Learning context models for the recognition of scenarios

Sofia Zaidenberg, Oliver Brdiczka, Patrick Reignier and James Crowley
Laboratoire GRAVIR
655, avenue de l'Europe
38334 Saint-Ismier Cedex, France
{Zaidenberg, Brdiczka, Reignier, Crowley}@inrialpes.fr

Abstract. This paper addresses the problem of automatic learning of scenarios. A ubiquitous computing environment must have the ability to perceive its occupants and their activities in order to recognize a context and to provide appropriate services. A context (a scenario) can be modeled as a temporal sequence of situations. Hard coding contexts by hand is a complex task. Our goal is to learn these context models based on a set of videos showing actors playing predefined scenarios. Once these models are learned, we can use them to classify new scenarios. Hidden Markov Models (HMMs) are particularly well suited for problems with a strong temporal structure; they are easily adaptable to variability of input and robust to noise. But two problems need to be addressed: how many HMMs do we need for all possible scenarios and how many states for each HMM. We propose in this paper an approach based on an incremental algorithm addressing these two problems. Under the best conditions we obtained the minimal error rate of 1.96% (2 errors in 102 validation entries).

1 Introduction

The goal of Ubiquitous computing is to build a computerized space serving human activities. This computerized space has to take into account the multiplicity of the platforms and to *perceive the context* for a better comprehension and anticipation of the user's needs.

A context can be defined as a temporal sequence of situations [2]. A situation is a set of entities playing roles. For example, a lecture scenario can be defined as a set of four situations: persons entering a room *followed by* an alternation of lecturer speaking and someone in the audience asking a question, *followed by* attendees

Please use the following format when citing this chapter:
Zaidenberg, Sofia, Brdiczka, Oliver, Reignier, Patrick, Crowley, James, 2006,in IFIP International Federation for Information Processing, Volume 204, Artificial Intelligence Applications and Innovations, eds. Maglogiannis, I., Karpouzis, K., Bramer, M., (Boston: Springer), pp. 86–97

leaving the room. Specifying a context manually can be a difficult task. The goal here is to let the system automatically learn the context model from a training set of videos.

The contexts we want to learn are: *walking*, *browsing*, *fainting*, *leaving bags behind*, *meeting*, *walking together and splitting up*, *two people fighting*. The videos are part of the CAVIAR project and can be found on the project web site [1].

The Caviar project defines a hierarchy of perceptual components (see Fig. 1). It goes from low level images analysis to high level context interpretation. Each video is associated with an XML file describing for each frame the entities with their position, movement, role and situation. Groups of entities are detected as well, and described by the same elements (movement, role and situation). These files (called *Ground Truth*) have been created manually. They can be used to validate low level perceptual components or, in our case, to replace them as they were not available at the beginning of the project.

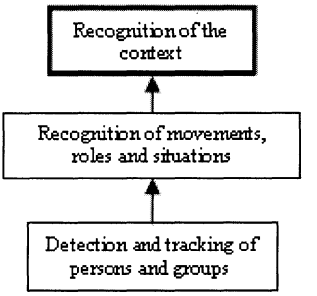

Fig. 1. Hierarchy of components for the perception of context

2 Related Work

The problem of recognizing human activities from videos using machine learning techniques is widely addressed. We can distinguish probabilistic approaches from deterministic ones. Most of the existing works in the probabilistic area use Hidden Markov Models (HMMs) because of their adequacy for temporally correlated sequential data. The study undertaken by [7] relates to the modeling of interactions for the automatic analysis of multimodal group actions in meetings. They first deal with individual actions and then model the interactions by HMMs. Group actions in meeting were as well studied by [14] who proposed a two layer HMM framework. A lot of work has been done to detect usual and unusual activities [1], [4], [5], [6], [13], [15]. Other probabilistic methods have been undertaken: [12] models sequential activities by *Propagation Networks* which can take into account parallel activities. An alternative to HMM methods are *Context-free Grammars* used by [3], [8], [9]. There is no learning in this case and the model is predefined.

[1] European CAVIAR project/IST 2001 37540: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

All these methods, as they are presented, are not quite appropriate for our problem because we do not deal directly with videos but we have higher level information like the movement or the role of entities at every frame. For instance, we know that at a particular frame, a person is "*walking*" (her `movement`) and is a "*fighter*" (her `role`).

