Automatic Annotation of Formula 1 Races for Content-Based Video Retrieval

V. Mihajlovic, M. Petkovic
Compute Science Department
University of Twente
The Netherlands
{vojkan, milan}@cs.uwente.nl

Abstract

Content-based video retrieval is emerging as an important part in the process of utilization of various multimedia documents. In this report we present a novel system for the automatic indexing and content-based retrieval of multimedia documents. We chose the domain of Formula 1 sport videos because the manual annotation of Formula 1 races is complicated and time consuming. Our system uses multi-modal clues, obtained from three different multimedia components: audio, video, and superimposed text. The audio and video feature extraction subsystems are developed to extract important parameters from multimedia documents. We also performed text detection and recognition to extract some semantic information superimposed in the Formula 1 race video. To unify the audio and video clues we employed dynamic Bayesian networks. Many experiments that we carried out are also presented, as well as the results and conclusions drawn from them.

1. Introduction

As human hunger for information grows, billions of different resources for contentment of human needs rise in the world. They evolved from simple mails and newspapers to radio broadcasts, TV news and finally led to global information storage and retrieval system called the Internet. Internet enables various data to be stored on different systems, as well as in different formats and at diverse places around the world. However, to find interesting information in that enormous mass of data become very difficult and arduous. People neither have time nor the energy to browse large amounts of data to find what they need. Therefore, inventing efficient techniques for information retrieval in order to enable satisfaction of human needs is an important and indispensable task.

Another interesting issue that needs to be exerted is that modern people want to have information in a form that is most appropriate for them. Thus, everyone wants to find information fast and want to receive it as a text, but in a visual or audio form. This is mostly because the amount of information humans can receive and handle through audio/visual channel is much greater than from newspapers columns, or solely audio.

One of the evolving areas that would certainly occupy computer scientists in the next decade is concerned with development of software tools, which will be able to make conclusions based on information gathered from various sources. An interesting approach to the solution of this problem would be to simulate reasoning process of humans, based on their ability to sense the environment in multiple ways and to integrate these sensed information in one global picture of the environment. Unlike humans, computers are at the present mostly restricted to three main types of sources. These are visual, textual and audio sources.

These three sources can be regarded as building blocks of a growing area in the computer science, termed multimedia. Storage places for multimedia documents are becoming very popular throughout the world, and they grow larger and larger every day. The data management in such multimedia libraries (databases) is shown to be a complex and demanding task that needs to be improved in order to fulfil the rising human requirements. Most of these requirements are related to the process of querying a multimedia database, i.e. retrieval of specific parts of a multimedia document that are of interest to a user. However, traditional database management systems do not provide enough facilities for processing and retrieving of multimedia contents. Therefore, we must address new techniques for characterizing and querying the multimedia database. Some of these techniques will be presented in this report.

Storage facilities would make enormous growth of multimedia files stored in different databases around the world really usable only with new multimedia browsing and indexing techniques. We could try to develop one general multimedia retrieval system, but at the very beginning we will conclude that this would be an impossible problem to deal with. This is mostly due to the fact that various content can be stored in these multimedia files, such as: movies, commercials, news, music

videos, sport videos, etc. These documents cannot be uniquely described and processed with same algorithms. Even if we find some kind of the universal solution to this problem it would be full of numerous errors, and approximations. Therefore, to develop a powerful and efficient retrieval tool we must discard the generality attribute, and choose one specific domain.

1.1. Why Formula 1?

For our research we decide to focus on sport videos, because sport games are usually quite long, and there are frequently a small number of interesting parts of the game. The extraction of these scenes would enable us to provide only interesting moments to the user. This would cut the amount of information presented to the user and this information would be much more relevant.

However, even sport videos can be significantly different. The duration of the game, camera motion, and properties of the game are only some of the distinct categories from multimedia documents of various sport events. Therefore, to further specify our tasks we restrict our interests only to one sport. Our choice is a Formula 1 race. We especially concentrate on the automatic extraction of highlights from the Formula 1 race.

Why do we decide to do our research on Formula 1 among all existing sport motion pictures? We can point out many reasons to support our decision, but mainly we decide to focus our attention on analysing Formula 1 based on next:

1. The manual annotation of Formula 1 races is tedious and time consuming,
2. There are usually very few important parts of the race,
3. Race segments can be easily classified as different events, and
4. The duration of the race is very long.

The question that is always asked is: could we do this manually? The answer is of course positive, but we would have to pay the full price for this decision, because the manual annotation is tedious and time-consuming, and we could never satisfy all the answers that a user is able to request related to Formula 1 motion picture.

We can distinguish several categories of interesting events, such as start, finish, passing, fly out, pit stop, etc. Therefore, we need to characterize these types of events, model them in the appropriate way, and build a tool that will be able to extract them automatically from Formula 1 motion picture.

Considering all these facts presented above we decided to develop an automatic highlight extractor for the multimedia document of a Formula 1 race. We named this system Formula 1 Highlight Detection (F1HD) system, and in the sequel we will briefly describe its structure.

1.2. F1HD System

We decided to base our analysis of Formula 1 motion picture on combining the clues received from the audio and video signals. This decision was made based on recent publications and arguments from the multimedia and computer vision research communities. Video features have been widely used for many years for the task of indexing the multimedia documents. However, many experiments that were conducted by various authors showed that the information received from audio is highly correlated with the information in video [1]. But we did not stop at that point. Since the superimposed text in Formula 1 race usually contain a large amount of useful information, we also decided to pay attention on this information source.

Therefore, we employed three different sources of information in order to enable better indexing and characterization of multimedia documents. Each of them has its gains and drawbacks, but employed together the multimodal information sources will certainly bring more powerful results.

All these facts brought us to the challenging and difficult task to integrate the information obtained from three multimedia components (audio video, and textual information), in order to facilitate users' requests. Therefore, we developed an automatic audio/video extraction system, as well as a robust text detection and recognition system to enable multimedia characterization and indexing. Since, neither audio, nor video, is sufficient for the extraction of interesting events from multimedia files, we integrate these methods to achieve gratifying multimedia retrieval results.

The probability theory, with its inherent notions of uncertainty and confidence, has found a widespread popularity in the multisensor fusion community. Various researchers have proposed many different probabilistic models for this purpose. These probabilistic models can be classified into four broad categories: Bayesian reasoning, evidence theory, robust statistics, and recursive operators [2]. In this report we will be especially interested in Bayesian networks and their extensions that try to unify the temporal dimension with uncertainty.

Since user's requests can vary in their retrieval demands we decide to consider several approaches for the query synthesis. We have decided to extract only three global interesting events from the Formula 1 video, and to enable searching for particular segments of the race, based on keyword search. We believe that these querying techniques would be sufficient to fulfil user's requirements. Interesting events that we try to determine in the video and that can be considered as a special case of highlights are: start, passing and fly out. Keyword search is mostly based on the text extraction and recognition techniques

employed in our work. Mixing above-mentioned three groups of data types within a probabilistic network, namely the dynamic Bayesian network in our case, enables the extraction of interesting events.

To achieve highlight detection in the Formula 1 motion picture, we developed Formula 1 Highlight Detection (F1HD) system. This system is composed of next five components:

1. A video database, consisted of audio-video files ("avi") of several different Formula 1 races, and extracted meta data,
2. Audio processing subsystem – subsystem for extracting specific features from audio files ("wav"),
3. Video processing subsystem – subsystem that is used to extract visual features from motion pictures ("avi"), as well as to recognize the superimposed text,
4. Probabilistic subsystem – the dynamic Bayesian network for deciding which part of the race can be considered as an interesting event, and
5. User-query interface, which enables a user to retrieve different interesting events, or search for a specific part of the race.

1.3. Report contents

In this report we will present the system for automatic annotation of Formula 1 races. In section 2 we will discuss the related work. The techniques for the audio and video feature extraction and text detection and recognition will be presented in the third section. The characterization of Formula 1 races based on audio, video and textual clues will be described in the fourth section. In the fifth section we will present algorithms for learning and evaluation of dynamic Bayesian networks that are used for highlight detection. In this section, we will also describe numerous experiments that are conducted, and discus the use of multimodelities for the extraction of highlights. The sixth section shows an example on the usage of our system for content-based video retrieval. Some conclusions and remarks about future work will be presented at the end, in the last section.

2. Related Work

A rough categorization of the video retrieval approaches [3] yields two main classes.

The first class focuses mainly on visual features, such as colour histograms, shapes, textures, or motion, which characterize the low-level visual content. Although these approaches use automatically extracted features, representing the video content, they do not provide semantics that describe high-level video concepts, which is much more appropriate for users when retrieving video segments.

The second class concerns annotation-based approaches, which use free-text, attribute, or keyword annotation to represent the high-level concepts of the video content. However, this results in many drawbacks. The major limitation of these approaches is that the search process is based solely on the predefined attribute information, which is associated with video segments in the process of annotation. Furthermore, manual annotation is tedious, subjective and time consuming.

Obviously, the main gap lies between low-level media features and high-level concepts. In order to solve this problem, several domain-dependent research efforts have been undertaken. These approaches take an advantage of using domain knowledge to facilitate extraction of high-level concepts directly from features. In particular, they mainly use information on object positions, their transitions over time, etc., and relate them to particular events (high-level concepts). Methods have been proposed to detect events in football [4], soccer [5], and hunting [6], etc.

For example, in [7] an integrated audio and video analysis for content-based video indexing was presented. The goal was to develop a system for automatic indexing of sports videos based on speech understanding and video analysis. Authors chose to apply their algorithms for extracting touchdowns in a football game. For audio signal analysis authors used word spotting to recognize when announcer pronounce word "touchdown", and cheering detection. They fused these two features by using a simple logic. They also used video features to detect shot changes. Based on the audio clues they were able to detect touchdown shots.

On the other hand, some other approaches use stochastic methods that often exploit automatic learning capabilities to derive knowledge, such as Hidden Markov Models (HMMs), Bayesian belief networks, etc. Naphade et al [8] used hierarchical HMMs to extract events like explosions. Structuring of video using Bayesian Networks alone [9] or together with HMMs [10] has been proposed. In [11] a probabilistic model has been used to combine results of visual and audio event detection in order to identify topics of discussion in a classroom lecture environment. Another probabilistic framework that comprises multimedia objects within a Bayesian multinet has been proposed in [12].

The closest to our work is the one presented in [13]. It concentrates solely on the audio analysis for video characterization. The paper describes how audio features can be used to extract highlights for TV baseball programs. The authors relied only on audio features, but they used wide combination of them. They used a combination of generic audio features, and baseball-

specific features as well. Based on these features they developed the subsystems for: (1) noisy environment speech endpoint detection, (2) excited speech classification, and (3) baseball hits detection. A probabilistic framework and support vector machines are employed to obtain the best results.

However, only few of the mentioned approaches use fusion of audio and video clues using a probabilistic framework for such purpose. In our approach, we combined these clues to extract high-level semantics from sport videos using Dynamic Bayesian Networks (DBNs). To the best of our knowledge, this is the first time that DBNs are used for such purpose. Furthermore, we take an advantage of using the superimposed text as another information source to supplement audio-visual clues that we extracted form raw video data.

Another related problem, which has to be mentioned, is the video classification problem. In [14], Kobla et al. presented various techniques for extraction of video features that will enable identification of sports videos. They used the presence of action replays, amount of scene text in video, and computations of various statistics on camera and/or object motion. Authors also presented novel technique for the automatic detection of slow motion action replays. They focused on development of a system that was able to automatically distinguish sports scenes from other scenes.

3. Information Sources

As concluded in the previous section the biggest problem is to transform features obtained from a video sequence and to process them in order to extract semantic concepts (more about this issue can be found in [15]). Numerous approaches for extracting semantic concepts from different types of documents were presented in the literature. They showed that all above-mentioned techniques could be useful for this purpose.

In last few years, researchers in this field have noticed that audio plays a significant role in content-based video retrieval. Among the main reasons for this, we can exert much less complexity of the audio signal, and easier feature extraction. Thus, the audio signal is simpler to operate with, and audio processing needs much less computations than video processing.

For specific types of multimedia, the semantic content can be obtained based on the audio analysis only. Therefore, the audio analysis became an important part in the process of multimedia characterization and retrieval. The process of audio content extraction consists of several different sub-processes, such as speech recognition (if present), and extraction of various audio features. The audio content can be used solely for the multimedia analysis (as in [13]), discarding all the benefits from the video analysis. However, usually it brings much better results when it is combined with video clues.

Frequently, multimedia documents can contain various textual data superimposed on the screen. This superimposed text is also shown to be useful for the multimedia characterization for specific type of documents. It can help us to determine some hidden characteristics that cannot be retrieved from the audio, and video analysis. Therefore, detecting and recognizing text in multimedia documents plays a significant role in the process of content extraction.

Extracting the semantic content from video, and audio, exploiting their correlation, and their fusion into one expert model, would evidently be more efficient for indexing, and for performing high-level video abstraction. In sequel we will analyse auditory information sources, visual information sources, and their correlation in order to extract high-level semantic content from sport videos.

3.1. Audio Features

As mentioned, we are interested in extracting highlights from Formula 1 motion picture. The audio signal that we receive from Formula 1 multimedia files is very complex and ambiguous. It consists of human speech, car noise, and various background noises, such as crowd cheering, horns, etc. Usually, Formula 1 motion picture involves two or more announcers, pit reports, and on-line reports received from Formula 1 drivers. Car noise includes roaring of F1 engines, or the car braking noise. Extraction of basic characteristics from these audio recordings, which consist of complex mixtures of frequencies, is demanding and challenging task.

In order to detect highlights from Formula 1 race, we decided to analyse audio signals, because the audio information is highly correlated with the information in video. Furthermore, the occurrence of important events in a motion picture is often better characterized with the audio, than with the video. As an example, whenever there is a score in a football match, the announcer raise his voice due to his excitement. We can conclude that something important happen in the football match, even we are not watching the game. Therefore, we decided to base our detection of Formula 1 highlights mainly on audio recordings.

A great number of features can be extracted from audio signals. Some of them are useful for one specific purpose, and other for the other purposes. Also, some of them can be used for specific domain problems, and some of them are more general. Therefore, we have to choose between these features. The selection we made was based on experiments conducted with the real audio data for the specific domain.

Audio features that we chose, among the variety of others, are:

1. Short Time Energy (STE),
2. Pitch,
3. Mel-Frequency Cepstral Coefficients (MFCCs), and
4. Pause rate.

These audio features and processing techniques for their extraction that we used in our system will be explained in next subsections.

Short time energy

The main usage of this feature is for separating speech from non-speech segments in the audio signal. It is very useful in noisy environments, because noise signals have lower average short time energy than regular speech. Short time energy represents average waveform amplitude, defined over a specific time. Usually it is computed after performing sub band division of wide range signal. Indicative bands for audio characterization are lower sub bands. Therefore, short time energy for less than 4400Hz is usually used.

