GRAPHICAL MODELS FOR MULTI-MODAL AUTOMATIC VIDEO EDITING IN MEETINGS

Benedikt Hörnler and Dejan Arsić and Björn Schuller and Gerhard Rigoll

Technische Universität München
Institute for Human-Machine-Communication
80290 Munich, Germany
{hoernler,arsic,schuller,rigoll}@mmk.ei.tum.de

ABSTRACT

In this work we present a multi-modal video editing system for meetings, which uses graphical models for the segmentation and classification of the video modes. The task of video editing is about selecting the camera, that represents the meeting in the best way out of various available cameras. Therefore a new training structure for graphical models was developed. This is necessary for the learning of boundaries combined with the video mode itself. All developed and known decoding structures can be easily connected for an EM-training to our training structure. The achieved results of the system are state of the art.

***Index Terms*—** Machine Learning, Human-Machine Interaction, Multi cameras, Meeting Analysis, Multi-modal Low Level Features

1. INTRODUCTION

In this paper, we address the multi-modal problem of selecting one camera view from various available cameras in a meeting room. The problem derives from the fact that video conferences are getting more popular at the same time as it is getting more and more common that meeting rooms are equipped with various recording devices. This points out the two main usage scenarios of the system: On-line video conferences [1] and browsing of past meetings [2].
Most of the current video conference systems transmit all available camera streams to each participant and display them all. The main difference between the systems is the way they present the video data. Normally low-budget systems show different small windows on a computer monitor, while high end systems use multiple large screens for that purpose. These approaches are limited to a few participants, because the video of the individual participant is getting smaller if more persons are connected. Few other systems are using the audio channel for selecting the stream and so they transmit only one video to all participants, which saves bandwidth. The main drawback of these approaches are, that interesting non-verbal gestures are lost, because they cannot be detected from the acoustic channel or cannot be recognized in small videos. This shows the multi-modal nature of meetings, thus it can be very important to show people who currently do not speak. For example, in talk-shows professional editors follow this rule and show facial reactions and gestures from other participants [3].
The other scenario for automatic video editing is in the field of meeting recordings [4]. The capturing of meetings is very common, but an unsolved problem is, how to access the recorded data in a useful way. For this purpose meeting browsers [2, 5] have been developed. The browser displays the content of the meeting, but it is still very hard to watch several video streams at the same time. Normally one will miss some important visual hints, which one will recognize if one attends personally the meeting. Therefore, research in multi-modal approaches to solve problems in meeting scenarios is performed.
One of the emerging tools for pattern recognition in meetings are graphical models [6]. They help to describe complex problems, as for example video editing in meetings, in an intuitive and visually appealing way. The possibility of integrating context and semantic information on different levels of these models make them interesting for the domain of meeting analysis. A big advantage is the fact that the model can be easily adopted to a new idea and the algorithms, which are performing the calculations of the probabilities behind the graphical representation are not influenced by these changes. Therefore we applied graphical models to the pattern recognition problem of video editing in meetings.
For both scenarios, the problem of selecting a video mode is the same: for each time frame we need to choose a single video mode which represents best what is happening in the meeting. The video stream which represents the video mode is transmitted to the other participants in the case of a video conference or stored for later browsing. Video editing

This work was supported by the European IST Programme Project FP6-033812 (Augmented Multi-party Interaction with Distant Access). This paper only reflects the authors' views and funding agencies are not liable for any use that may be made of the information contained herein.

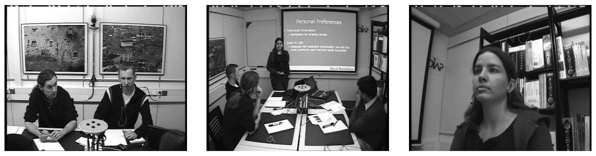
will help you to catch up quickly after a missed meeting, will make it easier to attend the meeting by using small screens, for example on a cell phone, will reduce the required bandwidth for mobile communications and will save storage capacities.