However, our activities have a temporal structure and we need the model to integrate variability of input. We also need robustness to noise because of the perception algorithms providing our input. These constraints point us to probabilistic methods, and precisely to HMMs.

When using HMMs, well-defined training algorithms, such as Baum-Welch [10], do exist, however the number of states of each HMM must be given "by hand". Some work has been done to optimize the number of states of each HMM in a classification system: [16] propose for instance to set the length of the model to a fraction of the average number of observations of the sequences used to learn the HMM. This fraction is chosen by measuring the recognition rate with different values. A classical model selection criterion is the Bayesian Information Criterion [11] (*BIC*), which maximizes the likelihood of the data while penalizing large-size models. Our method as well tries different numbers of states of each HMM. However our method is exhaustive: we keep several HMMs with different numbers of states (see section 3.3.1).

3 An Incremental Algorithm to Learn Models of Scenarios

3.1. An Approach to Learn Models of Human Activity

Our training set is composed of videos where main actors are playing a predefined scenario. However, secondary actors may be present and play a different scenario. Thus we separate different individuals for each video and label their activity.

To learn the context models, we first classically split the video set into two separate sets: the training and the validation set. To learn the training set, we have considered two approaches: a *supervised* one and an *unsupervised* one. In the first case, we identify five different scenarios and we assign a number of individuals to each scenario. A model is then learned for each scenario in order to represent the assigned individuals. This manual method was tested but, as it will be clear in section 4, results obtained are not as good as with the unsupervised method. Thus, only this last method will be detailed in the following.

In the second case, we use an automatic and *incremental* algorithm (see section 3.3 below) to learn the necessary number of models. We consider each person and compute a "score" of the person's activity on each existing model and compare it to a fixed threshold in order to see if we already have a model that describes this activity. If so, we re-learn that model including this activity. Else, we create a new model learned with the person's activity.

3.2. Feature Extraction from the Ground Truth

The input of an HMM is a sequence of observations. These observations are triplets per frame and entity. They are given by the *Ground Truth* (see Fig. 2). For practical reasons, we transform these triplets into numbers with no information loss [2].

```
<movement evaluation="1.0">walking</movement>
<role evaluation="1.0">walker</role>
<situation evaluation="1.0">moving</situation>
```

Fig. 2. A quote from the XML description of a video.

Table 1. Numbering of all existing values of movements, roles and situations

- "movement": 4 different values

walking	1
inactive	2
active	3
running	4

- "role": 7 different values

walker	1
browser	2
none	3
fighter	4
leaving group	5
leaving victim	6
leaving object	7

- "situation": 4 different values

moving	1
browsing	2
inactive	3
none	4

Each symbol is first transformed into a number (Table 1). By enumeration, the 3-vector is then transformed into a unique number: the HMM observation. For instance the triplet `[walking, browser, moving]` becomes `[1, 2, 1]` and then it becomes the code `5`.

3.2.1. Normalizing the Sequences

An observation sequence is the series of observation codes of one entity through all video frames. These sequences are the inputs of our system.

The "score" of a sequence for an HMM is computed using the *Viterbi* algorithm. This "score" corresponds to the probability for a sequence to be generated by this HMM. The order of magnitude of this probability varies with the length of the sequence. If we add an observation to a sequence (for instance, an entity walking during 20 frames instead of 19), the context might not change, however probabilities on the various HMMs will.

As explained in section 3.3, we need to compare the absolute value (the "score") of different sequences on a same HMM. To be able to make this comparison, we must normalize the length of all the sequences. We have chosen a size of 100 observations, which seemed reasonable and much less than the original size of se-

[2] In particular, it is easier with the Java HMM library we are using, called *Jahmm*: http://www.run.montefiore.ulg.ac.be/~francois/software/jahmm/

quences[3]. Here is an example on a schematic sequence of the activity "browsing", which consists of alternating the punctual actions "walking" and "browsing" (the punctual action "browsing" means consulting an information desk):

168×"walking"	+	174× "browsing"	+	168× "walking"	=	510 observations
⇩		⇩		⇩		⇩
33× "walking"	+	34× "browsing"	+	33× "walking"	=	100 observations

We change the scale of the sequence without changing its aspect and the proportions of the different observations.

