Recognition of Hand Gestures Tracked by a Dataglove: Exploiting Hidden Markov Models Discriminative Training and Environment Description to Improve Recognition Performance

Vittorio Lippi, Emanuele Ruffaldi, Carlo Alberto Avizzano and Massimo Bergamasco

PERCRO laboratory, Scuola Superiore Sant'anna, Pisa, Italy

Abstract. Sequence classification based on Hidden Markov Models (HMMs) is widely employed in gesture recognition. Usually, HMMs are trained to recognize sequences by adapting their parameters through the Baum-Welch algorithm, based on Maximum Likelihood (ML). Several articles have pointed out that ML can lead to poor discriminative performances among gestures. This happens because ML is not optimal for this purpose until the modellized process is actually an HMM. In this paper we present a gesture recognition system featuring a discriminative training algorithm based on Maximal Mutual Information (MMI) and the integration of environment information. The environment is described through a set of fuzzy clauses, on the basis of which a priori probabilities are computed. Adaptive systems such as unsupervised neural networks are used to build a codebook of symbols representing the hand's states. An experiment on a set of meaningful gestures performed during the interaction with a virtual environment is then used to evaluate the performance of this solution.

1 Introduction

1.1 Aim of the Work

Gesture recognition is becoming an important task in technology and science. At first it can be considered as an advanced way to interface the user to an interactive system: exploiting the significance of natural human actions as commands would increase the efficiency, both in using and in learning how to use the system[1].

While performing an operation in a virtual environment a user can test the effectiveness of his practical choices and train himself to work in various situation. Meanwhile the gesture recognition system could trace the steps performed: a model of the behavior of a skilled performer could be produced, and a generic user can be aided by the system to reproduce it. This can be useful in context where staff training, and producing documentation are crucial economical issues in the working process[2].

The gesture recognition system described in this work is designed to provide a tool to track tasks of interest during the assembly of mechanical parts. This system has been developed together with a virtual environment where the task is set. The system is oriented to scalability, in order to be suitable to work on task of realistic complexity. To

Lippi V., Ruffaldi E., Avizzano C. and Bergamasco M. (2009).
Recognition of Hand Gestures Tracked by a Dataglove: Exploiting Hidden Markov Models Discriminative Training and Environment Description to Improve Recognition Performance.
In *Proceedings of the 5th International Workshop on Artificial Neural Networks and Intelligent Information Processing*, pages 33-42
DOI: 10.5220/0002206800330042

achieve this the system takes advantage of discriminative training to adapt the hidden Markov models (HMM) used to recognize gestures and a system that produces prior probabilities for gestures. Prior probabilities are computed on the basis of the situation in which the gestures are performed. This makes possible to compare, at every recognition performed, just a small subset of plausible gestures, making the system scalable to an higher number of gestures. If an information about prior probabilities is known during the classifier training, the algorithm used incorporate it. The result is a classifier trained focusing on the differences between gestures more likely to be performed in the same context (and hence confused). Notice that when performing a meaningful task every object has really few ways to be used.

1.2 Structure of the Paper

Sections 2, 3 and 4 describe the solution proposed. In particular section 2 describes how a codebook to translate raw data into symbols is built; section 3 describes how HMMs are trained to classify sequences of symbols produces by user gestures and section 4 describes how the environment description is implemented and integrated with the recognition process. In section 5, an application of the system is presented together with the recognition accuracy achieved. Conclusions and planned improvements are in the section 6.

2 Codebook Definition and Use

2.1 Components and Parameters

Three steps are performed in order to build the codebook:

- complexity reduction by principal component analysis (PCA);
- frequency analysis by Fourier transform;
- quantization by neural networks;

The encoding process is summarized in figure 1, where a series of 11 signals is reduced to 3 by PCA. Note that once the PCA is applied the other two operations are applied independently on the series. This elaboration requires several parameters to be specified:

- the number of series kept after PCA. This parameter determines the total variance preserved;

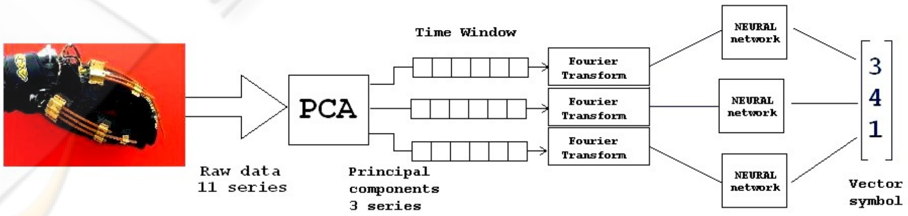

Fig. 1. Encoding input data.