In [7] a rule based system was introduced for video editing in lecture halls and meeting rooms. The system had two operation modes; one for a video conference and the second for stored meetings which were performed in the past. The algorithm of the system was taking into account some technical aspects, as well as aesthetic ones. For example, the duration of showing the same camera was the most important aesthetic aspect. From the technical point of view, it was important: Who was speaking? How long was someone speaking? Who was moving the head or hands? All this information was combined to a weight for each camera at each frame and the camera with the highest weight was selected. The system is also using virtual cameras for creating more camera changes, if for a long time no change happens. The main drawback of this system was that the selected camera was highly depending on the current speaker and important non verbals, as shaking the head or nodding, were not shown in the edited video. In [8] we introduced an approach to video editing for smart meeting rooms, which is based on some rules. Therefore, we extracted different acoustic, visual, and semantic features based on time frames from the meeting data. We investigated the influence of different feature types on the task of the automatic video editing. The first tests showed that frame based video editing by a rule based system can lead to unintentional twitches. Therefore, we derived additional features by windowing a range of subsequent frames for the audio and visual features which leads to better results specially for acoustic features. A third evaluation was done by performing late fusion of the different features. The outcome was, that the combination of audio and video information leads to better results then single modalities.

In [9] a video editing system based on Hidden Markov Models (HMM) [10] was introduced. For each video mode a HHM is trained by EM algorithm [11] and the best sequence of HHMs is derived by Viterbi decoding [12]. This system is state of the art for the used AMI database and achieves frame error rates of about 47.9%.

In this work we use a newly developed graphical model structure for the training and the decoding. By using the training structure it is possible to learn shot boundaries. Moreover, different combinations of features and settings for the models have been evaluated.

The next section gives an overview of the data set and the annotation, which are needed for the training of the models. Section 3 describes the used acoustic, visual and semantic features. The pattern recognition models used in this work are presented in Section 4. In Section 5 the results from the experiments are shown and finally the conclusion is drawn in Section 6.



(a) Camera Left (b) Camera Center (c) Camera Closeup

Fig. 1. Sample shots of three cameras from the IDIAP smart meeting room: left camera (L), centre view (O) of the room and a closeup (C_1) of participant one.

2. DATA SET

The AMI corpus [13], which is publicly available, is used for this work. A subset of 24 meetings with a duration of five minutes each, was created and four participants are always located somewhere in the IDIAP smart meeting room [14] during these meetings. The meeting room is equipped with seven cameras, 22 microphones, a projector screen and a whiteboard. This work uses only four close talking microphones and does not take into account the installed microphone arrays for the far field recordings. For the video capturing, seven cameras are installed: one closeup camera for each participant ($C_1 - C_4$). An overview camera (O) that records the table, the whiteboard and the projector screen. Two additional cameras are located at the left (L) and right (R) wall and are capturing two participants and the half table in front of them. Three example shots of these cameras are shown in figure 1.

2.1. Annotation

Annotation of the whole data set is needed for two reasons: First, it is necessary for the training of the graphical models and second the ground truth is used for the evaluation of the results. Thus 24 five minutes meetings have been annotated. The annotators have to decide which of the seven video modes represents the meeting best. This leads to seven different video modes which contain four closeups views, one for each participant, a left and a right view, which records two people at one side of the table, and a centre view, which covers the whole meeting room. In figure 2 three sketches are shown of the available cameras. In the future additional information, as the recorded slides from the projector, are planned to be annotated and insert into the video.

The task of annotating video modes is very subjective as the low average of inter-annotator agreement ($\kappa = 0.3$) shows. It is highly depending on the taste of the different annotators. Therefore, only one annotator labels the whole corpus to achieve a consistent annotation. Moreover, the inter-annotator agreement of a single annotator, doing the same meeting twice, raises to $\kappa = 0.6$. Even though the shot

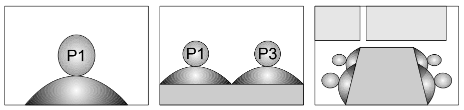


Fig. 2. Sketches of three available video modes in the IDIAP smart meeting room: closeup (C_1) of participant one, left camera (L) and centre view (O).

boundaries are on a frame base and no gray array is allowed around the shot change.

3. FEATURES

In this work, three different modalities of features are used: acoustic, visual and semantic. The first two modalities are low level features and are derived directly from the audio- and video streams. As the acoustic feature the well known mel frequency cepstral coefficients (MFCC) [15] are used and as visual feature we used global motions and skinblobs. The global motions have been successfully applied to various meeting tasks [16, 17] and can be calculated in real-time. In [18] various approaches of face detection are deeply investigated and one of these is a skin color look-up-table. This approach has been used in this work. The semantic features contain more related information of the occurrences in the ongoing meeting. In the following paragraphs the semantic features are described.