3.3. Automatic Choice of the Models to Learn

Persons with the same activity can produce very different observation series. Having just one HMM for each scenario means that this single HMM must be able to learn all those differ-ent observation series. At the same time, there are videos where the activity of a person is not very clear (for instance, someone coming near an information desk and slowing down in front of it: is it a browsing or just a walking by scenario?).

To overcome those problems, we have decided to let the system auto-organize the number of HMMs that are needed and how the videos will be grouped in terms of activities.

The incremental algorithm is described by Fig. 3 and is composed of the following steps:

1) **Initialization:** We create an initial HMM learned on one sequence randomly chosen. This first model is the starting point of the following loop.
2) **Loop on the sequences of the learning set:** We consider every sequence from the learning set at a time and apply the following steps:
   a) We evaluate the current sequence on each existing HMM using the Viterbi algorithm. We choose the HMM with the highest probability to have generated the sequence.
   b) If this maximal probability of "generation" is greater than a threshold[4], we assign the current sequence to the chosen HMM and we learn it again on the sequences it already had, plus the newly added sequence.
   c) If this maximal probability is less than the threshold, this means that there is no HMM that describes well enough the activity of the current sequence. This sequence represents a new scenario. We create a new HMM and learn it with the current sequence only. This new HMM is then integrated into the learning loop on the same basis as the other HMMs.

This algorithm lets the machine decide which and how many models to create. In the section 4.1 we will compare this method with the manual one.

---

[3] The average length of sequences is near 500. Sequences are that long because of numerous repetitions of attributes (movements or roles of entities do not change very often, they are stable).

[4] This threshold is set by hand, but we tested the influence of this choice in section **Error! Reference source not found.**

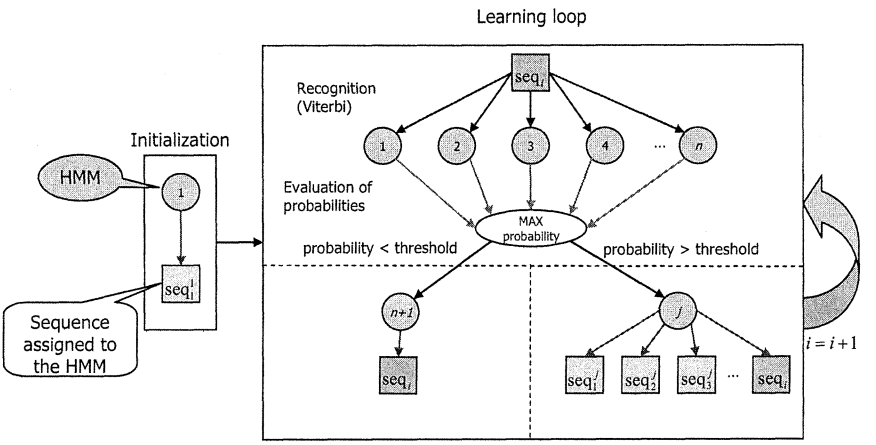

Fig. 3. Schema of the learning algorithm

3.3.1. Automatic Choice of the Number of States

One limitation of the HMMs is that we have to choose the number of states for each model. To avoid under fitting or over-fitting problems if the number of states is too low or too high, we learn several HMMs with different numbers of states for the same scenario. This method lets the machine decide and spares us a decision we might not be able to make correctly. We call this set of HMMs learned on the same sequences but with different numbers of states, a class.

In our experiments a class is composed of HMMs whose number of states varies between 1 and 8. We can count the number of sequences choosing each particular HMM and notice that some of them are never chosen and some are chosen more frequently then others. We could a posteriori decide to delete the unused HMMs and even to keep only the most popular one of each class. This would be similar to [16] where the authors choose the number of states of each HMM in order to maximize the global recognition rate. In our case there is no limitation in us-ing the exhaustive method.

3.3.2. Cross-Validation

In order to evaluate the quality of our models, we use *cross-validation*. We separate randomly our 147 sequences into two subsets:

- **A training set** composed of 30% of the sequences and used only to learn the models;
- **A validation set** composed of 70% of the sequences and used only to validate the models.

Because this partition has a huge influence on the resulting models, we repeat the whole process n times with a different random partition of the Ground Truth sequences each time.