- the time window on which the Fourier transform is applied. This is a trade off between the precision and the responsiveness of the system;
- the number of overlapping samples between time sequences;
- the number of symbols that the sequence classifier should recognize: If the symbols are too many the parameters of the sequence classifier can grow excessively in number and overfitting to the training set can occur.

2.2 Principal Component Analysis

Principal component analysis[3] is a procedure to obtain a linear transformation that, when applied to data series results in independent series. The series produced are ordered on the basis of the variance they hold: almost the whole variance of the system is kept using just some of them.

Usually PCA is applied to data offline, once a series is fully available. In this work an online encoding is needed. The codebook should be defined in a unique way, so the transformation can not be computed online just on a given time window. The training set is hence used to define the transformation offline and to estimate how many components should be hold to achieve a decent precision. The transformation is then applied online to incoming data.

2.3 Frequency Analysis

A discrete Fourier transform is applied on the series before using the neural networks. This allows the system to focus on the dynamics more than on the position. To make the results independent from the position, the first component of the Fourier transform, which represents the mean, is dropped from the obtained vector. The Fourier transform results in a complex vector, and in this form it is used as input for the neural network.

The implemented neural networks can work with complex values as well as real ones. When just the modulus of the Fourier transform is used as input for the neural networks the performance can be strongly degraded. This happens because, often, the phase is an important element in defining the nature of a gesture.

2.4 Neural Quantization

The neural networks are used to finally produce the encoding symbol. The competitive neural network exploited performs a quantization splitting the input space into several clusters[4].

The neural network output is an integer representing the cluster in which the input vector ha been classified. These numbers have no particular meaning, and no relation one with the other.

During the training phase the same training set is used both for the neural networks and the HMMs. The neural networks are trained before the HMMs because they are needed to produce the symbols. The networks are trained by iteratively minimizing the quadratic quantization error on the training set.

3 Sequence Recognition

3.1 Classification Principles

In recognition applications HMM are trained through examples to modelize the process to recognize,typically with an HMM for every process[5] [6]. The classification is usually performed by the assignment of a given input sequence to the class associated to the HMM with the highest probability to emit it, as described in the formula:

$$C = maxarg_i \left(P(S|M_i)\right) \tag{1}$$

Where C is the class given as output for the input sequence S, and M is the performed gesture. In this application, for reasons that will be explained in the next section, the classification is performed by a different formula:

$$C = maxarg_i \left(P(M_i|S)\right) \tag{2}$$

By this formalism, which represents the emission probability as a conditioned probability, the sequence is considered as the evidence produced by the gesture performed. Every gesture M_i is associated to its own probability

$$P(M_i) \tag{3}$$

That is supposed to be given *a priori* and not to be dependent on parameters. This makes sense considering P(M_i) to be a higher level concept than P($S|M_i$). For example, when recognizing gestures, the probability of a sequence to be observed during a gesture is related to movement coordination (low level), while the probability of a gesture to be performed is connected to the task performed (high level). The probability of a sequence S to be observed (namely, produced by the models featured by the classifier) is

$$P_\theta(S) = \sum_i P_\theta(S|M_i)P(M_i) \tag{4}$$

where the subscript θ marks the probabilities dependent on the parameters, represented by the vector θ

3.2 Parameter Estimation

Training the parameters of an HMM to maximize the emission probability of a given sequence is a well known problem for which several algorithms have been proposed in literature. Maximization of likelihood is the basis of the most of them. But representing a well-known and (relatively) easy principle to match the model to the distribution of examples is not optimal for classification problems in the general case[7]

Hence the principle of Maximum Mutual Information (MMI) is followed. The Mutual Information between a sequence S and the model M_i associated to the class *i* is:

$$I(M_i; S) = \log \frac{P_\theta(S, M_i)}{P_\theta(S)P(M_i)} = \log P_i(S|M_i) - \log(P(S)) \tag{5}$$

The aim of the training is to maximizing the Mutual Information between the given sequence and the model representing the class which the sequence is supposed to be classified in. Another interpretation of MMI[8] is useful to show how the term $P(M_i)$ is not dependent on θ:

$$I(M_i; S) = \log P_\theta(M_i|S) - \log P(M_i) \tag{6}$$

Finding the MMI over θ is equivalent to maximize $P_\theta(M_i|S)$, or adapt parameters to *enhance the probability that, **if** the sequence S is observed, it has been produced by the process* M_i. Notice again that $P_\theta(M_i|S)$ is a property of the whole classifier rather than the single HMM. Unlike ML, MMI approach could not be implemented by an expectation-maximization algorithm[9]. A gradient algorithm is therefore exploited.