The average short time energy of *m* samples can be expressed using next term:

$$E_m = \frac{1}{N}\sum_{n=0}^{N-1}(x(n)w(m-n))^2, \qquad (3.1)$$

where $x(n)$ is the input sample, and $w(m-n)$ is a processing window function. This term can be rewritten as:

$$E_m = \frac{1}{N}\sum_{n=0}^{N-1}x(n)^2 h(m-n), \qquad (3.2)$$

where $h(m-n)$ represents the linear filter with impulse response $h(m-n)$. Four types of filters are frequently used for short time average energy computation: (1) square window filter, (2) Hamming window filter, (3) Welch window filter, and (4) Bartlett window filter. We employed Hamming window filter, because it brought the best results for speech endpoint detection and audio feature extraction. We can calculate the values of Hamming window filter of size *N* by using the next term:

$$h(p) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi p}{N-1}\right), & 0 \le p \le N-1 \\ 0, & \text{otherwise} \end{cases}, \qquad (3.3)$$

Pitch

Pitch (fundamental frequency – F0) is a very important feature for audio analysis, especially for detection of emphasised human speech. It represents the leading frequency of a complex audio signal. In speech, pitch gets higher values as a result of speaker excitement. Many techniques have been proposed for pitch estimation and tracking, such as cepstrum analysis, harmonic product analysis, autocorrelation analysis, maximum a posterior (MAP) pitch estimation, autocorrelation analysis, difference analysis, etc. We will describe autocorrelation analysis and MAP estimator in the sequel.

All these techniques for pitch estimation demand appropriate bandwidth of audio signal for accurate estimation of the pitch. Since human speech is usually under 1000Hz, we are particularly interested in determining pitch that is under this frequency range.

We decided to use the autocorrelation function in our approach of pitch estimation. The autocorrelation function for a random signal is defined as

$$A(k) = \frac{1}{2N+1}\sum_{n=-N}^{N}x(n)x(n+k), \qquad (3.4)$$

where *k* is the number of overlapping samples. Important properties of the autocorrelation function are that the autocorrelation function of a periodic signal (like speech signal) is also periodic, and that its first value *A*(0) is the average energy of the signal.

From equation (3.4) we can compute peak values for different values of *k*. Pitch for the particular window is defined as a location of the largest peak value ($\max(A(k))$) in the selected window, if the autocorrelation function is above a particular threshold (0.3 in our work). If last constraint is satisfied, we can calculate pitch as:

$$P = \frac{f}{k_{\max}} \quad [Hz]. \qquad (3.5)$$

It is shown that detection of emphasised human speech for complex sounds can be achieved by pitch tracking. We can even learn these pitch characteristics for a particular talker (an example in [16]). Only thing we have to do is to carefully determine the pitch threshold for every talker.

In our approach we use pitch as one parameter of a DBN for finding the segments of excited speech.

Mel-frequency cepstral coefficients

This is a type of phoneme-level features for characterizing audio signals. It is also based on sub band division of entire frequency spectrum. MFCC are based on Mel-scale. Mel-scale is gradually warped linear spectrum, with coarser resolution on a higher, and finer resolution on lower frequencies. It is metrically adapted to the human perception system. Based on this division, Mel-frequency sub band energy is defined. With an extension of Mel-frequency energy to phoneme-level features, this energy is shown to be useful for automatic speech recognition. The MFCC are a simple cosine transform of the Mel-scale energy for different filtered sub bands.

However, for audio characterization, Mel-frequency cepstral coefficients are shown to be more appropriate than Mel-frequency sub band energy ratio. Details on this can be found in [17]. MFCC and especially theirs first derivatives bring good results for determining speech phonemes, and are very useful for speech recognition and speaker detection. More details on usage of MFCC and their first derivatives can be found in [17].

Pause rate

The pause rate feature intends to determine the quantity of a speech in an audio clip. The pause rate can be used as an indication of emphasised human speech. It can be easily calculated by counting the number of silent audio frames in an audio clip. If we denote the number of silent segments in an audio clip *u* as S_u, we can write:

$$S_u = \sum_{j=0}^{k} \begin{cases} 1 & \text{if silent audio frame} \\ 0 & \text{otherwise} \end{cases}. \qquad (3.6)$$

Then we can average this on each audio clip with the number of audio frames in a clip ($|u|$) and obtain pause rate (P_r):

$$P_r = \frac{S_u}{|u|}. \qquad (3.7)$$

3.2. Video Features

This section describes the extraction of visual features from the video sequence of Formula 1 motion picture and various processing techniques used for extraction. Despite the fact that the video processing of multimedia documents is very time and power consuming, it yields significant information about the motion picture. Therefore, we need to investigate the usefulness of video clues and cost of their computation.

One of the most important tasks in the video retrieval and characterization process is to model the video content. Video usually contains large amount of information that can be used for the characterization of motion pictures. Numerous techniques and procedures have been proposed for the extraction of different semantic information from videos. All these procedures require high computational power and are time consuming.

Although, video processing is much more demanding than audio processing it can supply us with much more useful information about the motion picture. Therefore, we can say that the cost of video processing is high but the benefits from it are priceless.

As stated in [18], the video content can be described throughout three different levels: (1) raw data, (2) low-level visual content, and (3) semantic content. The video data is represented by elementary video units and general video attributes (for example format and frame rate). Visual features, such as colour histogram, dominant colour, shape moments, etc., characterize low-level visual content. The semantic content consists of high-level concepts, such as objects and events.

There are many video features that can be extracted from video. Recently, many papers that review video features have been published [19,20,21]. In this variety of possible solutions we should navigate properly and pick features that can be useful for our purpose.

Numerous features are categorized into four groups, according to [19]: (1) colour, (2) texture, (3) shape, and (4) motion. In the sequel we will describe some video features that we used in our system.

Colour

Colour is the basic attribute of an image and is very important for the image description. Based on colour we can define two basic video features. One is based on the pixel colour intensity (for example, red, green and blue intensity), and the other represents the colour distribution of an image and is termed colour histogram. Calculating the pixel colour intensity is computationally expensive, and is rarely used in the field of motion pictures. However, colour histogram is one of the most frequently used video features. It is invariant to image rotation, translation, and viewing axis (camera zooms and pans).

We can use various quantization methods and colour coordinates to form the colour histogram. These quantization methods for different colour spaces are: RGB, YUV, HSV, etc. However, main drawback of a colour histogram is that it does not include the spatial configuration of pixels with the same colour. Therefore, colour histograms for two completely different pictures can be very similar.

Shape

A shape is represented using a moment invariants, Fourier descriptors, autoregressive models, and geometry attributes. Shape features can be classified into global and local features. Global ones are the properties derived from the entire shape (for example roundness, central moments, eccentricity, and major axis orientation). Local features are derived from the partial processing of a shape (size orientation of consecutive boundary segments, points of curvature, corners, and turning angle). In our work we used global properties such as roundness and major axis orientation.

Motion

Motion information is based on block-matching or optical flow techniques. Motion features can be seen as low or high-level features. Low-level features are moments of the motion field, motion histogram, and global motion parameters. They can be extracted from the motion vectors. High-level features reflect the camera motion such as panning, tilting, and zooming. For our purpose, we used optical flow techniques based on motion vectors formed from pixel colours.

3.3. Text

Sport clips usually contain a large amount of textual information. This information can be useful for characterization of videos. The technique used for text recognition is frequently termed as Video Optical Character Recognition (VOCR). If we combine this technique with other video characterization techniques we will be able to better describe the semantic content of motion pictures. The text that appear in the digital video can be broadly divided into two classes:

1. Scene text, and
2. Graphic text

Scene text occurs as a natural part of the actual scene captured by the camera. Examples of screen text can include billboards, text on vehicles, writings on human clothes, etc. Graphic (superimposed) text is mechanically added text to video frames in order to supplement the visual and audio content. It usually brings much more useful information, since it represents additional information for better understanding of the video, and is closely related to it.

One characteristic of text strings is that the width and height of a string can be used to filter out the text. For horizontally aligned text strings, their aspect ratio varies in magnitude if we are recognizing one or several words, or several of them. We can also put minimum and maximum bounds for the detected text regions, and eliminate each region if it does not satisfy these constrains. The importance of these issues comes to light when we perform the text recognition.

Another useful characteristic of the superimposed text is that it usually spans over hundreds of milliseconds or several seconds. That means that the number of frames with the same superimposed text is about several tens or hundreds of fames. This fact is very important from computational point of view, because the text detection on each frame would be too expensive for computation.

Two tasks that will enable us to use the superimposed text from the complex background of Formula 1 motion picture are:

1. To find a region in the video where the superimposed text is located, and
2. To recognize the superimposed text from these regions.

Of grate importance for the text detection on the scene, are next three characteristics: (1) text poses certain frequency and orientation information, (2) text shows spatial cohesion – characters of the same text string (one word, or several words in the same line) are of a similar height, orientation and spacing, and (3) graphic text regions are usually placed in the specified place on the screen (in the top or in the bottom of the screen, or in the bottom right angle, etc.).

For the second task, current optical character recognition (OCR) systems would give good recognition accuracy, if we have a text printed against the clean background. However, since we are interested in recognition of the text printed against shaded and textured backgrounds, OCR technology cannot handle such texts. Hence, we had to develop a specific algorithm for that purpose, which will be explained in section 4.6.

4. Characterisation of Formula 1 Races

In this section we will present all the audio and video clues that we extract from Formula 1 motion pictures. We will start by explaining the audio transformations, needed for audio feature extraction. Then we describe how the audio features are used to detect speech endpoints, moments with excited speech, and keywords. In section 4.5 we will explain video segmentation and event recognition. Finally, the algorithms used for text detection and recognition will be presented.

4.1 Pre processing of audio signal

Audio-video files ("avi") for Formula 1 race are usually very long and it is more complicated and time consuming to extract audio features from them, than from much smaller and easier to process audio ("wav") files. Therefore, we decided to extract the audio signal from "avi" files and to store it as "wav" files. After this extraction we must filter the signal to an appropriate form that can be used for the feature extraction. Since we are going to do keyword spotting as well, the audio signal must be reshaped to a specified form for the keyword-spotting tool we use.

The keyword spotting tool demands an input with next parameters: (1) frequency of audio signal must be 16kHz, (2) samples must be 16b long, and (3) it must be mono audio signal. Since the original frequency of our digitalized motion picture is 22kHz, we must transform this 22kHz signal into 16kHz signal.

The feature extraction can be very complex and can demand various bandwidth of audio signal as input. To deduce which band of audio signal is most appropriate for extraction of a specific feature numerous experiments were conducted. These experiments lead us to the division of wide spectral range of the audio signal into two categories. We decided to form the different measure of the audio signal: (1) filtered original signal below 882Hz, and (2) between 882Hz and 2205Hz. This is mainly because these frequency bands are shown to be the most appropriate for extraction of the short time energy for speech endpoint detection and detection of emphasised announcer speech, as we would see in the next sections.

For time-segmentation of audio signal we decided to use 10ms for audio frames and 0.1s for audio clips. The duration of the audio frame of 10ms is chosen because it is most frequently used in literature for similar purposes, and because experiments showed that it brings good results. However, duration of 0.1s for audio clip is somewhat short. Two main reasons are major for this shortness: (1) we need probabilistic values as output from this subsystem that will feed the network frequently, and (2) the duration of audio clip should be about the duration of a video frame in order to fuse them better.

Therefore, a pre-processing system prepare next inputs for feature extraction system:

1. Wide range audio signal for feature extraction (0-22kHz, 16b, mono)
2. Wide range audio signal for keyword spotting (0-16kHz, 16b, mono)
3. Low-pass audio signal for feature extraction (0-882Hz, 16b, mono)
4. Band-pass audio signal for feature extraction (882Hz-2205Hz, 16b, mono)
5. Audio frames of 10ms for short term calculations of audio features
6. Audio clips of 0.1s for long term normalized parameters extracted from audio signal

4.2 Speech endpoint detection

For speech endpoint detection we choose two features:

1. Short time energy for filtered audio signal, and
2. Mel-frequency cepstral coefficients

We use 0-882Hz filtered audio signals for calculating short time energy, because this bandwidth diminish car noises, and various background noises as well. This elimination is very important, especially when we have to determine silent segments (see Figure 4.1). We decide to employ the Hamming window for processing STE because of its good characteristics. For MFCC calculations we use only first three coefficients of total number of 12 coefficients, because they are shown to be the most indicative for speech detection. We calculate the value of these features for each audio frame, their average values and dynamic range, and maximum values of STE for audio clips. After setting the appropriate thresholds for these parameters, based on numerous trials, we were able to perform speech endpoint analysis of an audio signal. The thresholds we used are 2.2×10^{-3} for the weighted sum of the average and maximum values and dynamic range of STE, and 1.3 for the sum of the average values and dynamic range of first three coefficients of MFCC. As a result we receive indication for each 0.1s segment whether it can be considered as a speech or non-speech segment. The whole system is depicted in Figure 4.2.

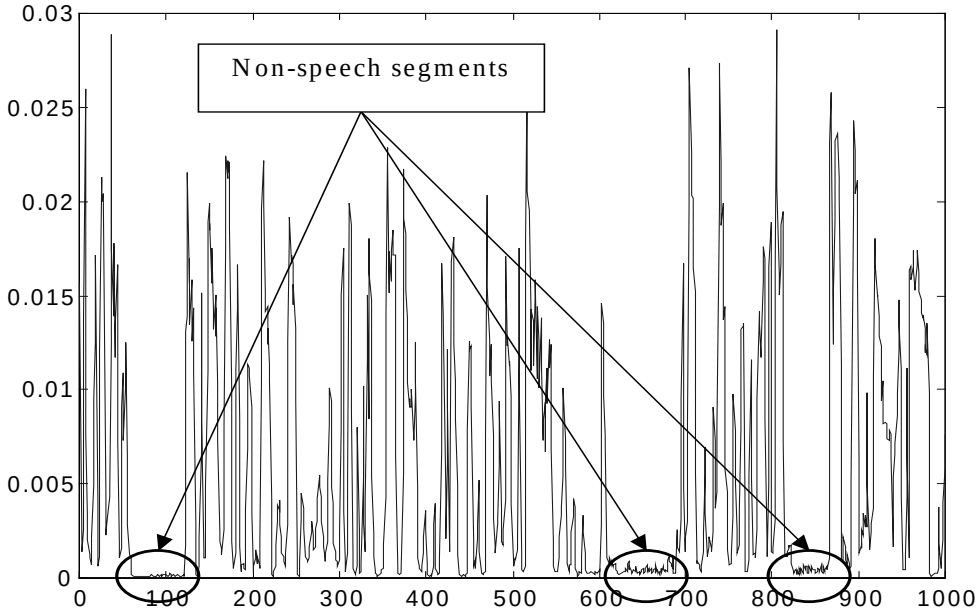

Figure 4.1 *Short time energy calculations for 10s (1000 audio frames) with indications of non-speech segments. These small sample variations of short time energy are eliminated when we perform filtering*

Our experiments with entropy and zero crossing rate showed powerless when applied for speech endpoint detection in a noisy environment such as ours. Therefore we found STE and MFCCs more appropriate for speech endpoint detection in Formula 1 motion pictures. Our speech endpoint detection is similar to one proposed in [13]. This is mostly because our audio signal is similar to one that was extracted from baseball games. However, our audio signal differs in many parameters (frequency bands, thresholds for speech endpoint detection, etc.). Thus we had to investigate its characteristics to enable speech endpoint detection.