3.3.3. Smoothing the Emission Probabilities

An HMM is defined with a vector of emission probabilities of every symbol of its vocabulary in each state. In our case the vocabulary is composed of all possible codes (see section 3.2), that is to say $M = \|V\| = 112$ symbols, and $V = \{1, 2, 3, \cdots, 111, 112\}$. So the emission vector in each state has 112 components, but in practice only a few of them are not zero. If the emission probability for a symbol, say the code 5, is zero in all states, a sequence which contains the code 5 will have a zero probability with that HMM. This makes the system very sensitive to noise and we need to avoid that.

We will apply the same principle as in "Laplace Smoothing": what was never seen is not impossible. We reduce the non-zero values to 90% and we equally distribute the collected value on the zeros. For instance with a vocabulary of $M = 7$ symbols we would perform the transformation opposite.

(0	0	**0.6**	0	**0.4**	0	0)
		⇩		⇩		
(0.02	0.02	**0.54**	0.02	**0.36**	0.02	0.02)

With this modification, a noisy sequence will still have a chance to be correctly classified. This also means that we will never have the probability 0, but the order of the probabilities for one sequence on all the HMMs will not change because of this smoothing.

3.3.4. Adding Information about Groups of Persons

The Ground Truth contains labels about groups of persons at each frame. These groups are labeled like the objects, with movement, role and situation attributes. We also have the infor-mation about who are the group members. We tested two methods taking this into account.

The first method consists in adding a boolean attribute of membership to each object at each frame. The codes are now based on the vector of attributes [movement, role, situa-tion, group], for instance [walking, browser, moving, 0].

But the Ground Truth contains hand made labels . A real tracker will not always be able to distinguish persons who are too close, within a group. When the tracker gets confused between persons, it will assign the attributes of the group to each member. We also made this substitu-tion in order to test this case and compare the results with the first method (see section 4.1).

3.3.5. Consequence of the Incremental Algorithm: the Labeling of the Models

Using the learned models for classification

Once the models learned, to classify a new entry the system will compare it to every model. Each model will compute a "score" as response to the entry. The output of the classification is the model or the class which had the best score and which correspond the best to the entry.

Evaluation of the classification output

To evaluate the results of our method we need a quality criterion: the percentage of misclassified validation sequences. Our output is a class of HMMs. To know if

the activity represented by this class corresponds to the activity of the sequence, we need a label for the sequence and one for the class. We already have labels of sequences, but how to label a class which was automatically created?

We get the label, which represents the scenario or activity modeled by the class, by looking at the labels of the sequences used to learn its HMMs. These labels are composed of key words like "walking", "waiting" or "fighting". A label can contain several key words when the activity is not clear or if there are two activities in the sequence. We count the number of times each key word appears in the labels and we keep the most frequent ones. So the labels of classes are also composed of several key words, which imply that the created classes may overlap.

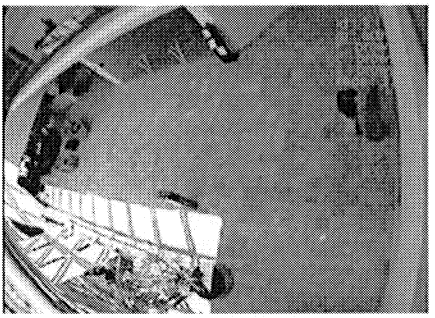

Fig. 4. A frame from the testing scenarios

4. Tests and Results

To realize the following tests we used the validation set described in section 3.3.2. Fig. 4 shows a frame from the CAVIAR scenarios used in this work where a person is reading an information desk. His role is therefore "browser".

4.1 Evaluation of the Incremental Algorithm and of the Additional Group Information

We tested 6 combinations of possibilities to compare the reference method where we define 5 classes by hand, and the incremental algorithm where classes are automatically created. In both cases we tested the two methods described in section 3.3.4 taking into account the information about groups of persons.

The results in Table 3 show that the automatic method where the machine organizes itself produces better results than when we choose the scenarios to learn by hand. Besides, the infor-mation about groups of individuals seems not to improve the results.

We can notice that the standard deviation of the results on different random partitions is quite high. For instance in the test number 5, executed on 20 random partitions, the average er-ror percentage is 10.44%, and the standard deviation is 5.40% (over 102 validation sequences). This is explained by the dependence of our method on the ordering of the training data.