The sets of sequences and models are considered in the training by maximizing the function:

$$F_\theta = \sum_i \sum_k I_\theta(M_i; S_k) \tag{7}$$

So the gradient is the sum of the gradients of $I_\theta(M_i; S_k)$ for every HMM M_i and every sequence S_k:

$$\nabla_\theta F_\theta = \sum_i \nabla_\theta \sum_k I_\theta(M_i; S_k) \tag{8}$$

Function 7 resembles the mutual information defined between two probability distributions of sequences and gestures that caused it

$$\sum_i \sum_k P(M_i, S_k) I_\theta(M_i; S_k) \tag{9}$$

but the term $P(M_i, S_k)$ is missing because the probability distribution $P(M, S)$ is not directly available, and computing it from the emission probabilities would cause the gradient to lose the property stated in 8. Anyway the training set itself works as a representation of sequences distribution, in the sense that a sequence S_k associated to an high actual $P(M_i, S_k)$ is represented by a large number of sequences similar to it in the training set, so the formula 7 is an approximation of formula 9. The derivative of formula 7 respect to a parameter θ_i could be written as

$$\frac{\partial I(M; A)}{\partial \theta_i} = \frac{\frac{\partial P_\theta(S|M)}{\partial \theta_i}}{P_\theta(S|M)} - \frac{\sum_{\hat{M}} \frac{\partial P_\theta(S|\hat{M})}{\partial \theta_i} P(\hat{M})}{P_\theta(S)} \tag{10}$$

where $\hat{M}$ are the generic models and M is the model related to the class wanted to recognize the sequence S. In order to compute

$$\frac{\partial P_\theta(S|M)}{\partial \theta_i} \tag{11}$$

the parameters could be evidenced in $P_\theta(S|M)$, for a and b

$$P_\theta(S|M) = \sum_{t=1}^{T} \sum_i \alpha(t-1) a_{ij} b_j(O_t) \beta(t) \tag{12}$$

and for π:

$$P_\theta(S|M) = \sum_i \pi_i a_{ij} b_j(O_0)\beta(0) \tag{13}$$

Applying the product rule to compute, for example, the derivative for a:

$$\begin{aligned} &\tfrac{\partial P_\theta(S|M)}{\partial a_{ij}} = \textstyle\sum_{t=1}^{T}\sum_i \alpha(t-1)\tfrac{\partial a_{ij}}{\partial a_{ij}} b_j(O_t)\beta(t) = \\ &\textstyle\sum_{t=1}^{T}\sum_i \alpha(t-1) b_j(O_t)\beta(t) \end{aligned} \tag{14}$$

Only the sequences considered likely to be confused are chosen as negative example. A *confusion margin* is quantified by the formula:

$$P(M_k|S_j) > \alpha P(M_i|S_j) \tag{15}$$

Where S_j is a sample for the class i, and k is the class which takes S_j as negative example when the condition is verified and α is a parameter to be specified: for $\alpha = 0$ all the sequences are taken in account, for $\alpha = 1$ only the ones misclassified at a certain step of the training algorithm become negative examples.

As previously explained $P(M|S)$ is connected with prior probabilities. This allows to take advantage from high level information in the training process. Batches of samples could be presented with different prior probabilities in order to decide which classes should be affected by negative training.

For example, if a sample S_j for the class i is presented during the training together with a null prior probability $P(M_k)$,S_j will not be a negative example for the class k, being $P(M_k|S_j) = 0$ and so, never greater than $\alpha P(M_i|S_j)$ that is never negative. This is useful because there are some gestures that could occur together in a certain environment and in a certain situation and hence *a priori* more likely to be confused.

4 Environment Description

4.1 Representation through Fuzzy Logic

An *ad hoc* defined language [10] is used to describe a logical model of the environment: objects, classes of objects and proposition about both objects and classes can be defined in a fashion that represents a simplification of a complete syntax for predicative logic. Fuzzy logic is particularly suitable to represent physical concepts like proximity or alignment[11].