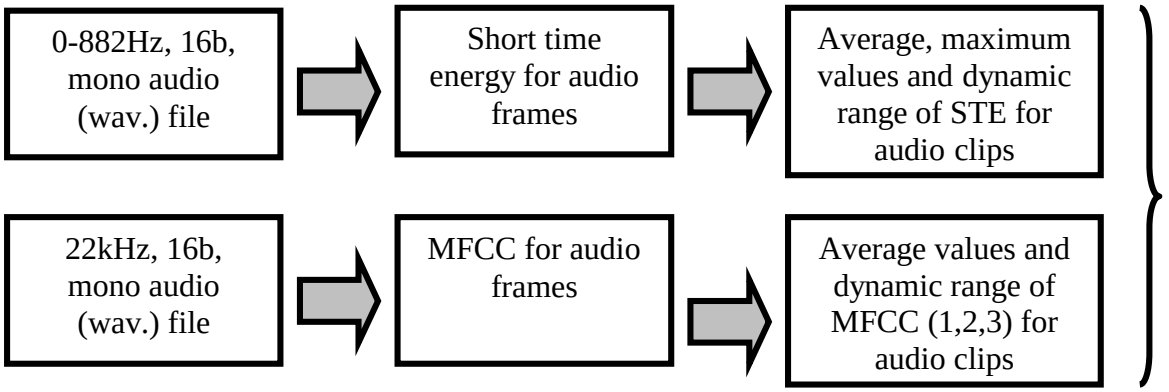

Figure 4.2 *Structure of speech endpoint detection subsystem*

4.3. Classifying Excited Speech

For detection of the emphasised announcer speech we decided to use next features:

1. Short time energy
2. Mel-frequency cepstral coefficients
3. Pitch, and
4. Pause rate

For different features we used different frequency bands. For STE we used filtered audio signal, 882Hz-2205Hz, and for MFCC and pitch we used low passed audio signal, 0 - 882Hz. We compute average and maximum values in an audio clip for all these features obtained for audio frames. Additionally, we compute a dynamic range for STE and pitch as well. These computations are only performed on speech segments obtained by the speech endpoint detection algorithm. To compute

average and maximum values, and dynamic range we used only the values of first three features from those audio frames marked as speech frames.

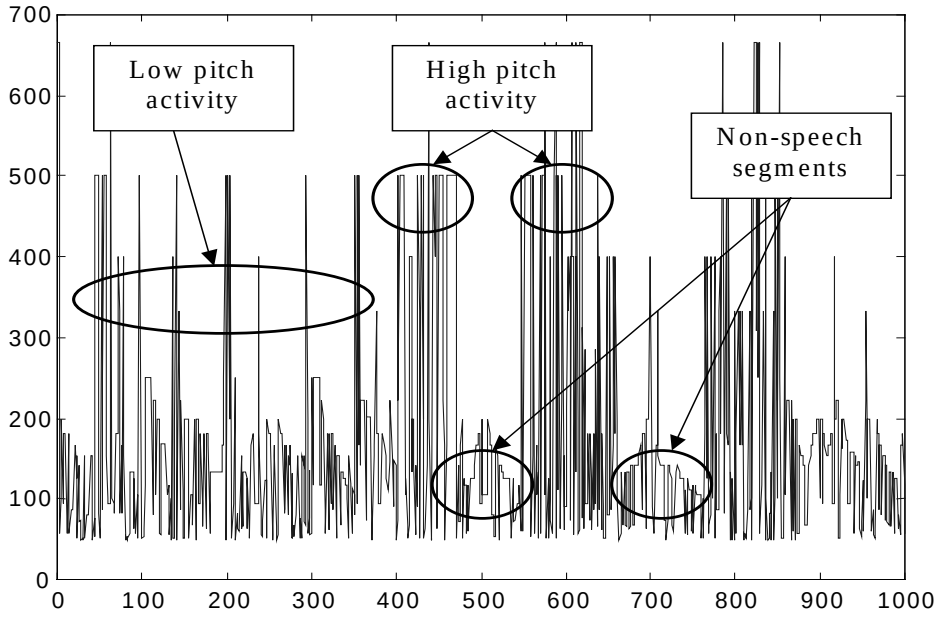

Figure 4.3 *Pitch values for 1000 audio frames*

In Figure 4.3 we can see pitch values of parts that correspond to segments of excited, and non-excited speech.

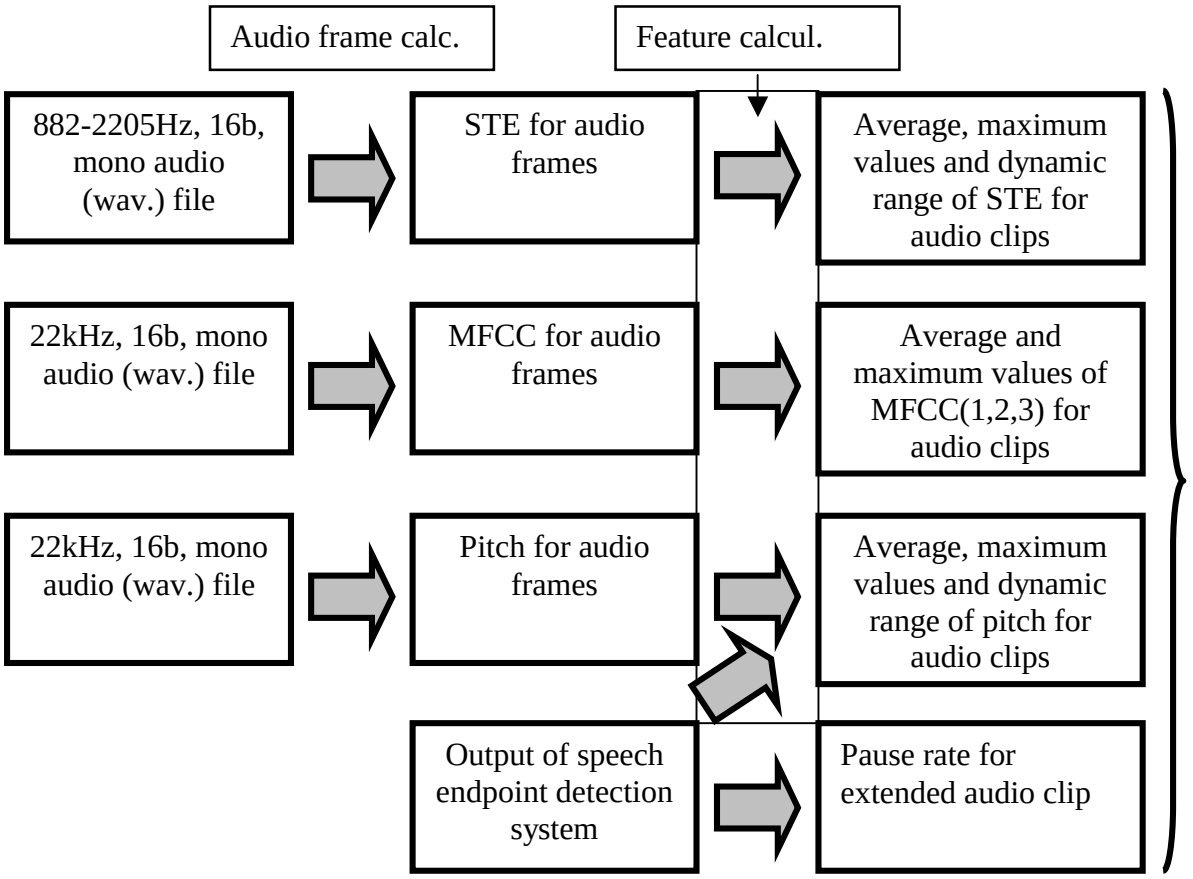

Figure 4.4 *Subsystem for feature extraction*

For the pause rate we used different calculation techniques. Since the pause rate represents duration of non-speech segments in a specified time interval, it has to be calculated for a longer period. Therefore, we calculate it for an extended audio clip of 5s, and re-estimate it for all 0.1s audio clips.

The structure of the audio feature extraction subsystem can be seen in Figure 4.4.

4.4 Keyword Spotting

Speech patterns have regular structure where a spoken utterance usually consists of several words. In each spoken utterance every word is made up of syllables followed by vowels and then consonants. As explained in [22], vowels bear high energy due to the lack of the vocal tract constriction whereas consonants bear low energy due to the constriction of the vocal tract.

Based on these regularities, a Finite State Grammar Decoder (FSGD) package can be developed to perform keyword spotting, like in [23]. This package produces word strings based on posterior probabilities, phone models, and a finite state grammar. Training phase is used to generate both, phone models and posterior phone probabilities.

This keyword spotting approach is based on recognition of a string of keywords, and "garbage". For this purpose, a simple grammar is used. The topology of Finite State Grammar (FSG) can be seen in Figure 4.5. As a garbage model, a simplified unigram phone model is used.

The FSG is consisted of one node, with N+M transitions, where N denotes the number of phones in the phone set, and M represents the number of keywords. The task is then to assign probabilities to each of these transitions (P[P*i*] or P[W*i*]). Counting the frequencies of keywords and garbage phones within the training data does this. Based on the training process, we can later perform speech recognition task.

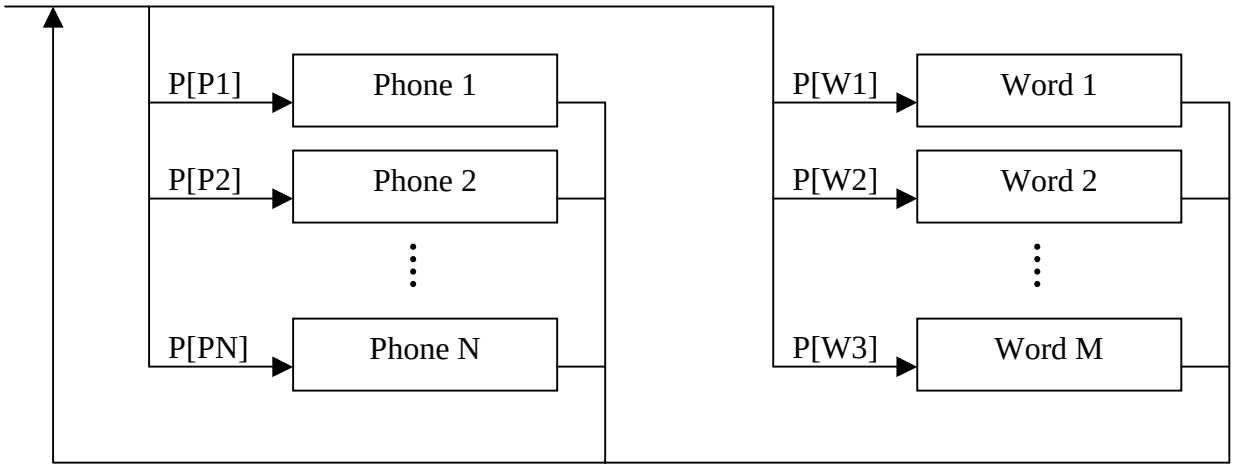

Figure 4.5 *Topology of the finite state grammar*

As stated in [22], the optimal performance of any keyword spotting system is task-dependent. We should decide whether we want to find every occurrence of every word at the expense of generating false alarms, or we want to pick out the keywords we can be absolutely sure of. The former method will bring fewer false alarms. Therefore, we can conclude that this decision is based on a trade off between the numbers of correct hits versus the number of false alarms. In the sequel we will explain which experiments were done in our approach and what are the obtained results.

The keyword spotting system has 16kHz "wav" file as input. We extract about 30 words that can be usually heard when

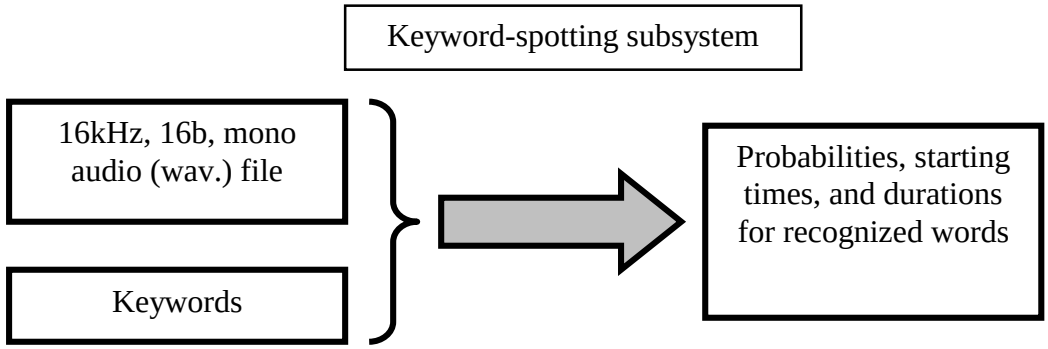

Figure 4.6 *Keyword spotting subsystem*

commentator is excited, or when it is a specific segment of the race that we are interested in. We tried two different acoustic models for this purpose. One was trained for the clean speech, and the other was aimed for word recognition in TV news. The latter showed better results. Thus, we employed it for keyword spotting in our system. Note that even better results could be obtained using a specific acoustic model for the Formula 1 race.

The keyword spotting system calculates the non-normalized probability for every specified word, the appropriate starting time when the word is recognized, as well as the duration of the recognized word. These parameters are then stored into a file and can be further processed later.

We transform this information into probabilistic values. Probabilistic values are computed based on the number of correctly recognized words. The structure of the keyword spotting tool can be seen in Figure 4.6.

4.5. Video Analysis

The visual analysis of a video sequence is based on extracted video features. Since purpose of our system is to process Formula 1 motion pictures, and because of the possibility to perform this task in many distinct ways and extract different video attributes, at the beginning we had to decide what kind of information we need to extract from these documents. This information must be useful for further processing in a probabilistic framework and must enable us to make important conclusions about the recorded race, which we can later present to the end-user.

After considering various scientific research projects that had a goal similar as ours, we decided to employ next general techniques for video segmentation and characterization:

1. Shot segmentation, and
2. Replay detection.

Since the main goal of our system is to detect highlights from Formula 1 motion picture, we need to extract some domain dependent characteristics of video source. These characteristics can be considered as video contents correlated with:

1. Start,
2. Passing, and
3. Fly outs.

To extract these concepts from video features we had to develop a model for each of them. These models will be presented in the sequel.

Shot segmentation

For shot segmentation we employed a simple histogram based algorithm. According to [7], shot cuts can be detected by comparing histograms from two consecutive frames. If these two frames are “substantially different” we could separate them as belonging to different shots. This histogram difference is defined as:

$$HD(H_t, H_{t-1}) = \sum_{i=1}^{N} \frac{(H_t(i) - H_{t-1}(i))^2}{H_t(i)}, \qquad (4.1)$$

where H_t is the histogram for the time *t*, and *N* is the total number of colours in an image. If we put an appropriate threshold for this histogram difference, we will be able to separate shots with the high accuracy. Unfortunately, if we have the digital shot change effects (such as fade, morphing, etc.) this technique will yield poor results.

Therefore, we modified it in sense that we calculate histogram difference among several consecutive frames in the multimedia document. This algorithm resulted in over 90% of accuracy, which we considered satisfying. Therefore, we did not make experiments with more sophisticated methods for shot segmentation.

Replay detection

Broadcast sport videos usually contain a large amount of replay scenes. Extraction of these scenes is not an easy task. Therefore, the domain knowledge about the production of TV sport programs and type of the broadcasted game play a significant role in the development of replay detection systems.

The simplest way to detect replays is to distinguish when the superimposed text like “Replay” is put on the screen, or when a textual overlay arise during the replay scene. However, this is not always the case. Sometimes replays are slowed down representation of live scenes. Another case is that they have special shot change operations. They are termed Digital Video Effects (DVEs), and are present in the beginning and at the end of a replay scene. A technique to find replays based on DVEs is described in [24].