Table 2. The tests executed on 20 random partitions (for tests 4, 5 and 6).

Test number	Method to create the classes		Groups of individuals		
	Supervised	Unsuper-vised	Ignored	Boolean attr.	Replace by group attrs.
1	×		×		
2	×			×	
3	×				×
4		×	×		
5		×		×	
6		×			×

Table 3. The results obtained on different tests

Type of result	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
Minimal PE[5] on validation sequences	11,65%[6]	13,40%	12,12%	1,96%	1,96%	1,96%
Minimal PE on learning sequences	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Average PE on validation sequences	11,65%	13,40%	12,12%	8,77%	10,44%	10,44%
Average PE on learning sequences	0,00%	0,00%	0,00%	3,89%	4,11%	4,00%
SD[7] of errors on validation sequences	11,65%	13,40%	12,12%	4,50%	5,40%	4,10%
SD of errors on learning sequences	0,00%	0,00%	0,00%	3,29%	2,63%	2,94%

4.2. Robustness to Noise

The labels of the *Ground Truth* are made by hand; therefore they do not contain noise. But in the real application of the system, the sensors and perception algorithms below our context recognition tool (Fig. 1) will add noise to the data. For the moment, we do not have the possibility to test our approach on real data, thus we measured the robustness to noise by adding simulated random noise [8].

The results on Fig. 5 below show that when the models are learned on clean data, the system can recognize validation sequences with noise. But when the learning set contains more than 30% of noise, the system cannot recognize clean data. When the percentage of noise is over 30%, the sequences do not have a consistent

[5] percentage of errors

[6] Tests 1, 2 and 3 have been executed only once, thus the minimal and average percentages, as well as the standard deviation, have no meaning here. They only do for tests 4, 5 and 6, executed on 20 random partitions of the data set.

[7] Standard deviation

[8] Random noise is not the same as problems that are to face with real low level features, but it gives us an estimation of our method's robustness.

structure anymore and the HMMs memorize sequence more than learn a generalized model. In fact, the number of classes increases extensively (Fig. 6). However, the system is robust up to 30% of noise.

Table 4. The tests executed to measure the robustness to noise

		Learning	
		Clean	Noisy
Validation	Clean	Test CC	Test NC
	Noisy	Test CN	Test NN

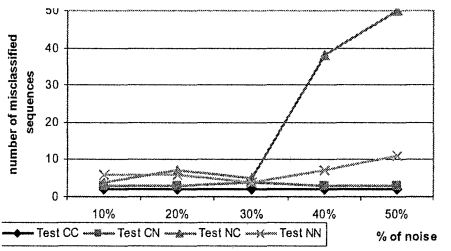

Fig. 1. The number of errors on validation sequences with 10% to 50% of noise.

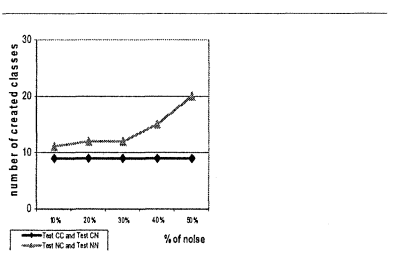

Fig. 2. The number of classes created with 10% to 50% of noise.

4.3. Tuning the Threshold

Tuning the threshold is a way to influence the number of created classes. When the threshold grows, the number of classes grows (Fig. 8). These classes are more specialized and bring out slight differences between scenarios. When the threshold decreases, the added classes are scarcer and each class is more general and explains different sequences. We measured the variations of the number of errors and the number of classes with the variation of the threshold. The results below (Fig. 7 and Fig. 8) admit to choose the value of the threshold depending on error rate and number of classes.

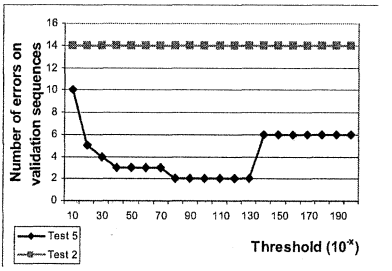

Fig. 3. The number of errors on validation sequences when the threshold varies.