3.1. Semantic Features

Not only acoustic and visual low level features are applied to the detection task, but also features that contain more semantic information are used. These features are interesting because of the close relation between what a person or the group is doing and which camera is important. The features, which have been applied are group action, person action and person speaking.
The group action has been deeply investigated in the research community over the last couple of years [19, 20, 21]. The systems are working directly on audio and video streams and achieve reliable results, but they are currently not real time capable. The meeting is segmented into a sequence of labels like monologue participant one to four, discussion, presentation, whiteboard and note taking.
Moreover, a person action detection system has been developed [16, 17]. These systems create a sequence of actions for each of the participants, thus four features for each time frame are available. The labels used, are similar to the group actions but contain some more classes: sitting down, standing up, nodding, shaking the head, writing, pointing, using a computer, giving a presentation, writing on the white-board, manipulation of an important item and idle. Idle for example is used if the person is speaking or listening to the meeting. The classes nodding or shaking should help to find points in the meeting where a person should be shown even though he is not speaking.
The last semantic feature which is currently used is the person speaking. It is a four dimensional vector which contains binary information for each participant and each time frame. The bits are set to one if a person is speaking.

4. CAMERA SELECTION MODELS

For typical static pattern recognition problems, we are looking for the class k with the parameter set λ_k which most likely produce the observation $\vec{o}$ [10]:

$$k^* = \underset{k \in K}{\operatorname{argmax}}\, p(\vec{o}_1, \dots, \vec{o}_T \mid \lambda_k). \quad (1)$$

In this section, we introduce different types of graphical models (GM) which perform an automatic segmentation and classification at the same time. For this, a sequence of video modes k has to be aligned to an observation $\vec{o}$, so that the most likely class boundaries and classes can be detected. This can be written as

$$\{k_1^*, \dots, k_T^*\} = \underset{k_1, \dots, k_T}{\operatorname{argmax}}\, p(\vec{o}_1, \dots, \vec{o}_T \mid \lambda_{k_1}, \dots, \lambda_{k_T}), \quad (2)$$

which is adapted from (1).
The basics for the segmentation are described in [6], where the models are applied to speech recognition and language processing. The model is adopted from [22] and uses the properties of it. The message passing [23] in the junction tree is modified for the segmentation in a way, that all sums are replaced by maximizations. This means that only the most likely configuration for each vertex in the GM is passed on and the approach is similar to Viterbi decoding [12].
In this work, a general structure for segmentation based on [6] is described and used for multi-modal fusion of three different feature domains. All the developed structures are GMs, therefore the models, as well as the combination with other GMs, can be used for training and decoding without any modification of known algorithms for GMs.

4.1. Linear Training Structure

The training of GMs can be split into two steps. The first one, where the model structure is defined and the types of the probability density functions of each vertex is defined, and the second, where the parameters of the probability density functions are learned. The first step is normally done by expert knowledge, but it is also possible to learn the structure directly from the data [24]. These two approaches are possible for the second step as well, but in the field of pattern recognition it is common that the parameters are learned from training data. Mostly used is the maximum likelihood learning (ML) [25] and the expectation maximization algorithm