Replays in Formula 1 are usually neither slowed down, nor marked. Therefore, we could not employ any of the algorithms. Replays are represented as a gradual shot changes or morphing scenes in the motion picture. We could employ the DVE-based detection, but the problem was that these DVEs must be learned for every race. Even then, since they may vary, and

even be omitted for some replays, this would lead to small replay recognition accuracy. These algorithms are also time consuming and computationally expensive. Therefore, we decide to employ a simpler algorithm based on motion flow and pattern matching, and manually improve the results if necessary.

Start

Start is defined by two parameters: (1) the amount of motion in the scene, and (2) the semaphore presence in the image. To detect the amount of motion we used pixel colour difference between two consecutive frames. We tried to employ colour histogram for this purpose, because it is less computationally expensive, and therefore faster, but the results did not show useful. However, red, green, and blue pixel colour difference brings good results for describing amount of motion in the scene. The semaphore was described as a rectangular shape, because the distance between red circles is small and they touch each other. This rectangular shape is increasing its horizontal dimensions in regular time intervals, i.e. after constant number of video frames. This rectangular region is detected by filtering the red component of RGB pixel colour representation of a still image.

Passing

Since we did not try to use powerful motion analysis, because it will take to much time, we use some less computationally demanding features. We calculate the movement properties on several consecutive pictures, based on motion histogram obtained from them. Such obtained outputs enable us to compute the possibility that there is a chance of one car passing another. Note, that we employed very simple and naïve approach for passing detection. By applying more powerful techniques for object tracing we could obtain much better results.

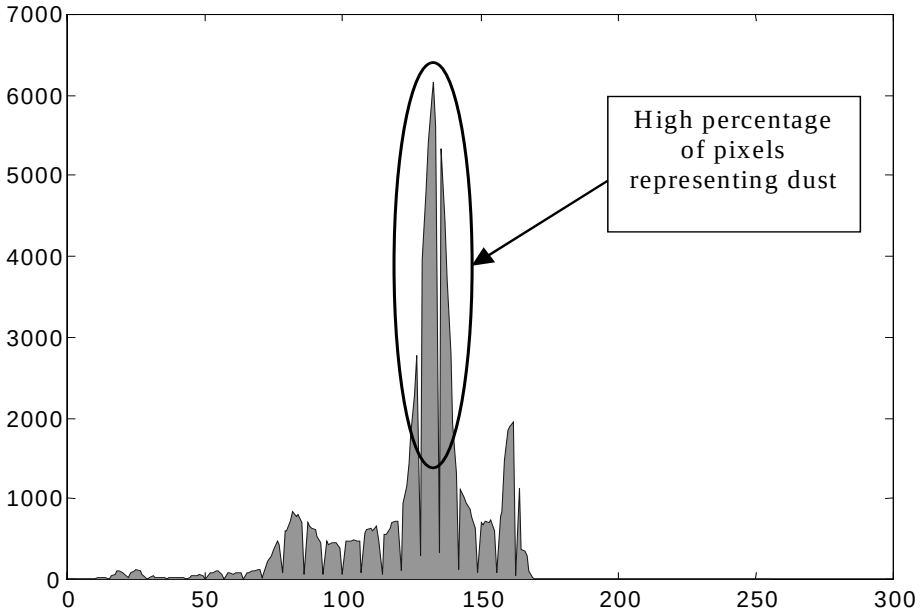

Figure 4.7 *Colour histogram of a frame that represents fly out*

Fly outs

Fly outs usually come with a lot of sand and dust. Therefore, we had to recognize presence of these two characteristics in the picture. We resolve this problem using the filtered RGB image for dominant colours. First we determined a dominant colour in several still images of fly outs. Then, we employ colour filtering to extract the amount of presence of these dominant colours in the still image of Formula 1 motion picture. An example of a colour histogram of such an image is given in the Figure 4.7.

Post processing of features

As a result of previous content extraction tasks we receive evidence that contains non-normalized values, which characterize the specific type of content (except the values obtained from the shot segmentation task). Since we need to use them as an input for a probabilistic framework, we had to reshape them into probabilistic values. This transformation was done based on experiments with video sequences from Formula 1 motion picture. We calculate maximum and minimum values of these parameters for large number of frames, and select appropriate thresholds and weights. Then, these weighs and thresholds are used to normalize the values of previously described content characteristics.

4.6. Detecting and recognizing superimposed text

In sport videos a large amount of text is present in the images. This text is usually mechanically added to the video sequence, which means that it belongs to the class of graphic text. Since the text is added on purpose, it frequently contains information that is closely correlated with the actual scene, and brings a lot of useful information for video characterization.

In our case of Formula 1 motion picture, this superimposed text can be used as additional information, together with other video and audio clues. It is important to note that textual information in the Formula 1 motion picture is placed in the bottom part of the image, and has a uniform structure. This means that we do not need to analyse the whole picture for text detection and recognition. We can also make patterns for these textual structures that will ease recognition task, because of their property of regularity.

Since the process of text detection and recognition is complex, we will divide it into three steps, as follows:

1. Text detection
2. Refinement of text regions
3. Text recognition

We will describe techniques that were used in our system for each step. An example of text detection and recognition can be seen in Figure 4.8.

Text detection

Since the number of frames in typical motion video is large, processing each frame for text recognition is not computationally feasible. Therefore, the first step of text recognition task will be to find text regions in a still image. According to [25], a text region can be defined as a horizontal rectangular structure of clustered sharp edges. This is due to the fact that characters form regions of high contrast against the background.

For this task, a horizontal differential filtering is usually used. This filtering searches for horizontal rectangular structures of clustered sharp edges, which in most cases represents text regions. Usually filter magnitude is 3 by 3 pixels. By applying appropriate binary thresholds we are able to extract vertical edge features, and based on them we can define text regions.

We used the property of our domain that the superimposed text is placed in the bottom of the picture. To detect whether the superimposed text is present in the picture, we simply need to process this part of the picture. When the text is superimposed, the background is shaded in order to make characters clearer, sharpened, separated from the background, and easier to read. The characters are usually drawn with high contrast to the dark background (light blue, yellow, or white).

Another significant property of the superimposed text is that it remains on the motion picture for more than one frame (usually more than 50 fames), together with the shaded background. Based on this we develop a text detection algorithm.

Our algorithm is a two-pass text detection algorithm. In the first part we analyse if the shaded region is present in the bottom part on every image in the video sequence. We conclude that the image is shaded based on the colour feature for every pixel in the bottom part of the image. Since shaded region in Formula 1 motion picture can have two different sizes we employ synthesized detection for both of them. Computing the number of these shaded regions in consecutive frames, we skip all the short segments that do not satisfy the duration criteria.

The second part analyses the same time-dependent properties of shaded regions. We calculate the duration, number, and variance of bright pixels present in these shaded regions. If computed values for the video sequence satisfy constrains defined for the text detection algorithm then this video sequence is marked as a sequence that contains the superimposed text.

Refinement of text regions

However, characters in these text regions cannot be extracted in regular shape, and they are hard to be recognised. Two most important difficulties that arise in the text recognition task are: (1) complicated background with a lot of movement and overlapping, and (2) insufficient image resolution. Therefore, for the text recognition, frequently, we must further clean up text regions. This is done in several ways, and such techniques are termed as text refinement.

A text region that we receive as a result of text segmentation usually contains the text of similar intensities and same background. Therefore, we can put a simple threshold to distinguish characters from the background. Then we could binarize the text region, to make characters stand out. The binarization threshold is based on histogram values obtained from the text region. This is one way of cleaning up the text. However, it does not always supply us with clean and regular character forms, especially if the size of characters in the text is small. To overcome this problem, we must employ more powerful algorithms.

The refinement process consists of next steps: (1) filtering of text regions, and (2) interpolation of text regions.

We need to filter the text regions in order to enable better separation from the background, as well as for sharpening the edges of characters. The filtering was executed through minimizing or maximizing pixel intensities over several consecutive frames. However, this filtering is not sufficient for text recognition. Therefore, we had to employ an interpolation algorithm

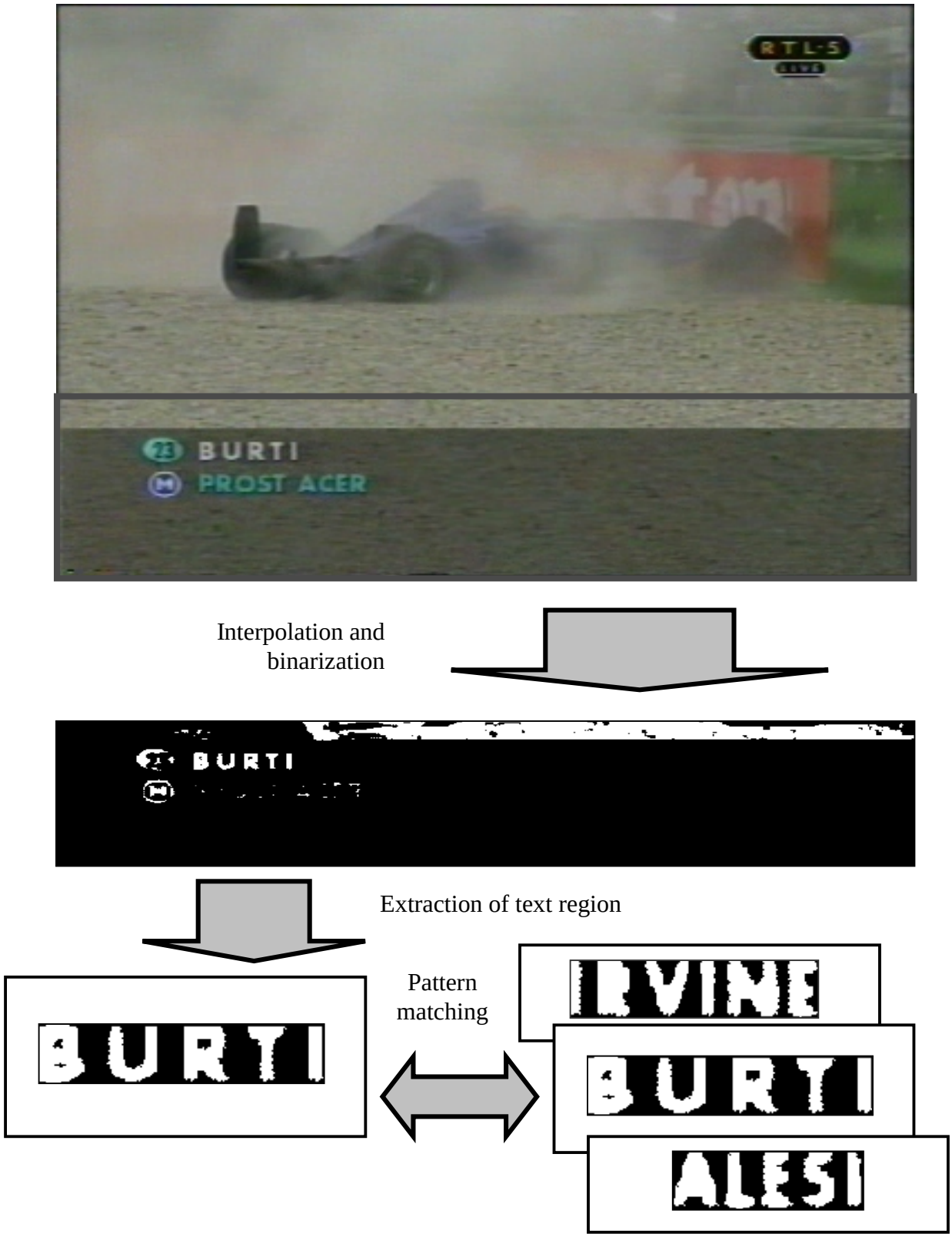

Figure 4.8 *Text recognition*

to enlarge characters and make them clearer and cleaner. In this interpolation algorithm the text area is magnified four times in both directions. After this refinement, we have magnified text regions with much better character representations. After these actions, the text is ready for the text recognition step. The filtering and interpolation algorithms will be described in the sequel.

In a video sequence, we can utilize the fact that text is placed in the same position for a whole sequence of frames. Therefore, to sharpen these regions, sequential filtering of these text regions across a defined number of consecutive frames (20 in our case), is performed. Filtering is usually done by minimizing or maximizing intensities of every pixel across the selected sequence of frames. Shall we minimize or maximize intensity of a pixel depends on the colour and background of

superimposed text. Since we have the light text regions we minimize the pixel intensities. This results in the increased text resolution and eases the task of text recognition.

To enhance text quality on each image we can use next equation:

$$I_e(x, y) = \min(I_i(x, y), I_{i+1}(x, y), ..., I_{i+n-1}(x, y)) . \qquad (4.2)$$

Here, *x* and *y* indicate the position of a pixel in an image, *i* represents the starting frame and *i+n* is the last frame of the video sequence with the text region detected in it. The enhanced image is termed as I_e, while *I* represents the original image.

The superimposed text usually consists of small characters. This results in insufficient character magnitude and resolution for employment of standard character recognition techniques. To enlarge and to perform further purification of an image, a linear interpolation technique is used in our work. With this technique we magnify characters and picture resolution for four times .

The interpolation is performed as described in the sequel. Each pixel of the original image (now $I_e(x, y)$) is placed on each fourth pixel in both directions (x and y) in the interpolated magnified image $I_m(x, y)$:

$$I_m(4x, 4y) = I_e(x, y) . \qquad (4.3)$$

Taking all neighbouring pixel values, other pixels are computed by linear interpolation:

$$I_m(x, y) = \frac{\sum_{(x_0, y_0) \in N(x,y)} d(x - x_0, y - y_0) I_e(\frac{x_0}{4}, \frac{y_0}{4})}{\sum_{(x_0, y_0) \in N(x,y)} d(x - x_0, y - y_0)} . \qquad (4.4)$$

where $N(x, y) = \left\{ (x_0, y_0) \mid x_0 \in \left[4\left\lfloor \frac{x}{4} \right\rfloor, 4\left\lceil \frac{x}{4} \right\rceil \right], y_0 \in \left[4\left\lfloor \frac{y}{4} \right\rfloor, 4\left\lceil \frac{y}{4} \right\rceil \right] \right\}$, and $d(x, y) = \|(x, y)\|^{-1}$.

After performing these text refinements we obtained a much bigger, cleaner and sharpened picture. We can now apply a binary threshold at a fixed value of the pixel intensity.

Text recognition

The text recognition task should enable extracting words and sentences from purposely-added superimposed text. This procedure should supply us with additional information about a video scene that can be used for conclusions about the content of a video sequence (in our case also as an input in a probabilistic framework).

In our domain, two important issues can be exerted about the superimposed text: (1) there are not many different words superimposed on the screen, and (2) font used for superimposed text is the same. Based on these issues we developed our text recognition algorithm.

Assuming that we received a clean and sharpened text from the previous step, we can now begin character/word recognition process. The character recognition process is somewhat more complex and time consuming, but it is more robust and more general, because we can employ it on any font size. On the other hand, word recognition is limited on a specified number of defined words (extracted reference patterns), but it is less computationally demanding. We decided to recognize words based on extracted text regions with one or several successive characters.

Our algorithm is based on pattern matching techniques, mainly because of a small number of different words superimposed on the screen. These words are names of the Formula 1 drivers, and some informative words, such as pit stop, final lap, classification, winner, etc. Therefore, the first task was to extract patterns for these words.