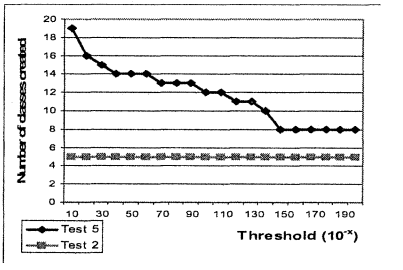

Fig. 4. The number of classes created when the threshold varies.

5. Conclusions and Future Work

We proposed a method capable to automatically acquire the needed number of models representing scenarios of a given training set. The strong point of our method is its degree of automatism. Not only the number of HMMs and the sequences attributed to them are determined automatically. The number of states of each HMM does not need to be specified by hand either. It is easy to add more data and to learn new scenarios.

This approach can be categorized as a *semi-supervised* method. Model creation is completely automatic, the only parameter being a threshold controlling the number of models to create. However, labeling of sequences, that is to say the attribution of a scenario to each individual from a video, is supervised as it is done by a human operator. This step consists in attaching symbols, semantics, to numeric data. In order to avoid that, we could add an "interactive feedback".

In the end we obtained a complete and automatic method, comprising all the necessary steps, from XML conversion into observations to recognition and classification of sequences, while passing by automatic learning of HMMs. Under the best conditions, our tests resulted in 2 errors on 102 validation sequences, that is to say an error rate of 1.96%.

References

[1] R. Bodor, B. Jackson and N. Papanikolopoulos, "Vision-Based Human Tracking and Activity Recognition", *in* 'Proc. of the 11th Med. Conf. on Control and Automation', 2003.

[2] J.L. Crowley, J. Coutaz, G. Rey and P. Reigner, "Perceptual components for context awareness", *in* 'International conference on ubiquitous computing', 2002.

[3] Y.A. Ivanov and A.F. Bobick, "Recognition of Visual Activities and Interactions by Stochastic Parsing", *in* 'IEEE Trans. on Pattern Analysis and Machine Intelligence', 2000.

[4] P. Lichodzijewski, A. Zincir-Heywood and M. Heywood, "Dynamic Intrusion Detection Using Self Organizing Maps", *in* '14th ACITSS', 2002.

[5] G. Ma and X. Lin, "Typical Sequences Extraction and Recognition", *in*, Computer Vision in HCI: ECCV 2004 Workshop on HCI, Prague, Czech Republic, 2004.

[6] D. Mahajan, N. Kwatra, S. Jain, P. Kalra and S. Banerjee, "A Framework for Activity Recognition and Detection of Unusual Activities", *in* 'ICCVGIP', 2004.

[7] I. McCowan, D. Gatica-Perez, S. Bengio, G. Lathoud, M. Barnard and D. Zhang, "Automatic Analysis of Multimodal Group Actions in Meetings", *in* 'IEEE Trans. PAMI' , 2005.

[8] D. Minnen, I. Essa and T. Starner, "Expectation grammars: leveraging high-level expectations for activity recognition", *in* 'CVPR', 2003.

[9] D. Moore and I. Essa, "Recognizing multitasked activities using stochastic context-free grammar", *in* 'CVPR Workshop on Models vs Exemplars in Computer Vision', 2001.

[10] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *in* A. Waibel & K.-F. Lee, ed., 'Readings in Speech Recognition', 1990.

[11] G. Schwarz, "Estimating the dimension of a model", *in* 'Ann. Statist.', pp. 461-464, 1978.

[12] Y. Shi, Y. Huang, D. Minnen, A.F. Bobick and I.A. Essa, "Propagation Networks for Recognition of Partially Ordered Sequential Action", *in* 'CVPR (2)', 2004.
[13] N. Vaswani, A.R. Chowdhury and R. Chellappa, "Activity recognition using the dynamics of the configuration of interacting objects", *in* 'CVPR', 2003.
[14] D. Zhang, D. Gatica-Perez, S. Bengio, I. McCowan and G. Lathoud, "Multimodal Group Action Clustering in Meetings", *in* 'ACM 2nd Intern. Workshop on Video Surveillance and Sensor Networks in conj. with 12th ACM International Conference on Multimedia', 2004.
[15] H. Zhong, J. Shi and M. Visontai, "Detecting Unusual Activity in Video", 'CVPR', 2004.
[16] M. Zimmermann and H. Bunke, "Hidden Markov Model Length Optimization for Handwriting Recognition Systems", *in* 'IWFHR', 2002.