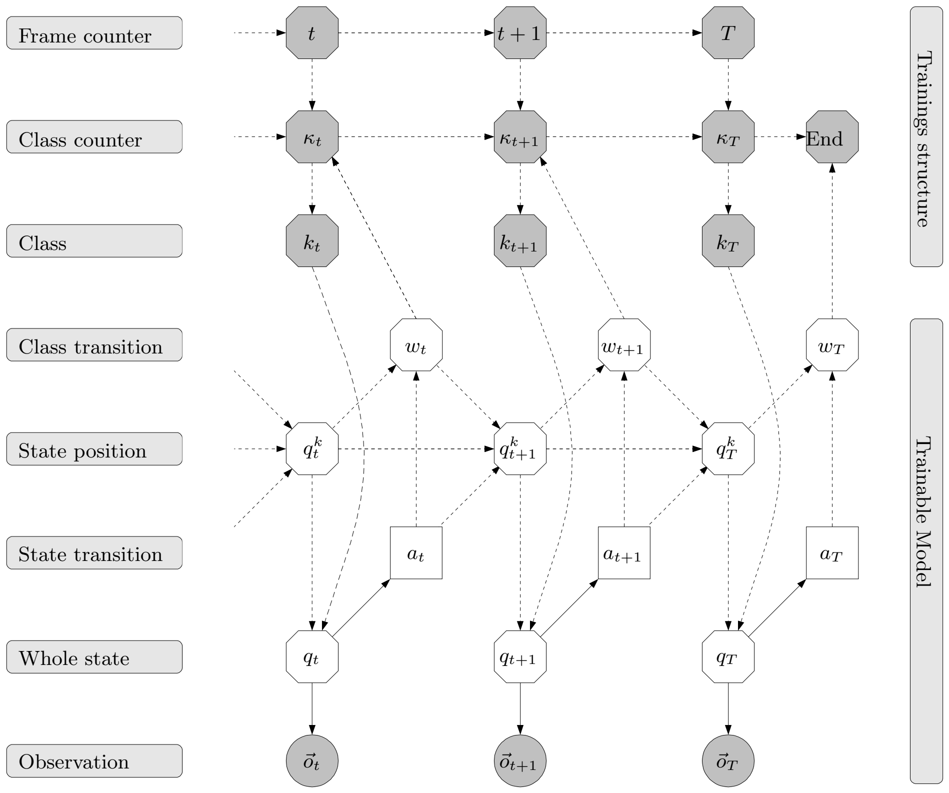


Fig. 3. Training structure for the training of segment boundaries. The upper three layers can be connected with every GM. The model, which should be trained, (for example we use the model from figure 4) does not need to be modified, because of the same interface of the training structure and the decoding structure.

(EM) [11], because of the fact that not for all vertices in the model training data is available. By combining these two methods, it is possible to learn the parameters of the various structures directly, which is an unsupervised learning. The drawback is that a decomposition of the class vertex occurs and the best relation between observation and class is not learned.

Therefore, it is necessary to observe the class k_t and to use a supervised learning for the GMs. The consequence of this change is, that the connection from class transition to class is removed, because the class is a deterministic vertex in the case of training the model. The vertices below can not detect whether a class transition occurs after removing the connection, therefore we have to add a connection from the observed class k_t and the previous k_{t-1} to the vertex w_t. This means that the model has to be changed dramatically for the training. These changes of the model structure can be avoided by using a common training structure, which for example is shown in figure 3. This structure can be used with a lot of structures for integrated segmentation and classification and the decoding structure has to be modified only slightly. Another class oriented training structure is described in [22]. The drawback of this structure compared to ours is, that the class boundaries are not predefined, so the observation sequence $\vec{o}$ is unsupervised aligned to class sequence. The training is in this case partly supervised, which is common for automatic speech recognition, because only the word alignment is known and the phone boundaries are unknown. For many applications, for example automatic video editing, it is necessary to learn the correct class boundaries as well and not only the correct sequence of classes, as common in speech recognition.

Figure 3 shows the training structure combined with a linear decoding structure from section 4.2, which allows supervised and unsupervised learning of the class boundaries. The upper three layers in the figure are the training structure, and the layers underneath represent the decoding structure of the linear model. The learning structure contains three observed and deterministic vertices: a class k_t, a class counter κ_t, and the frame counter t. They are shortly described here:

Frame counter t is necessary for the supervised learning. At the prologue, it is set to one and is incremented at each time frame. The model knows implicitly through the frame counter in which time frame it is located.

Class counter κ_t is directly depending on the frame counter t and is also set to one in the first time frame $f(\kappa_1 = 1 \,|\, t = 1)$. For all other time frames the implementation is $f(\kappa_t \,|\, \kappa_{t-1}, t, w_t)$, so it is depending on the training data. The class counter is incremented if a class transition occurs in the training data $f(\kappa_t = i+1 \,|\, \kappa_{t-1} = i, t, w_t)$, and for the case no class change happens the class counter is left unchanged $f(\kappa_t = i \,|\, \kappa_{t-1} = i, t, w_t)$. The class counter counts the class transitions and makes sure that the class boundaries are at the right time frames. It has no knowledge about the current class. The deterministic connection to the class transition vertex w_t makes it possible that the class transition is set to one, if the class counter is incremented. Thereby it is assured that all vertices below know that a class transition occurs and that no vertex below has to be changed. At the epilogue an additional vertex "End" is inserted, so that all class transitions have occurred and that the training sequence ends in the final state of the last class.