Since the processing of a colour image is computationally expensive and slow, we decided to extract reference patterns, and to perform matching with black-white pictures. Black-white text regions are obtained from the colour text regions by filtering RGB components. After applying thresholds on the text region, we marked characters as white space on the black background.

For character extraction we used the horizontal and vertical projection of white pixels. Since characters can have different heights we used double vertical projection in order to refine the characters better. However, we did not match characters to reference patterns because they are usually irregular and can be occluded or deformed. Thus, we connect characters that belong to one word into a region. This was done based on the pixel distance between characters. Regions that are closed to each other are considered as characters that belong to the same word.

Having the regions containing one word, we perform pattern matching. To make this matching algorithm faster and more powerful, we separate words into several categories based on their length, and perform matching procedures only for reference patterns with similar length. A simple metric of pixel difference that is used for pattern matching is described by next term:

$$m_c = \sum_{(x,y)} I_r(x,y) I_{bw}(x,y), \qquad (4.5)$$

where $I_r(x,y)$ represents the reference pattern image, and $I_{bw}(x,y)$ represents the black-white image of the extracted text region. By specifying an appropriate threshold for similarity matching (0.225 in our case), we were able to recognize the superimposed word. Thus, a reference pattern with largest metric above this threshold is selected as a matched word or character.

5. Extracting Highlights Using DBNs

The aim of this section is to explain the fourth component of our system, i.e. the probabilistic subsystem. In this section we will explain how results, which are obtained from the audio and video subsystems and described in the previous section, are incorporated into one compact scheme of a dynamic Bayesian network. We will also present results obtained from numerous experiments conducted for evaluation, and comparison of different DBN structures and modified inference algorithms. We will also present a DBN learning algorithm.

5.1. Data Set

Here, we will give the characteristics of multimedia documents that we used in our experiments. We digitalized three motion pictures of Formula 1 races, as can be seen in Table 5.1. The duration of digitalized Hockenheim Formula 1 race ("HockenheimF1.avi") is relatively short in comparison to the other two races, which we obtained by recording the whole Formula 1 races. For the former race, we decided to take only several segments from the race with a lot of interesting moments. Therefore, we use this relatively short "avi" file for numerous experiments that we conducted to test various DBNs algorithms and structures, as well as to test the usefulness of our system. Two other files were used only for evaluation of our system.

Table 5.1 *Data set for experiments and evaluation*

Circuit	Hockenheim	Spa	Indianopolis
"avi" file	"HockenheimF1.avi"	"SpaF1.avi"	"USAF1.avi"
Media length	22:21 (1341s)	1:15:10 (4510s)	1:23:08 (4988s)
File size	424MB	1762MB	1639MB
Frame rate	25	25	25
Video size	384*288	384*288	384*288
Audio sample rate	22kHz	22kHz	22kHz
Bits per audio sample	16	16	16

Not only these data were stored in our system. After audio and video processing of these three "avi" files we received evidence that was also stored in our system. Feature values extracted from the audio signal, as well as from the video sequence, represent probabilistic outputs and have values in range from zero to one. They are stored as vectors of values for each feature. Since the parameters are calculated for each 0.1s, the length of feature vectors is ten times longer than the duration of video measured in seconds.

The audio vectors (features) we extracted from audio component of multimedia document are:

1. "Keyword" vector – vector of probabilities for spoken words,
2. "Pause rate vector" – vector of probabilistic values for the relative duration of speech segments,
3. Average values of short time energy,
4. Dynamic range of short time energy,
5. Maximum values of short time energy,
6. Average values of pitch,
7. Dynamic range of pitch,
8. Maximum values of pitch,
9. Average values of Mel-frequency cepstral coefficients (based on first three coefficients), and
10. Maximum values of Mel-frequency cepstral coefficients (based on first three coefficients)

From the video sequence we extracted next vectors (features):

1. “Part of the race” vector,
2. “Replay” vector,
3. “Color difference” vector,
4. “Semaphore” vector – vector of probabilities of semaphore presence in the picture,
5. “Dust” vector – vector of probabilities that the dust is in the picture,
6. “Send” vector – vector of probabilities that the sand is in the picture, and
7. “Motion” vector – calculates the amount of motion

Since we also employed text detection and recognition algorithms, we were also able to extract text from video sequence. We decide to extract the names of Formula 1 drivers, and the semantic content of superimposed text (for example if it is a pit stop, or driver’s classification is shown, etc.). These parameters are saved as a multidimensional field with next components:

1. “Type” vector – vector of types of the superimposed text for each segment with superimposed text,
2. “Names” matrix with dimensions N*M, where N is the number of segments with superimposed text, and M is the number of recognized Formula 1 driver; Each row in this matrix corresponds to recognized names of Formula 1 drivers in one segment, and
3. “Frames” matrix with dimensions N*2, where N is the number of segments with superimposed text; The first row of this matrix represents the starting frames for each superimposed sequence, while the second row shows the ending frames of superimposed text.

All these parameters are stored in such form in order to enable simple access and later usage for querying.

5.2. Learning and Inference Algorithms for DBNs

The probabilistic framework for processing of audio and video outputs will be presented in this section. We chose the DBN for this framework as it is shown to be the most powerful tool for fusing different uncertain clues.

DBN learning

Since we work with DBNs that have hidden states, we employed Expectation Maximization (EM) learning algorithm, which is based on the Maximum Likelihood (ML) learning algorithm. This algorithm is briefly described in the sequel.

If we denote unobserved variables with u_t, and observed with o_t, for time slice *t*, we can write probability distribution among all variables in a DBN as a:

$$\Pr(o_1,...,o_T,u_1,...,u_T) = \Pr(u_1)\Pr(o_1 \mid u_1)\prod_{t=2}^{T}\Pr(u_t \mid u_{t-1})\Pr(o_t \mid u_t). \quad (5.1)$$

If we denote model parameters as θ we can define this EM learning algorithm for DBNs as showed below. Here θ_i^* denote optimised model parameters.

EM algorithm:

get initial guess of θ_0;
do
infer hidden variables $\hat{u}_t$ from measurements $o_1,...,o_T$, using model θ_k;
$\theta_{k+1}^* = \arg\max_{\theta} \Pr(o_1,...,o_T,\hat{u}_1,...,\hat{u}_T \mid \theta)$;
until (convergence).

BN inference

Since DBN inference algorithms are based on BN inference algorithms, here we will present the algorithm used for inference in multiple connected Bayesian networks originally described by Lauritzen and Spieglhater in 1988 [26]. Various authors later improved this algorithm, and its final version can be found in [27]. We used two different techniques for the DBN inference algorithm, as we will se in the sequel.

The original inference algorithm consists of two parallel processes:

1. Graphical transformations of the original BN, and
2. Numerical computations for the transformed graph.

Graphical transformations

This process is based on one significant graphical representation property of a BN structure. This property is concerned with the directionality of Directed Acyclic Graphs (DAGs), representing BNs. It states that if we change the arc directionality it can result in changes in d-separation properties in the network, but the overall joint probability distribution will remain invariant. In that manner, we can consider those networks that only differ in arc directionality mutually equivalent.

Based on this assumption, some graphical operations are used for transforming the original DAG of a BN into a junction tree. Thus, we construct a singly connected tree instead of a multiply connected graph, which are basically identical. To perform this task we need to:

1. Moralize the network,
2. Triangulate the network,
3. Form the junction graph, and
4. Form the junction tree.

We will analyse each step on a BN depicted in Figure 5.1.

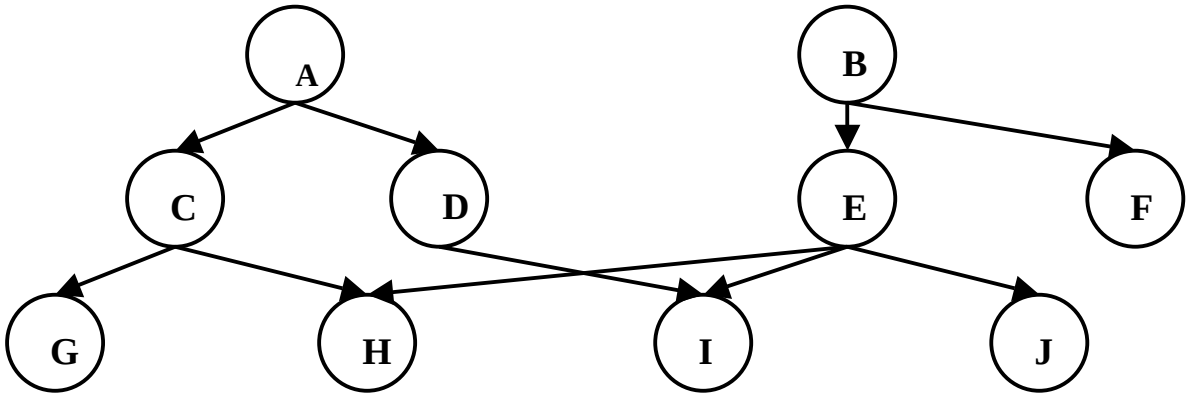

Figure 5.1 *DAG of Bayesian network*

By marrying co-parent nodes and dropping the directionality of all the arcs, we perform the moralization step. The marrying of co-parents is accomplished by connecting all the parents of the same node that are not already neighbours. As we can see in Figure 5.2 this operation can be easily performed.

"Maximum cardinality search" algorithm

Let assume that the graph is triangulated.
Set *i*:=1.
Set ***L***:={}.
Set ***N*** as a set of all nodes in the network
For each node *N* in ***N***, set *c*(*N*):=0
Do while ***L*** not equal to ***N***
    Set ***U***:=***N***-***L***.
    Select any *N* from ***U*** maximizing *c*(*N*) and number it *i* (*Ni*:=*N*).
    Set ***P****i*:=(***N***-*Ni*) intersected with ***L***.
    If ***P****i* is not a complete subgraph,
        Then graph is not triangulated.
    Else
        Set *c*(*W*)=*c*(*W*)+1for each *W* in ***N****Ni* intersected by ***U***.
    Set ***L***:=***L*** union {*Vi*}.
    Increment *i*.

In the triangulation step chords are added to each cycle of length greater or equal to four that does not already have a chord. This operation is performed until we process all cycles in the graph. The problem of graph triangulation is NP-hard. Therefore numerous approximate algorithms for this purpose have been proposed. For this purpose the algorithm described by Cowell et al. [28] is shown to be very useful. In this publication the other algorithms for performing BN inference can also be found. Here we will note that these algorithms are as well the basic algorithms that we used in our work.

However, the triangulation process needs not to be necessarily performed. Therefore, it can be useful to determine whether it is necessary to triangulate the graph. For this purpose "Maximum cardinality search" algorithm is used. This algorithm is described above.

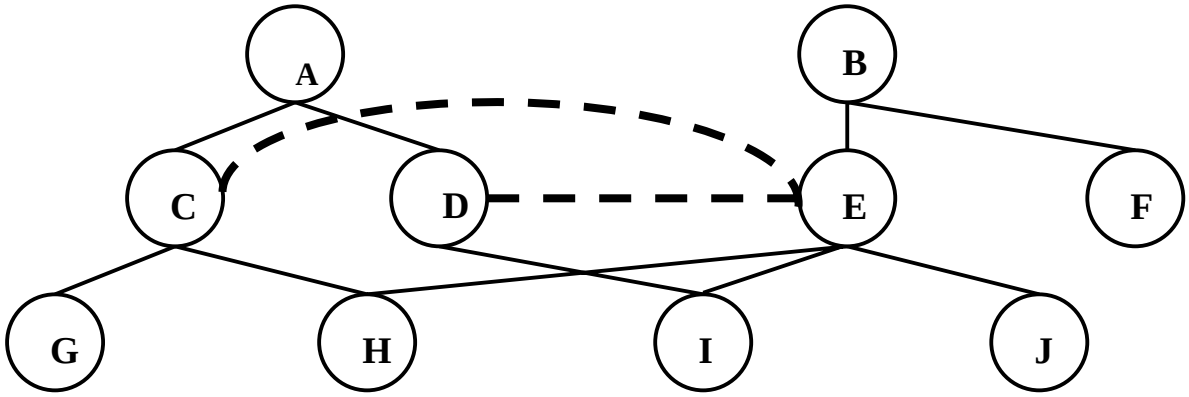

Figure 5.2 *Moralized graph*

Based on outputs of this algorithm we can now perform graph triangulation. For this purpose we used "One-step look ahead" triangulation algorithm, described in the sequel. This algorithm resulted in a triangulated graph, depicted in Figure 5.3.

"One-step look ahead" triangulation algorithm

Start with all nodes unnumbered.
Set i:=n, where n is the number of nodes in the graph.
Do until there are no more unnumbered nodes:
    Select an unnumbered node N that optimises the criterion $c(N)$.
    Number this node with i
    Form the set Ci consisting of Vi and its neighbours in the graph
    Add arcs between the nodes in Ci to make Ci a complete subgraph
    Decrement i

We can see that the ordering number of nodes in the graph depend on a criterion $c(N)$. Therefore the efficiency and the resulting triangulated graph will depend on this criterion. For this purpose we compared two approaches. In the first we used the weight obtained from the maximum cardinality search algorithm, and in the second one we used the weight of the triangulated graph. The weight of a graph is defined as a sum of the weights of each of its cliques, where the weight of a clique is the product of the weights of each of its nodes, and the weight of a node represents the number of values it can take on.

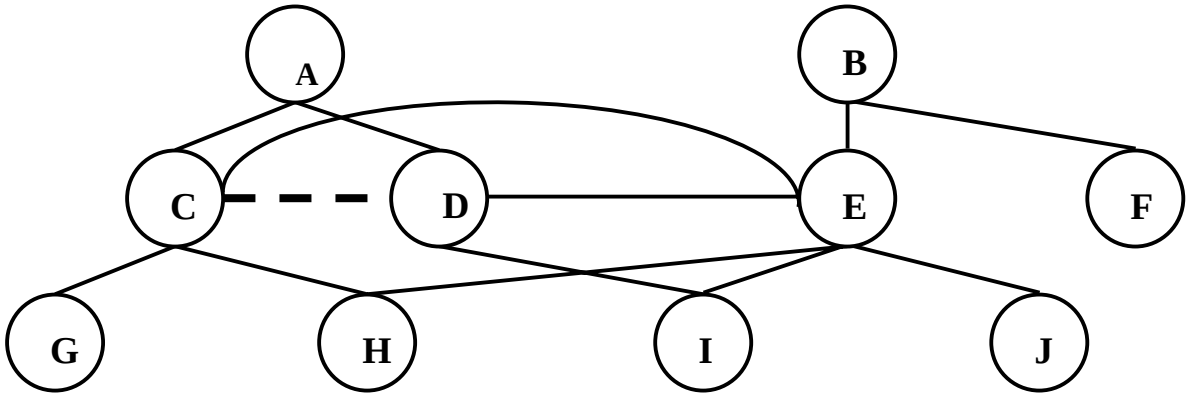

Figure 5.3 *Triangulated graph*

The ordering of nodes was obtained by choosing the node that will add the smallest number of new edges in the graph. If there is more than one possibility, we choose the node that induces the smallest clique.

Another useful property of the maximum cardinality search is that it produces the numbering of nodes that can enable efficient construction of junction trees with the running intersection property, as explained in [29]. Therefore here we will present algorithms for junction tree and junction graph construction based on maximum numbering obtained from maximum cardinality search.

