.52 %
0.19	79.05 %	0.00 %	20.95 %

Table 2: Correct, Error and Reject rates on the test set of LVQ1+LVQ2 for different rejection threshold ρ.

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.
Handy compares favourebly with data gloves in terms of invasivity, costs and performances. Firstly, our system is less invasive than a data glove. Wearing a wool glove is much more comfortable than wearing a data glove since the latter is plenty of sensors and wires. Regarding the performances, most data gloves cannot measure angles between fingers. This implies that they cannot discriminate gestures that differ each from other only by the angles between fingers, e.g., the gestures 'B', 'P' and '5' in Fig. 5. The unique data glove able to measure angles

between fingers is *Cyberglove*[4] (Drew Kessler et al., 1995). Nevertheless, since its cost can exceed several thousand of dollars, Cyberglove is too expensive to be used extensively in classrooms or in other field of interest.

7. Conclusions

In this paper we have described Handy, a real-time hand gesture recognition system based on a three 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 recognition rate larger than 98%. The system implemented on a netbook requires an average time of 140 CPU msec to recognize a hand gesture. As a general comment, we can claim that Handy compares favourably, in terms of costs, invasivity and performances, with data gloves. Finally, in the next future we plan to investigate the usage of Handy in teaching, virtual reality and motor rehabilitation applications.

Acknowledgments

The authors wish to thank the anonymous reviewers for their valuable comments. Luigi Lamberti dedicates the work to his wife Rosanna for her priceless support and encouragement.

References

A. Kendon, How gestures can become like words, in: Crosscultural perspectives in nonverbal communication, Hogrefe, Toronto, 131–141, 1988.

[4]Cyberglove is a registered trademark by Immersive Corp.

S. Mitra, T. Acharya, Gesture Recognition: A Survey, IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 37 (3) (2007) 311–324.

T. Frantti, S. Kallio, Expert System for gesture recognition in terminal's user interface, Expert Systems with Applications 26 (2) (2004) 189–202.

D.-Y. Huang, W.-C. Hu, S.-H. Chang, Gabor filter-based hand-pose angle estimation for hand gesture recognition under varying illumination, Expert Systems with Applications 38 (5) (2011) 6031–6042.

C.-Y. Tsai, Y.-H. Lee, The parameters effect on performance in ANN for hand gesture recognition system, Expert Systems with Applications 38 (7) (2011) 7980–7983.

A. Chaudhary, J. Raheja, K. Das, S. Raheja, A Survey on Hand Gesture Recognition in Context of Soft Computing, in: Advanced Computing, Springer, 46–55, 2011.

G. Burdea, P. Coiffet, Virtual Reality Technology, John-Wiley & Sons, New York, 2003.

L. Dipietro, A. Sabatini, P. Dario, A Survey of Glove-Based Systems and Their Applications, IEEE Transactions on Systems, Man and Cybernetics 38 (4) (2008) 461–482.

R. Wang, J. Popovic, Real-Time Hand-Tracking with a Color Glove, ACM Transactions on Graphics 28 (3) (2009) 461–482.

R. Gonzales, R. Woods, Digital Image Processing, Prentice-Hall, Upper Saddle River, 2002.

A. DelBimbo, Visual Information Processing, Morgan Kaufmann Publishers, San Francisco, 1999.

H. Cheng, X. Jiang, Y. Sun, J. Wang, Color image segmentation: advances and prospects, Pattern Recognition 34 (12) (2001) 2259–2281.

G. Bradski, A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly, Cambridge (USA), 2008.

L. Lamberti, Handy: Riconoscimento di semplici gesti mediante webcam., b. Sc. Dissertation (in Italian), University of Naples "Parthenope", 2010.

Y. Ren, C. Gu, Real-Time Hand Gesture Recognition Based on Vision, in: Entertainment for Education, Digital Techniques and Systems, Springer, 468–475, 2010.

M.-K. Hu, Visual Pattern Recognition by Moment Invariants, IRE Transactions on Information Theory 8 (2) (1962) 179–187.

C. Cortes, V. Vapnik, Support Vector Networks, Machine Learning 20 (1995) 1–25.

B. Schölkopf, A. Smola, Learning with Kernels, MIT Press, Cambridge (USA), 2002.

J. Shawe-Taylor, N. Cristianini, Kernels Methods for Pattern Analysis, Cambridge University Press, 2004.

R. Herbrich, Learning Kernel Classifiers, MIT Press, Cambridge (USA), 2004.

T. Kohonen, Learning Vector Quantization, in: The Handbook of Brain Theory and Neural Networks, MIT Press, 537–540, 1995.

T. Ho, Recognition of handwritten digits by combining independent learning vector quantizations, in: Proceedings of the Second International Conference on Document Analysis and Recognition, IEEE, 818–821, 1993.

F. Camastra, A. Vinciarelli, Cursive character recognition by learning vector quantization, Pattern Recognition Letters 22 (6-7) (2001) 625–629.

A. Zapranis, P. Tsinaslanidis, Identification of the Head-and-Shoulders Technical Analysis Pattern with Neural Networks, in: Artificial Neural Networks-ICANN 2010, Lecture Notes on Computer Science vol. 6354, Springer, 130–136, 2010.

P. Ciosek, W. Wróblewski, The analysis of sensor array data with various pattern recognition techniques, Sensors and Actuators B: Chemical 114 (1) (2006) 85–93.

X. Mouy, M. Bahoura, Y. Simard, Automatic recognition of fin and blue whale calls for real-time monitoring in the St. Lawrence, Journal of the Acoustical Society of America 126 (6) (2009) 2918–2928.

T. Kohonen, Self-Organizing Maps, Springer, Berlin, 1997.

R. Duda, P. Hart, D. Stork, Pattern Classification, John-Wiley & Sons, New York, 2001.

K. Crammer, R. Gilad-Bachrach, A. Navot, N. Tishby, Margin analysis of the LVQ algorithm, in: Advances in Neural Information Processing Systems 2002, MIT Press, 109–114, 2002.

V. Vapnik, Statistical Learning Theory, John Wiley and Sons, New York, 1998.

M. Stone, Cross-validatory choice and assessment of statistical prediction, Journal of the Royal Statistical Society 36 (1) (1974) 111–147.

T. Hastie, R. Tibshirani, R. Friedman, The Elements of Statistical Learning, Springer, 2001.

T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, K. Torkkola, LVQ-PAK: The Learning Vector Quantization Program Package, Tech. Rep. A30, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996.

G. Drew Kessler, L. Hodges, N. Walker, Evaluation of the CyberGlove as a Whole-Hand Input Device, ACM Transactions on Computer-Human Interaction 2 (4) (1995) 263–283.