Class k_t is only depending on the class counter κ_t and is a deterministic vertex with information of the class sequence from the training data. There is no knowledge about the class boundaries stored in the class k_t. The class counter κ_t selects the class in the class vertex k_t. If a class transition occurs, the class counter is incremented and selects the next class. By this setting, the behavior of the connection from the class k_t to the whole state q_t is similar to the decoding structure, so it is not necessary to modify the model which should be trained.

The training structure itself is again a GM, therefore it can be connected to every other GM. The interface of the structure is designed in a way, that few modifications are necessary in the decoding model. The modifications are: the edge from the class transition w_t to the class k_t is removed, and an additional edge from class transition w_t to class counter κ_t is inserted, because of the additional edge no vertex in the model has to be modified. During training all vertices with probability functions are trained, and the deterministic vertices are left unchanged.

If the frame counter t is removed the training structure does not know when a class transition occurs, so the training only has knowledge about the class sequence. The training of the class boundaries would be unsupervised in the case. By adding an additional probability vertex with a connection from k_t to it and a connection from it to k_{t+1} it is possible to train class bigram probabilities.

4.2. Linear Decoding Structure

Figure 4 illustrates the simplest model for integrated segmentation and classification which is taken from [6]. Each time frame consists of six vertices which are described shortly as:

The observation vector $\vec{o}_t$ models the probability of the observation depending on the current state $p(\vec{o}_t \,|\, q_t)$. In this case the random variable has Gaussian distribution.

Whole state q_t contains all possible states of the model and is a deterministic function $f(q_t \,|\, q_t^k, k_t)$. The total number of states in the vertex is depending on the number of states per class N and the number of classes K, and has the size $N \times K$. It is possible to share some states, which means that at least two classes refer to the same entry in this vertex.

The state transition probability a_t models for each state q_t the transition probability $p(a_t \,|\, q_t)$. In the case of a linear model it is only possible to stay in the current state or to move to the next state.

The pointer to the current state q_t^k always use the first state, independent of the the class n the first frame $t = 1$. For all other time frames the state pointer is a deterministic function $f(q_t^k \,|\, a_{t-1}, q_{t-1}^k, w_{t-1})$. If a class transition occurs ($w_{t-1} = 1$) always the first state applies. When no class transition emerges, then the state transition is checked and if $a_t = 1$ then the state pointer q_{t-1}^k is incremented. For the case $a_t = 0$ the state position remains unchanged.

The class transition w_t is again a deterministic function $f(w_t \,|\, q_t^k, a_t)$. Only if a state transition occurs and the state pointer points to the last state, a class transition is possible. The observed child of the class transition assures that in the last chunk $f(w_t = 1 \,|\, q_T^k, a_T)$. Thus an observation sequence always ends in the last state of a class.

The current class k_t is depending on the class transition w_t and the previous class k_{t-1}. This vertex has switching parent functionalities and switches the conditional probability function from a deterministic function to a bigram language model probability. If there is no class transition, the previous class is assumed. For the case a class transition occurs, the next class is set by using a bigram class distribution. In the first frame the class is assigned by using a unigram class distribution.

The joint probability of the graph is

$$
\begin{aligned}
&p(\vec{o}_1, \ldots, \vec{o}_T, q_1, \ldots, q_T, a_1, \ldots a_T, q_t^k, \ldots, q_T^k,\\
&w_1, \ldots w_T, k_1, \ldots, k_T) =\\
&p(\vec{o}_1 \,|\, q_1)\; f(w_1 \,|\, q_1^k, a_1)\; p(a_1 \,|\, q_1)\; f(q_1 \,|\, q_1^k, k_1)\; f(q_1^k)\; p(k_1)\\
&\prod_{t=2}^{T-1} p(\vec{o}_t \,|\, q_t)\; f(w_t \,|\, q_t^k, a_t)\; p(a_t \,|\, q_t)\; f(q_t \,|\, q_t^k, k_t)\\
&f(q_t^k \,|\, a_{t-1}, q_{t-1}^k, w_{t-1})\; p(k_t \,|\, k_{t-1}, w_{t-1})\\
&p(\vec{o}_T \,|\, q_T)\; f(w_T = 1 \,|\, q_T^k, a_T)\; p(a_T \,|\, q_T)\; f(q_T \,|\, q_T^k, k_T)\\
&f(q_T^k \,|\, a_{T-1}, q_{T-1}^k, w_{T-1})\; p(k_T \,|\, k_{T-1}, w_{T-1}).
\end{aligned}
\tag{3}
$$

The third line describes the prologue, the fourth and fifth line the chunk and the last two the epilogue. After some steps of