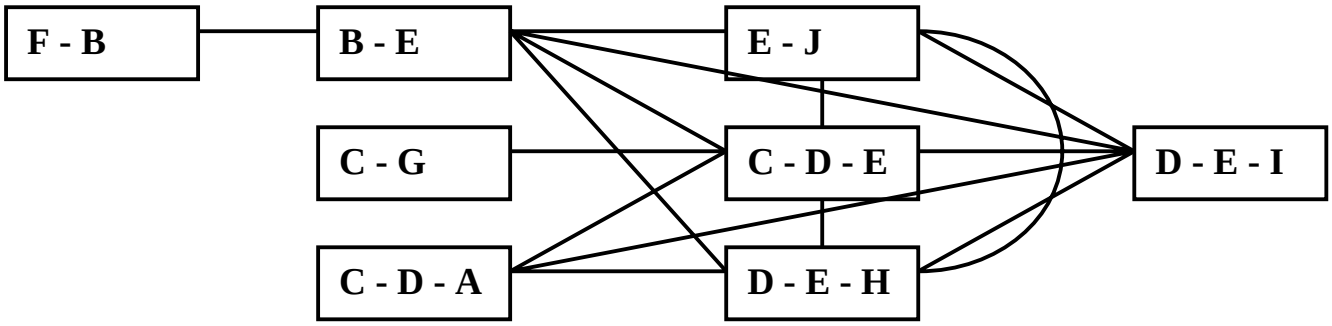

Figure 5.4 *The junction graph*

The third step of the BN inference algorithm identifies the cliques in the triangulated graph and forms a junction graph. In the junction graph nodes from the original graph usually appear in more than one clique (see Figure 5.4). To represent this property, an edge is added between two cliques if they have common nodes (i.e. their intersection is not empty). In next algorithms the cliques will be denoted as ***CL***. The procedure for finding cliques in a triangulated graph is as follows:

Algorithm for finding cliques in a triangulated graph

Start with the node numbering {*Ni*} and the sets {***P****i*} obtained by maximum cardinality search, where *i*=1,…,*k* .
Denote the cardinality of ***P****i* by *Pi*.
Consider *Ni* as a "ladder node" if *i*=*k* or *i*<*k* and *next*(*Pi*)<1+*Pi*.
Denote the *j*th ladder node, in ascending order, by *Yj*
Define the clique ***CL****j*={*Yj*} union ***P***(*Yj*).

These cliques are then used to form a junction tree. Forming of the junction tree is based on running intersection property: if any two cliques in the junction tree have common node *X* from the original network, then every clique on the path between those two cliques must also contain *X*. According to this rule we can form a junction tree from the junction graph (shown in Figure 5.5). Junction tree construction algorithm is described above.

Junction tree construction algorithm

Associate a node of the tree with each clique ***CL****i*.
For *i*=2:*p*
Add an edge between ***CL****i* and ***CL****j* where *j* is any value in {1,…,*i*-1} such that the clique ***CL****i* intersected by a union of *first*(***CL****i*) and *previous*(***CL****i*) represents regular subset of ***CL****j*.

However, we can modify the last steps of the DBN inference algorithm. The difference between these algorithms is in the forming of the junction tree.

Since the junction tree is formed to ensure that local evidence propagation retain the global consistency, a junction tree can be created by forming the maximal spanning tree of the junction graph. The best junction tree is one that minimizes the sum of the costs on each edge. The cost between two cliques is defined as a sum of weights on both cliques, where the weight of a clique represents the number of values a clique can take.

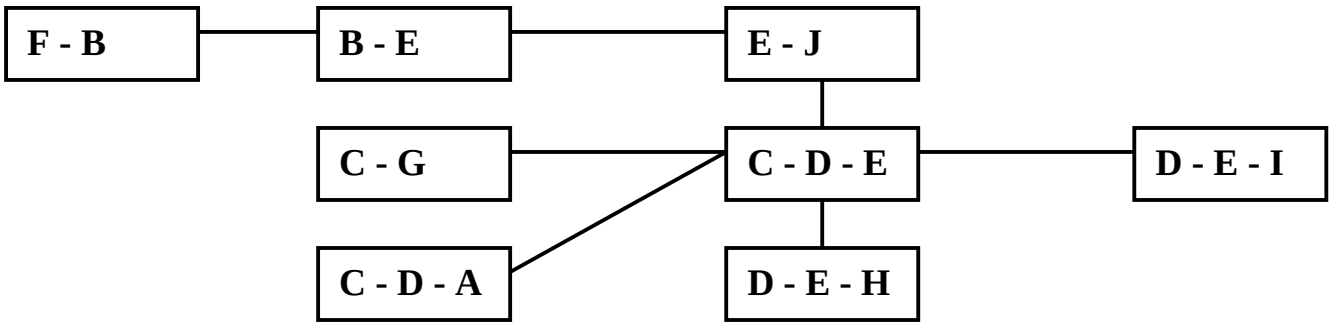

Figure 5.5 *A junction tree with running intersection property*

We conducted several experiments to compare the variations of the inference algorithm. Although the algorithm that uses junction tree construction based on numbering and node ordering obtained from "maximum cardinality search" is much less

computationally demanding the algorithm that uses the weight of the triangulated graph and the maximal spanning tree yielded much better results. Therefore, we choose to employ it for DBN inference.

Numerical computations

The numerical computations are similar to ones that can be performed for DAG of a non-transformed network. The difference is only that now we have cliques instead of nodes, and that we have to transform equations for probabilistic calculations in every clique and for each node in one clique. If we denote the set of cliques obtained from graphical transformations with *C*, we can transform equation for the joint probability distribution into next form:

$$P(X_1, X_2, ...X_n) = \prod_{i=1}^{n} P(X_i \mid Pa(X_i)) = \prod_{C_i \in C} P(C_i) \quad , \qquad (5.2)$$

where $P(C_i)$ is called clique marginal, and represents the joint probability distribution over the variables in the clique.

A difference from the exact BN inference calculations is that we allow the individual probabilities to exceed value of 1, and we do not take care of the restriction that the probabilities in a probability table or vector must sum to unity. Therefore, these probabilities are rather called beliefs, as we will term them in our work. Reason for absence of this restriction can be found in more efficient computations, which now do not include normalization step in the calculation process. Table that represents the belief distribution over each clique will be denoted with *BEL*, and termed as belief table. This table can be easily transformed into probability table, dividing every entry in the belief table by the normalization constant *Z*:

$$P(X) = Z^{-1} BEL(X), \qquad (5.3)$$

where *Z* represents the sum of the entries in $BEL(X)$. Now, we can transform equation (5.2) into next one:

$$P(X_1, X_2, ...X_n) \propto BEL(X_1, X_2, ...X_n) = \prod_{C_i \in C} BEL(C_i) . \qquad (5.4)$$

Cowell et al. [28], define the algorithm, which can be used for transformation of the recursive factorization in DBNs into potential representation. This potential representation can then be used for performing marginalisation on cliques. The algorithm is presented in the sequel.

Clique marginalisation algorithm

For each clique *Ci* in set *C* define the function *Fi*(*Ci*).
Initialise *Fi*(*Ci*):=1 for each *Ci*.
For each factor *P*(*X*|*Pa*(*X*)) in the recursive factorisation do:
Find one clique *Ci* containing both *X* and *Pa*(*X*) and redefine *Fi*(*Ci*):=*Fi*(*Ci*)*P*(*X*|*Pa*(*X*)).

Based on this potential representation we can define propagation on a junction tree. When one clique receives the evidence, it passes this evidence to its neighboring cliques. This process of distributing and absorbing evidence is termed as a calibration. The evidence is passed to the neighboring cliques by the nodes that are common to these cliques.

Therefore, belief propagation in junction trees can be described through two recursive methods: (1) *DistributeEvidence*, and (2) *CollectEvidence* [30]. The *DistributeEvidence* method pass the evidence to the neighboring cliques and ask them to calibrate, and then they recursively call same procedure for their neighbors. *CollectEvidence* has the opposite goal. Each clique calls this method in all the neighbors, and then calibrates itself using the values received from the neighboring cliques.

Based on the product rule, and above defined procedures belief propagation can be described with two sub processes. First is distributing evidence from one clique to another, and the second is absorption of multiple evidences.

To describe these processes we will use next notation. With $BEL^s(X)$ we will denote the belief distribution over variables in set of nodes *X* at step *s*. If two cliques CL_i, and CL_j are adjacent in the junction tree, their intersection will be denoted as:

$$S_{i,j} = CL_i \cap CL_j . \qquad (5.5)$$

$S_{i,j}$ is called the separator of CL_i and CL_j. The remainders of these cliques, called residuals, are defined as:

$$R_i = CL_i - S_{i,j}, \text{and } R_j = CL_j - S_{i,j} . \qquad (5.6)$$

We can now regard each clique as a union of the residuals and separators. In that manner a clique marginal can be defined by the product rule as:

$$BEL^s(CL_i) = BEL^s(R_i, S_{i,j}) = BEL^s(R_i \mid S_{i,j}) BEL^s(S_{i,j}) . \qquad (5.7)$$

If we perform marginalisation, based on previous equation we can compute the subjoint belief over the separators from the joint belief over the clique:

$$BEL^s(S_{i,j}) = \sum_{CL_i - S_{i,j}} BEL^s(CL_i)\,. \qquad (5.8)$$

We can see that the $BEL^s(S_{i,j})$ can be computed from both CL_i and CL_j. Therefore, we will notify this by a subscript index (for example $BEL_i^s(S_{i,j})$).

Since we can compute belief distribution of the common separator for two different cliques using one or the other clique, we say that these two cliques are consistent if $BEL_i^s(S_{i,j}) = BEL_j^s(S_{i,j})$. Otherwise they are considered to be inconsistent. Therefore, we can look at the process of propagation in a junction tree as a process of successive calibration of cliques in order to ensure consistency between cliques, as stated in [29].

When new evidence arrives, represented by $BEL_i^s(S_{i,j})$, we can rearrange equation (5.7) in order to make two adjacent cliques consistent. This will result in next equation:

$$BEL^{s+1}(CL_j) = \frac{BEL^s(CL_j)}{BEL_j^s(S_{i,j})} BEL_i^s(S_{i,j})\,. \qquad (5.9)$$

This term can now be used for evidence propagation from one clique to others, throughout the whole junction tree.

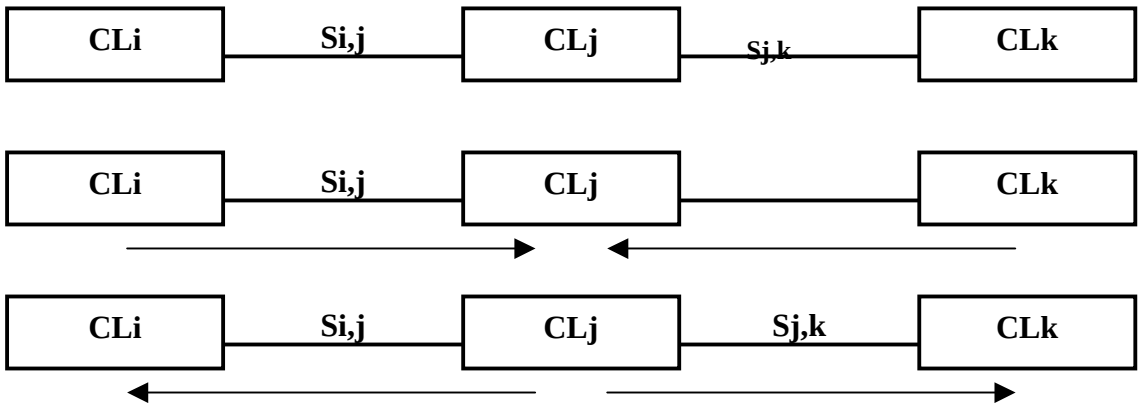

Figure 5.6 *Evidence arrives at cliques CLi and CLj, at time step s. CLj calibrates itself to both CLi and CLk at step s+1, and then CLi and CLk calibrate themselves to CLj at time step s+2.*

We can generalize this term to update the cliques when evidence arrives at more than one of their neighbors, as shown in Figure 5.6. We can describe this process as:

$$BEL^{s+1}(CL_j) = \frac{BEL^s(CL_j)}{BEL_j^s(S_{i,j})BEL_j^s(S_{j,k})} BEL_i^s(S_{i,j})BEL_k^s(S_{j,k})\,. \qquad (5.10)$$

Here, CL_j is being updated after receiving evidence from neighbouring cliques CL_i and CL_k. After this, clique CL_j will be inconsistent with its neighbours, because only this clique has received evidence and updated itself. Therefore, we must calibrate cliques CL_i and CL_k with clique CL_j:

$$BEL^{s+2}(CL_i) = \frac{BEL^s(CL_i)}{BEL_i^s(S_{i,j})} BEL_j^{s+1}(S_{i,j})\,,\ BEL^{s+2}(CL_k) = \frac{BEL^s(CL_k)}{BEL_k^s(S_{j,k})} BEL_j^{s+1}(S_{j,k})\,. \qquad (5.11)$$

As a last step of inference process, we need to calculate the posterior probability for one node (or possibly several of them) in the network. If we want to compute the posterior of the node *N*, we only need to find one clique that contains this node, and marginalize the $BEL(N)$ from it. After performing normalization operation on such computed value, we will receive designated posterior probability over node *N*.

DBN inference

The DBN inference algorithm is based on the BN inference algorithm. Here we will present the modified Boyen-Koller algorithm for approximate inference that we used in our work. This algorithm is originally presented in [31]. It model DBN as a sequence of BNs each representing two consecutive time slices. It extracts persistent state variables from the network. Persistent state variables are those variables whose value at time *t* directly or indirectly affects some variable at time *t*+1. Based on these variables we can form canonical set of state variables as a union of all the parents of variables from the second time slice. Eventually all variables in the canonical set will be persistent.

We also used the approximation of Boyen-Koller algorithm in our work. It is based on partitioning of state variables into clusters. Each cluster is formed of disjoint subsets of canonical set consisting of variables that do not influence each other in the same time slice. These clusters are formed to make nodes that belong to the same cluster classified in the same clique for the inference algorithm.

The definition of a cluster is a difficult problem and depends on a particular network structure. Usually clusters are formed small enough to enable efficient inference, and to preserve the fact that no edges between clusters in the same time slice exist. Some explanations about clustering process, and the benefits and drawback of its usage can be found in [32]. If the network structure is simple and we do not have the time limit, all nodes from one time slice are set as nodes that form one cluster ("exact" inference).

In this algorithm we use next notation: (1) X_i^t denotes the *i*th cluster with variables (nodes) at time slice *t*, (2) for belief state representation we use σ^t for time slice *t*, and (3) for the observed evidence at time slice *t* we use term ξ^t.