In details the environment description consists of:

- a set of statements with their truth values, needed to describe environment state i.e., ”the hand is near the screw, with truth value 0.5”;
- a set of rules to compute the truth values of statements from other statements truth values, used to compute the state of the system and to update the description coherently i.e., ”if a screw is screwed on a bolt then that screw is blocked”;
- a set of rules to compute the probability of a certain gesture to be performed by the user, needed to use the environment description in the recognition process, a brief explanation of the method used is in the next paragraph;

- a set of rules to define the consequences of an action performed by the user, needed to update the scheme in realtime i.e. ”if the user performs the action *pick* and the hand is near an object that object is then in-hand with truth value 1”;

Describing the language in details is beyond the intention of this article, the syntax actually used in the application is hence not presented in this context where it could appear confusing. The examples in the previous list, given in plain text, are intended to show quickly the kind of statements that the described system can represent.

4.2 Inference Engine and Prior Probabilities

The *inference engine* is the set of functions that computes truth values of arbitrary statements from the logical description. In this section the algorithms exploited are described. When the truth value of a statement is required, the following action are performed by the engine:

1. The statement is searched in the description, if it is present its truth value is returned and the research is ended;
2. A rule having the state as *effect* is searched. If found the procedure is repeated for all the terms in the *cause*. The value of truth is than computed and returned.
3. If a value is not found in previous steps, 0 is returned.

Searching the truth value of a statement is like building an *inference tree* where the nodes are rules and leaves are known statements. To avoid that an infinite loop arises sub-threes in the *inference tree* cannot include the rule that represent their root. There is another feature that makes this system different from an usual theorem demonstrator[11]: it relies on rules written by the user to compute the truth values of statements from other statements truth values the latter exploits fixed inferential rules to find true sentences from true sentence.

A vector of gestures prior probabilities is specified for several logical condition. The logical conditions are fuzzy statements. When prior probabilities are required by the application a weighted sum of all the prior probability vectors is given. The truth value of the condition is used as weight for the related vector.

5 Performance

5.1 An Application

This application consists in assembling a small computer case in the virtual environment. The user interacts with the components through an avatar of his/her hand. The

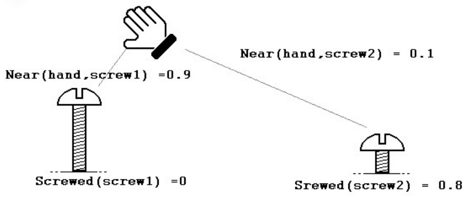

Fig. 2. A situation in a possible environment and fuzzy values for assumption on it.

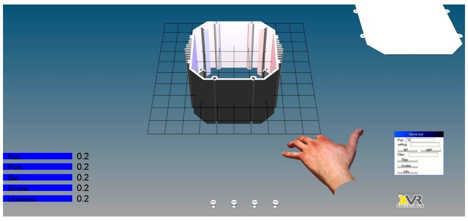

Fig. 3. The virtual environment at the beginning of the task.

recognized gestures trigger effects in the virtual environment. The finger movements performed are registered by sensor equipped glove [12]. The three-dimensional representation of the environment is updated coherently with the logical description. In order to build the case components should be put in place, set with the right orientation and than blocked. To complete the task the operations should be performed in the right order. The case is made with a box, a base and four screws to fix them together. The recognized gestures are: *null*, *pick*, *set*, *screw* and *unscrew*. The gesture *null* does not produce any action, it is recognized when the hand is almost still. It is needed to avoid recognition of random gestures when the use is not doing something. The gesture *pick* is used to take and release objects. *Set* is used to put an object in the right position for assembling. This overcomes the fact that in this application there is not a force feedback and the right position is hence difficult to find. Notice that ”being set” is a state of an object represented in the logical environment and it is anyway needed to mount an piece. The gestures *screw* and *unscrew* are used to fix screws on the box and to remove them. There are 5 degrees of screwing represented by a natural number, when the gesture *screw* is recognized the number is increased, on the opposite unscrewing decreases the number.

5.2 Accuracy

The system has been tested using examples of the 5 gestures produced by 5 users. Every user has been asked to perform every gesture for a minute. To build the codebook the 11 angular values describing the finger positions, sampled at 1 MHz are reduced to 3 series of data through a PCA, then the Fourier transform is applied on overlapping time windows of 50 samples. The resulting sample series have been divided in sequences of 8 symbols. The system has been cross-validated splitting the set of examples into 5 subset, used 4 by 4 as training set.