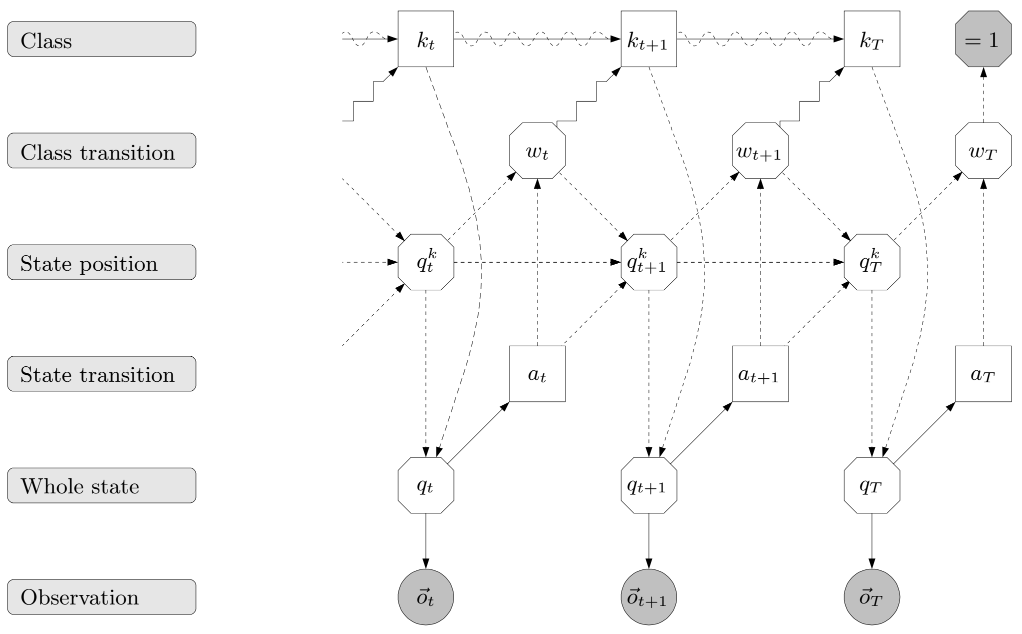


Fig. 4. Linear decoding structure for integrated segmentation and classification. Each time frame consists of six vertices: An observation vector $\vec{o}_t$, all possible states q_t, a state transition probability a_t, a pointer to the current state q_t^k, a class transition probability w_t, and the current class k_t. The model separates the states from the current observation and therefore the current observation from the class. A common pool of states is used to model the observation. A state change is performed by using the state transition probability. The state position vertex points to the current state and this finally models the class. Only in the last state a class transition is possible and thus the segmentation is possible. A connection between the class and the current states is only possible through the class transition vertex. The notation of all following GMs is based on [26] with the extension of deterministic components from [6].

conversion, the joint probability is reduced to

$$
\begin{aligned}
& p(\vec{o}_1,\dots,\vec{o}_T,q_1,\dots,q_T,a_1,\dots a_T,q_1^k,\dots,q_T^k,\\
& \quad w_1,\dots w_T,k_1,\dots,k_T) = \\
& \quad \prod_{t=1}^{T} p(\vec{o}_t \,|\, q_t)\; f(q_t \,|\, q_t^k, k_t)\; p(a_t \,|\, q_t) \\
& \quad f(q_1^k)\; p(k_1) \prod_{t=2}^{T} f(q_t^k \,|\, a_{t-1}, q_{t-1}^k, w_{t-1})\; p(k_t \,|\, k_{t-1}, w_{t-1}) \\
& \quad f(w_T = 1 \,|\, q_T^k, a_T) \prod_{t=1}^{T-1} f(w_t \,|\, q_t^k, a_t)
\end{aligned}
$$

but the prologue, the chunk, and the epilogue are not visible in this form.
This model can be represented as a linear HMM with $N \times K$ states, if no states are shared and the emission is modeled as a Gaussian distribution or a GMM. An additional edge between the class k_t and the class transition w_t makes it possible to have a different number of states N_k for each class. This leads to a new number of states $\sum_{k=1}^{K} N_k$ for the model if no states are shared. The benefit of this additional edge is that it provides an easy option to model short pauses and moments of silence for automatic speech recognition as well as video editing.