Modified Boyen-Koller algorithm for approximate inference:

1. Construct a clique tree from DBN unrolled for two consecutive time slices, requiring that every X_i^t and X_i^{t+1} be contained in full in at least one clique and using the junction tree algorithms for inference in Bayesian network presented in previous section.
2. Initialise each clique factor (defined in the clique marginalisation algorithm in the previous section) to the constant function (:=1).
3. Incorporate the conditional probability tables of the DBN into the appropriate factors. Let γ_0 be the non-calibrated junction.
4. For *t*=0,1,2,…
   - Let γ be a working copy of γ_0. Create σ^{t+1} as a new empty belief state.
   - For each *i*, incorporate the marginal $\sigma^t(X_i^t)$ in the appropriate factor of γ.
   - Incorporate the evidence ξ^{t+1} in γ.
   - Calibrate the potentials, using the computations described in the previous section.
   - Discard γ and report output as σ^{t+1}

5.3. Audio BNs and DBNs – Experiments and discussion

As we mentioned earlier, most of our experiments have been done on Bayesian and dynamic Bayesian networks constructed for the purpose of processing features extracted from the audio signal. We assume that it would be more appropriate to make experiments with a simpler audio network than employ a complex audio-video network. In such manner we were able to make more experiments and evaluate them, since the learning and inference algorithms take less time for processing a simpler network. The results that we obtained from experiments with audio BNs and DBNs will be presented below.

For BN/DBN learning and inference we employed algorithms presented in the previous section. For the learning we limited the maximum number of iterations to 10, and set the threshold for logarithmic difference to 10^{-3} in order to preserve the same experimental environment for each experiment. That means that we limited the time available for learning in order to conduct more experiments. These limitations will surely decrease the precision of obtained result. Therefore, we can exert that we would obtain better results if we increase the maximum number of iteration, and decrease the value of threshold for logarithmic difference. However, this would take us much more valuable time for experiments.

BNs vs. DBNs

We decided to start our experiments by comparing the results that can be achieved by employing BNs versus DBNs. Therefore we developed three different structures of BNs for processing audio clues, and one representative DBN structure for the same purpose. Different BNs are developed in order to explore how different network structures can influence the inference step in this type of networks. The structures of these BNs are depicted in Figure 5.1.

The query node in all these BNs is **Excited Announcer**, since we try to determine if the announcer raise his voice due to the interesting event that is taking place in the race. The shaded nodes represent evidence nodes, which receive their values based on features extracted from the audio signal of Formula 1 video.

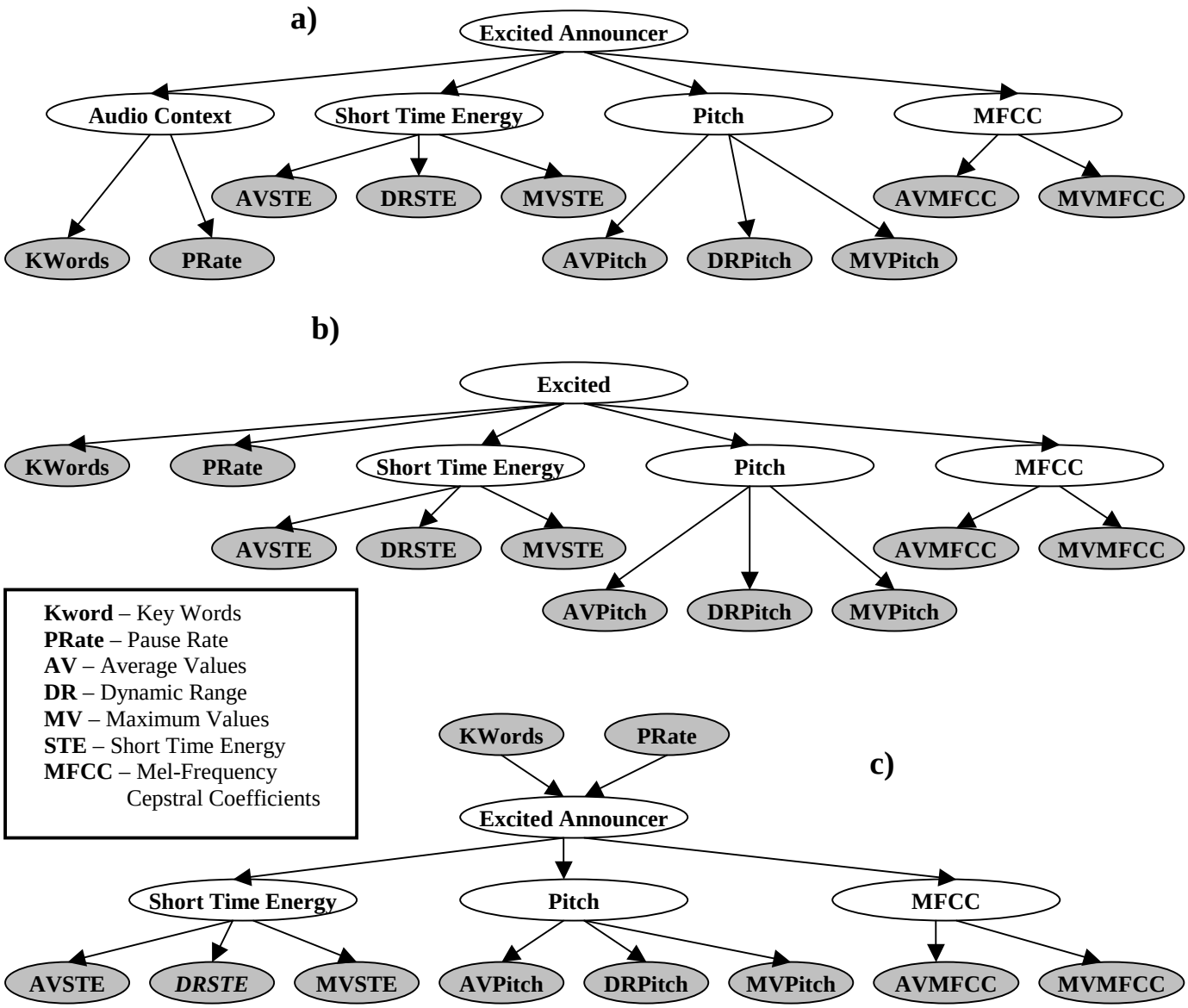

Figure 5.1 *Different structures of a Bayesian network for processing of audio features: a) Fully parameterised structure; b) Structure with direct influence from evidence to query node; c) Input/output BN structure*

As a representative DBN structure we construct one that has the same structure as the BN (shown in Figure 5.1.a) for one time slice. The temporal dependencies between nodes from two consecutive time slices of this network can be seen in Figure 5.2. For learning and inference algorithms we considered all nodes from one time slice as belonging to the same cluster ("exact" inference end learning).

We learn the BN parameters on a sequence of 300s, consisting of 3000 evidence values, extracted for every feature from the audio signal. For the DBN, we used the same video sequence of 300s, which was divided into 12 segments with 25s duration each.

The inference was performed on audio evidence extracted from the whole digitalized "Hockenheim" race, which is 1341s long. For each network structure we compute the precision and recall (defined in [33]). The precision is computed as:

$$P_r = \frac{1+card(RS \cap CS)}{1+card(RS)} \times 100 \,. \qquad (5.12)$$

Here, *RS* denotes set of segments recognized as excited speech segments, and *CS* represents the relevant set of correct segments.

For the calculation of the recall we used next term:

$$R_C = \frac{1+card(RS \cap CS)}{1+card(CS)} \times 100 \,. \qquad (5.11)$$

Note that we had to process the results obtained from BNs since the output values cannot be directly employed for distinguishing the presence and time boundaries of the excited speech. This can be seen in an example obtained from the BN, depicted in Figure 5.3. Therefore, we accumulated values of a query node over time to make a conclusion whether the announcer is excited.

However, the results obtained from the dynamic Bayesian network are much smoother (see Figure 5.4), and we did not have to process the output. We just employed the simple threshold to decide whether the announcer is excited. For experimental purposes we employed threshold value for query node of 0.5.

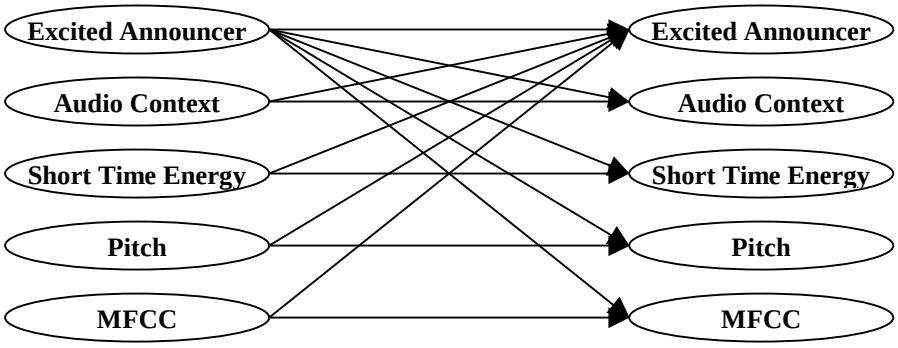

Figure 5.2 *Temporal dependencies for the" fully structured" dynamic Bayesian Network, composed for processing of audio features*

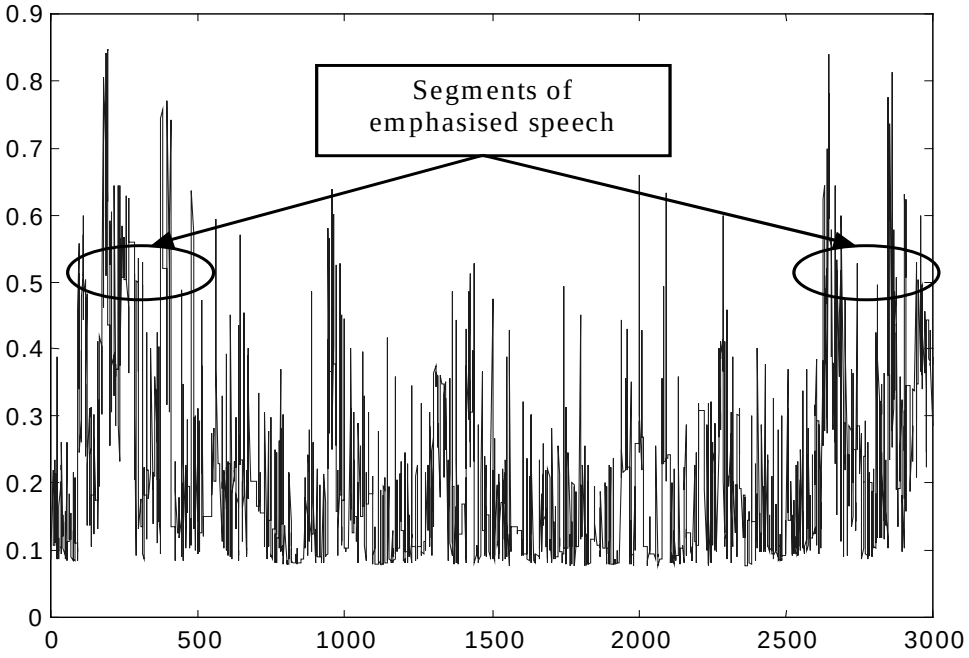

Figure 5.3 *Results of audio BN inference procedure for 300s long "avi" file*

The results from conducted experiments with previously described networks are shown in Table 5.2.

By comparing different BN structures we can see that there is no significant difference in precision and recall obtained from them. However, we can see that the "fully structured" BN structure yields the best results. That is the main reason why we decided to compare it with its upgrade, i.e. "fully structured" DBN. As we can see from the Table 5.2, the results from DBN inference are much better than ones received from BNs. To see whether those results are the best that we can obtain from the extracted audio-video parameters, we conducted several more experiments with DBNs that will be described in the sequel.

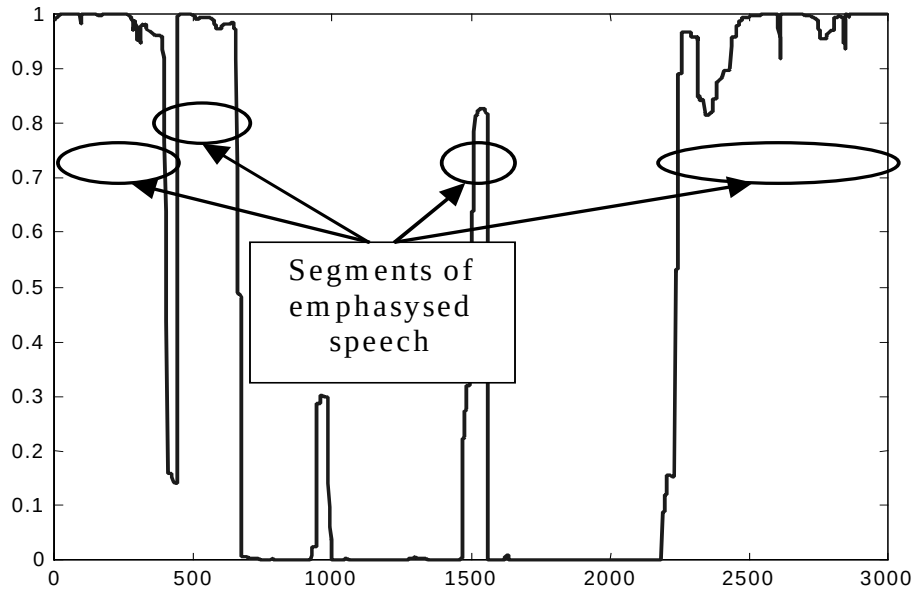

Figure 5.4 *Results of audio DBN inference for 300s long "avi" file*

Table 5.2 *Comparison of BNs and DBNs for detection of emphasized speech*

Used network structure	"Fully structured" BN (Figure 5.1a)	BN with direct evidence influence (Figure 5.1b)	Input/Output BN (Figure 5.1c)	"Fully structured" DBN (Figure 5.2)
Precision	**60.9 %**	**54.2 %**	**60.9 %**	**85.0 %**
Recall	**66.7 %**	**61.9 %**	**66.7 %**	**81.0 %**

Which properties can be changed in DBNs?

In this section, we will present the conclusions about the most appropriate DBN structure for our purpose, and we will discuss the most appropriate length of a learning sequence of evidence values. To investigate properties of different DBN structures we developed several different DBNs.

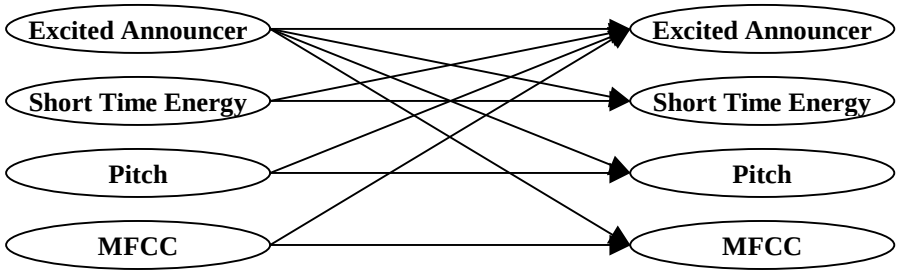

Figure 5.5 *Temporal dependencies for DBN structure with direct influence of observable nodes on query node*

First one has the same structure for one time slice as the BN depicted in the Figure 5.1.b. The temporal dependencies of nodes in this network are given in Figure 5.5. We use this network to show the influence that observable nodes have on query node if they are directly connected in the network structure for same time slice.

Other two network structures were developed to explore the influence that different temporal dependencies have on learning and inference procedures in DBNs. First of them is depicted in Figure 5.6. Here, all non-observable nodes distribute evidence to the query node in the next time slice, and only the query node receives evidence from the previous time slice. The second one uses the configurations where the query node do not distribute evidence to all non-observable nodes, but only to the

query node in the next time slice, as can be seen in Figure 5.7. Here, all other non-observable nodes pass their values to the corresponding nodes and the query node in the next time slice.