The main peculiarities of the proposed approach are the use of MMI instead of ML and the use of environment informations. Hence the test consists in comparing confusion matrices in case of HMM trained through ML and through MMI without and with environment information.

The MMI training outperforms ML training. This is evident especially for the gestures *screw* and *unscrew* that are particularly difficult to recognize, being very similar.

The last two tables show the improvement produced by the use of context information. As told in the tables captions, the information added consist just in the hand position and the screw state. Further performance gains would be possible adding information about the task performed by the user, for example taking in account that the

Table 1. Confusion matrix for MMI training.

	null	pick	set	screw	unscrew
null	**0.946**	0	0.054	0	0
pick	0	**0.927**	0	0.037	0.037
set	0.060	0.016	**0.890**	0.020	0.012
screw	0.008	0.042	0	**0.832**	0.117
unscrew	0	0.008	0	0.096	**0.896**

Table 2. Confusion matrix for ML training.

	null	pick	set	screw	unscrew
null	**0.915**	0.015	0.010	0.030	0.010
3	0.070	**0.668**	0.061	0.131	0.050
set	0.060	0.10	**0.702**	0.048	0.090
screw	0.070	0.80	0.030	**0.660**	0.150
unscrew	0.080	0.70	0.080	0.140	**0.680**

Table 3. Confusion matrix for MMI training, the recognition exploits context information: the hand is near a screw positioned to be screwed, the precision rises to 96.83%.

	null	pick	screw
null	**1**	0	0
pick	0	**0.934**	0.065
screw	0.008	0.021	**0.970**

user started to screw after positioning a screw would make more likely his/her intention to screw.

6 Conclusions and Future Work

The objective of the presented work was to improve HMM-based gesture recognition performance optimizing parameters adaptation and including high level information in the recognition process, as usually done in speech recognition applications where language models are exploited.In the proposed system this is represented by environment information. A test shown how adapting parameters with MMI produces a better classification performance compared with the usual ML-based training presented in literature, expecially in the classification of similar gestures. The algorithm high computational requirements have been compensated selecting a subset of the training samples at every step, without affecting the recognition performance of the algorithm. The introduction of environment information produced a further general performance improvement. The proposed fuzzy logic simplified inference system assures an efficient realtime update of environment description according to user actions.

In the presented version the action performed by the user can be saved as a list. It should be combined with the environment informations and, produce automatically a manual for the task performed. Another step forward will be the integration of this sys-

Table 4. Confusion matrix for MMI training, the recognition exploits context information: the hand is near an halfway screwed screw, the precision rises to 92.21%.

	null	screw	unscrew
null	**1**	0	0
screw	0	**0.935**	0.065
unscrew	0.084	0.021	**0.970**

tem into an application representing a complex industrial task, including the possibility for the user to use tools.

References

1. Melinda M. Cerney, J.M.V.: Gesture recognition in virtual environments: A review and framework for future development. Technical report, Iowa State University (2005)
2. Brough, J. E., Schwartz, M., Gupta, S.K., Anand, D.K., Kavetsky, R., Pettersen, R.: Towards the development of a virtual environment-based training system for mechanical assembly operations. Virtual Real. 11 (2007) 189–206
3. Moeslund, T.B.: Principal component analysis - an introduction. Technical Report Technical Report CVMT 01-02, ISSN 0906-6233, Aalborg University (2001)
4. Mathworks: Matlab manual. Mathworks. (2004)
5. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
6. L.R. Rabiner, B.J.: An introduction to hidden markov models. IEEE ASSP Magazine 3 (1986) 4– 16
7. Chow, Y.L.: Maximum mutual information estimation of hmm parameters for continuos speech recognition using the n-best algorithm. IEEE 2 (1990) 701–704
8. Bridle, J.S.: Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. Advances in neural information processing systems 2 (1990) 211–217
9. B. H. Juang, L.R.R.: Hidden markov models for speech recognition. Technometrics 33 (1991) 251–272
10. Lippi, V.: Design and development of a human gesture recognition system in three-dimensional interactive virtual environment. Master's thesis, University of Pisa, College of engineering. (2008) Consultation Allowed.
11. Stuart Russel, P.N.: Artificial Intelligence: A Modern Approach. Pearson (2003)
12. Rodriguez, O.P., Avizzano, C., Sotgiu, E., Pabon, S., Frisoli, A., Ortiz, J., Bergamasco, M.: A wireless bluetooth dataglove based on a novel goniometric sensors. Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on 1 (2007) 1185– 1190