5. EXPERIMENTS

Two experiments are performed to evaluate our proposed system: For the first task we know the true boundaries of each segment in the meeting, so the experiment is about assigning the correct video mode for each segment. This could be done, because we have annotated the full subset of the meeting data.
The real experiment for the video editing system is the second, because the boundaries are also unknown and so they have to be found as well. Therefore, the second task contains a segmentation and classification problem, which is the real application of the system. The subset contains 24 five-minute meetings with seven camera streams and four audio channels. To four meetings the same subjects are participating, so we perform a six-fold cross-validation, which is person disjoint. We did the experiments for the scenario, that seven video modes (all four persons, left, right, and center camera) were available.
Three different types of error measurement are used in this worke. The first is the recognition rate (RR), which is used for the classification task with known shot boundaries. As a

Table 1. Evaluation of different model configurations. For the evaluation a combination of global motion and acoustic features were used. In the table S describes the number of states per class and M is for the number of Gaussian mixture.

Model config.	RR (in %)	FER (in %)	AER (in %)
S=3, M=1	36.9	63.1	10.1
S=3, M=2	42.3	58.7	9.6
S=5, M=2	46.6	53.3	15.5
S=5, M=3	26.4	73.6	8.6
S=10, M=2	41.6	58.4	21.8

second measurement we use the frame error rate (FER) for the joint segmentation and classification task. Each frame is compared with the ground-truth and all wrong classified frames are counted. This measurement is the most important for this task, because it takes into account the length of the segments. The last measurement is the action error rate (AER). For this error measurement it is necessary to count the insertions (Ins), the deletions (Del), and the substitutions (Sub).

$$AER = \frac{\text{Ins} + \text{Del} + \text{Sub}}{\text{Annotated video modes}} \quad (4)$$

The results in table 1 are conducted from an evaluation where different model parameters are tested. The first column in the table shows the results which are achieved for the classification task only. The best model for it is the one with five states and two Gaussian mixtures with a recognition rate of 46.6%. The more complex models do not achieve this results. Therefore the recognition is highly depending on the number of available training data.

The results for the combined task of segmentation and classification are shown in column two and three in table 1. All models are trained and tested on the same combination of global-motion and acoustic features. The results show that the model with five states and two Gaussian mixtures achieves the best frame error rate with 53.3%, therefore it is the best model for video editing. The best action error rate of 8.6% achieves a linear one with five states and three Gaussian mixtures. The low action error rates of most of the models point out that the hardest task is to find the correct boundaries of the segments. In figure 5 a short output of the system is shown in a symbolic way. Each picture represents the selected video mode with various duration and for the whole meeting the number of shot changes is much higher then in this short example output.

6. CONCLUSION

In this work we showed how to extend graphical models for the combined task of segmentation and classification. The developed structure for the combined task can be universally used for known graphical structures. This also applies for the training structure which has been developed in this work. All known structures have been developed for automatic speech recognition, and so the class boundaries of phone are not trained. This is no problem for speech recognition, where the boundaries of phones are not known, but for the video editing is is mandatory to train the boundaries of the video modes. Therefore a new training structure was developed which can be applied for both types of problems. The training structure and the decoding structures can also be applied for other pattern recognition tasks, which require the combined segmentation and classification. The best model, with five states and two Gaussian mixtures, achieves a FER of 53.3% and an AER15.5%. The major problem of all models is finding the correct boundaries of the video modes.

For the future we plan to add more semantic information about the participants, for example the level of activity or dominance. The problem of the segmentation could be solved by using additional features, such as topic changes, which help to segment the meeting in a more accurate way. As further improvement it is planned to integrate an additional class which adds the slides. Finally more complex graphical models, as left-right-, ergodic-, multi-stream-models, should be evaluated because the most complex model used performs best.

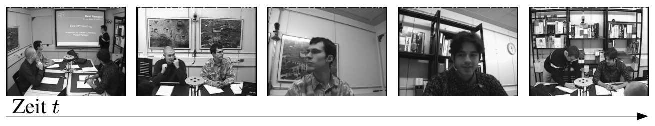


Fig. 5. An example for a video sequence, which is represented by single frames of each selected video mode. In the video each shot can have a different duration.

7. REFERENCES

[1] S. Sabri and B. Prasada, "Video conferencing systems," *Proceedings of the IEEE*, vol. 73, no. 4, pp. 671 – 688, 1985.

[2] P. Wellner, M. Flynn, and M. Guillemot, "Browsing recorded meetings with ferret," in *Proceedings of the 1st Joint Workshop on MLMI*, S. Renals and S. Bengio, Eds. 2004, Springer Verlag.

[3] Beller Hans, *Handbuch der Filmmontage - Praxis und Prinzipien des Filmschnitts*, TR-Verlagsunion, München, 5. edition edition, 2005.

[4] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters, "The icsi meeting corpus," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.

[5] P. Wellner, M. Flynn, S. Tucker, and S. Whittaker, "A meeting browser evaluation test," in *CHI '05 extended abstracts on Human factors in computing systems*, New York, NY, USA, 2005, pp. 2021–2024, ACM Press.

[6] J. Bilmes and C. Bartels, "Graphical model architectures for speech recognition," *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 89 – 100, 2005.

[7] S. Sumec, "Multi camera automatic video editing," in *Proceedings of the ICCVG*. 2004, pp. 935–945, Kluwer Verlag.

[8] M. Al-Hames, B. Hörnler, C. Scheuermann, and G. Rigoll, "Using audio, visual, and lexical features in a multi-modal virtual meeting director," in *Proceedings of the 3rd Joint Workshop on MLMI*. 2006, Springer Verlag.

[9] M. Al-Hames, B. Hörnler, R. Müller, J. Schenk, and G. Rigoll, "Automatic multi-modal meeting camera selection for video-conferences and meeting browsing," in *Proceedings of the 8th International Conference on Multimedia and Expo (ICME)*, 2007.

[10] L.R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–285, 1989.

[11] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society B*, vol. 39, no. 1, pp. 1–38, 1977.

[12] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260 – 269, 1977.

[13] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, I. McCowan, W. Post, D. Reidsma, and P. Wellner, "The AMI meeting corpus: A pre-announcement," in *Proceedings of the 2nd Joint Workshop on MLMI*. 2006, pp. 28–39, Springer-Verlag.

[14] D. Moore, "The IDIAP smart meeting room," Technical Report 07, IDIAP, 2002.

[15] Z. Fang, Z. Guoliang, and S. Zhanjiang, "Comparison of different implementations of MFCC," *Journal of Computer Science and Technology*, vol. 16, no. 6, pp. 582–589, 2001.

[16] M. Zobl, F. Wallhoff, and G. Rigoll, "Action recognition in meeting scenarios using global motion features," in *Proceedings of the 4th IEEE International Workshop on PETS-ICVS*, J. Ferryman, Ed., 2003, pp. 32–36.

[17] F. Wallhoff, M. Zobl, and G. Rigoll, "Action segmentation and recognition in meeting room scenarios," in *Proceedings of the 11th ICIP*, 2004.

[18] M.-H. Yang, D.J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Transasctions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34–58, 2002.

[19] M. Al-Hames, A. Dielmann, D. Gatica-Perez, S. Reiter, S. Renals, G. Rigoll, and D. Zhang, "Multimodal integration for meeting group action segmentation and recognition," in *Proceedings of the 2nd Joint Workshop on MLMI*, 2006.

[20] S. Reiter, B. Schuller, and G. Rigoll, "Hidden conditional random fields for meeting segmentation," in *Proceedings of the 8th International Conference on Multimedia and Expo (ICME)*, 2007.

[21] D. Zhang, D. Gatica-Perez, S. Bengio, I. McCowan, and G. Lathoud, "Modeling individual and group actions in meetings: a two-layer HMM framework," in *Proceedings of the Second IEEE Workshop on Event Mining: Detection and Recognition of Events in Video, in Association with CVPR*, 2004.

[22] J. Bilmes and G. Zweig, "The graphical model toolkit: An open source software system for speech and time-series processing," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002.

[23] F. Jensen, S. Lauritzen, and K. Olesen, "Bayesian updating in causal probabilistic networks by local computations," *Computational Statistics Quaterly*, vol. 4, pp. 269–282, 1990.

[24] D. Heckerman, "A tutorial on learning with bayesian networks," in *Learning in Graphical Models*, M.I. Jordan, Ed. MIT Press, 2001.

[25] Z. Ghahramani, "Learning dynamic bayesian networks," in *Adaptive Processing of Sequences and Data Structures*, C.L. Giles and M. Gori, Eds., Lecture Notes in Artificial Intelligence, pp. 168–197. Springer-Verlag, 1998.

[26] K. Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning*, Ph.D. thesis, University of California, Berkeley, 2002.