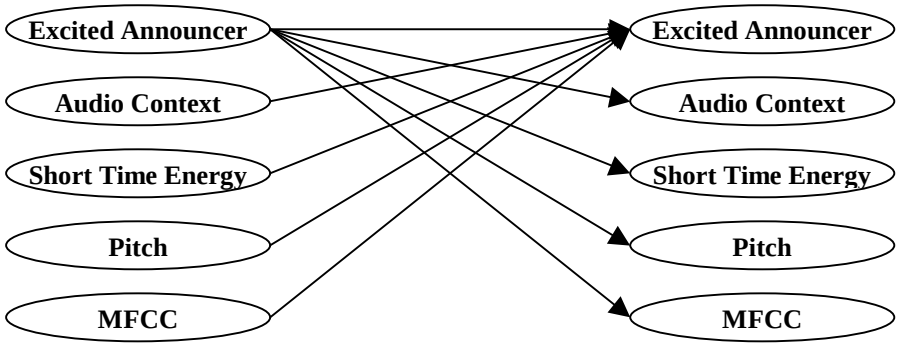

Figure 5.6 *Temporal dependencies for DBN structure with the query node as the only node for distributing the evidence to the nodes from the next time slice*

Results from experiments conducted on these three different structures can be seen in Table 5.3.

Table 5.3 *Precision and recall for different temporal dependencies in DBN*

Used network structure	"Fully structured" DBN (Figure 5.1a/5.2)	DBN with direct evidence influence (Figure 5.1b/5.5)*	DBN with "emission" query node (Figure 5.6)*	DBN with "collecting" query node (Figure 5.7)
Precision	**85 %**	**40.74 %**	**42.11 %**	**77.27 %**
Recall	**80.95 %**	**52.38 %**	**76.19 %**	**80.95 %**

***** *For these types of DBNs we had to process the output (connect the close time intervals)*

In addition we make some trials with different sizes of evidence vectors to see how this influences learning and inference algorithms. However, we keep the total length of the evidence used for learning the same. Except the standard learning parameters, i.e. 12 segments of 25s duration each, we used 6 segments of 50s duration, and 4 segments with 75s duration. All these experiments were conducted on the fully parameterized DBN.

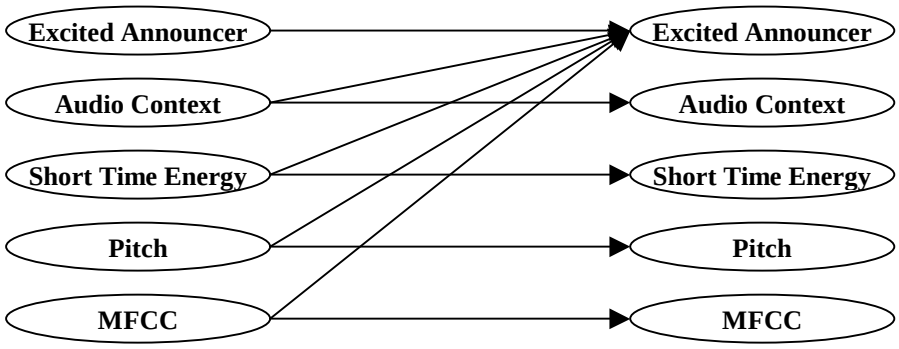

Figure 5.7 *Temporal dependencies for DBN structure with the query node distributing its evidence only to the query node in next time slice*

Precision and recall for this above described learning parameters that uses a different length of vectors for learning procedure are presented in Table 5.4.

Finally we make experiments with different clusters formed in the DBN structure. Since our network is relatively simple we made only one experiment with clustering. In this experiment we separate non-observable nodes from the other part of the network, as proposed by Boyen and Coller in [31]. In the original network, all nodes from one time slice are assumed to be in the same cluster. The difference between these two ways of clustering the DBN can be seen in Table 5.5. Evaluation was also performed for the "fully structured" DBN, with learning parameters: 6 sequences with 50s duration each.

Table 5.4 *Precision and recall for different length of learning vectors used in the DBN learning procedure*

Parameters for DBN learning procedure	**12 segments** **25 seconds** **250 time slices**	**6 segments** **50 seconds** **500 time slices**	**4 segments** **75 seconds** **750 time slices**
Precision	**85 %**	**86.36 %**	**54.55 %**
Recall	**80.95 %**	**90.48 %**	**85.71 %**

Table 5.5 *Precision and recall for different clusters used in the DBN learning/inference procedure*

Clustering technique	**One cluster per each time slice**	**One cluster for non-observable nodes**
Precision	**86.36 %**	**76 %**
Recall	**90.48 %**	**90.48 %**

After performing all these experiments we can make next conclusions:

1. From first group of experiments we conclude that the DBN learning and inference procedure depend a lot on the selected DBN structure for each time slice. We can see that this is not the case when we perform inference and learning in BNs.
2. As explained in [34], chosen temporal dependencies between nodes from two consecutive time slices have also strong influence on results of DBN inference. From Table 5.3 we can see that we can obtain the best result if we perform inference with the "fully structured" DBN.
3. If we compare the "fully structured" DBN with the DBN with "collecting" query node, we can see that exact representation of node dependencies results in a fewer number of misclassified sequences.
4. As for the best ratio *sequence duration/number of sequence* we can say that is better to have larger number of small sequences. However, there are an optimal number of sequences that is structure and time dependent and have to be found for each network structure and learning sequence.
5. We can see that the clustering technique did not bring significant drawback and error rate raise for the simple audio network, but it resulted in a larger number of misclassified sequences.

As a result, we decided to employ the "fully structured" DBN, with one cluster for nodes in same time slice, as the most powerful DBN structure for detection of emphasized announcer speech.

Evaluation of "Fully Structured" DBN for the Detection of Emphasized Speech

To evaluate the chosen network structure we employed it for detecting the emphasized speech from much longer sequences of evidence values. These evidence values are obtained from the audio signal of the Belgian and USA Grand Prix. Since these races had much less moments with excited speech, and much more noise (mostly car noises) we had to employ learning algorithms for longer sequence. Therefore, we employ it for a sequence of 10 minutes.

Table 5.5 *Evaluation results for audio DBN*

Race	**Spa Grand Prix**	**USA Grand Prix**
Precision	**76.66 %**	**75.86 %**
Recall	**79.31 %**	**80.77 %**

In Table 5.5, we can see the recall and precision obtained by employing the DBN inference algorithm for these two races, as an indication on usefulness of our system.

In next section we will evolve our selected network to more complex structure by incorporating video features.

5.3 Audio/Video DBNs – Experiments and Evaluation

The audio DBN can only extract segments of the Formula 1 race where the announcer raises his voice. Other interesting segments, which were missed by the announcer, could not be extracted. Therefore, the employment of the audio DBN would lead to a lower precision and recall for highlight extraction from Formula 1 race (about 50%).

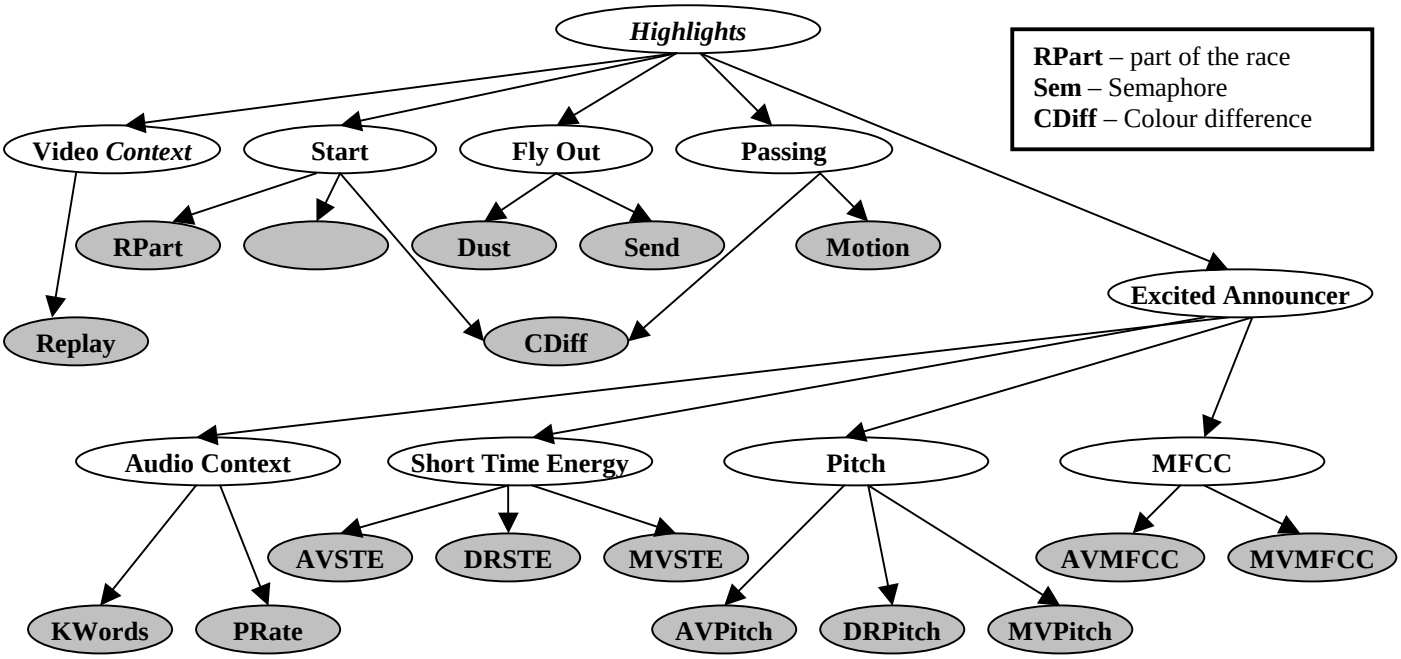

Figure 5.8 *"Asymmetric" Audio/Video dynamic Bayesian network for one time slice*

To improve the results obtained solely from audio clues we developed audio/video DBNs. We decided to explore the possibility to construct general audio/video DBNs for highlight detection. In the sequel we will present the results obtained from the general DBN depicted in Figure 5.8 and 5.9.

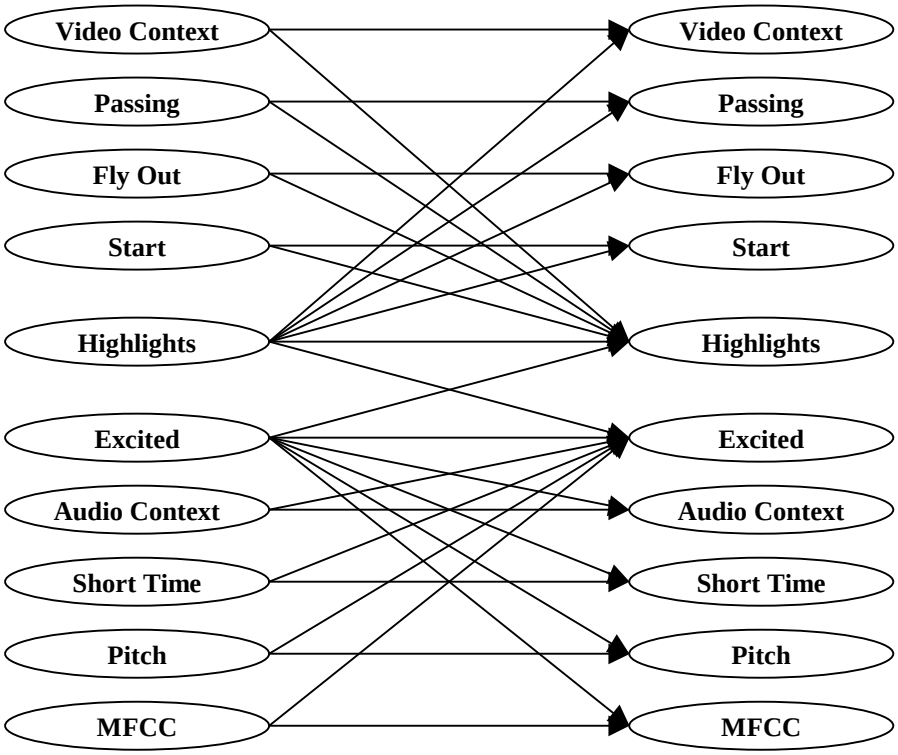

Figure 5.9 *Temporal dependencies for audio/video dynamic Bayesian Network*

The structure that represents one time slice of the audio/video DBNs is depicted in Figure 5.8. The **Highlights** node is chosen to be the main query node, while we will also query nodes **Start**, **Fly Out**, and **Passing** in our experiments. Chosen temporal dependencies between nodes in this network are shown in Figure 5.9.

Experiments were done similarly as for the audio DBNs. We employed again the learning algorithm on 6 sequences with 50s duration each. The results obtained by applying audio/video DBN to "Hockenheim" race are shown in Table 5.7. The precision and recall for highlights are calculated based on a probability threshold for output of 0.5, and minimal time duration of 6s.

The values of the other query nodes are calculated based on a value of main query node. We calculated the most probable candidates during the each "highlight" segment, and pronounce it as a start, fly out, or passing based on values of corresponding nodes. For segments longer than 15s we perform this operation on every 5s to enable multiple selection in a "highlight" segment.

Table 5.7 *Audio/Video DBNs*

Audio/Video DBN		**"Asymmetric"**
Highlights	**Precision**	**83.72 %**
	Recall	**85.71 %**
Start	**Precision**	**83.33 %**
	Recall	**100 %**
Fly Out	**Precision**	**63.64 %**
	Recall	**77.77 %**
Passing	**Precision**	**78.57 %**
	Recall	**50 %**

Table 5.8 *Evaluation results for audio/video DBN*

Audio/Video DBN		**Spa Grand Prix***	**USA Grand Prix**
Highlights	**Precision**	**42.86 %**	**73.33 %**
	Recall	**52.94 %**	**75.57 %**
Start	**Precision**	**100 %**	**100 %**
	Recall	**66.67 %**	**50 %**
Fly Out	**Precision**	**100 %**	**0** %
	Recall	**36.36 %**	**0** %

** These results are obtained by the full audio/video DBN, which includes the passing subnetwork*
*** There were no fly outs in the USA Grand Prix*

The supplemental query nodes are incorporated in the scheme in order to classify different interesting events that takes place in the Formula 1 race. We can see from Table 5.7 that we gained high accuracy for highlights and start, while the accuracy for fly out and passing were a little bit lower. Main reason for this is that we used very general and less powerful video clues for fly out, and especially passing.

We performed evaluation of this network structure on Belgium and USA Grand Prix, but we had a big decrement in our results, mostly because of the "passing" part of the network. Therefore, we simplified the overall audio/video network, and excluded the "passing" sub-network. A significant difference in results obtained with and without the passing sub-network is presented in Table 5.8.

Here, we can see that the precision and recall for highlights, fly outs, obtained from the audio/video DBN, are not so high as ones for excited speech detection solely based on the audio network. This is mostly due to the fact that we used less powerful clues in the video part of the network. However, using the audio/video DBN we can extract much more interesting events from the race than suing the audio network.

6. Our System in Practice

Up until now we described the framework of our system, along with all the algorithms used for various purposes. In this section, we will present the outputs of our system, and describe how the system can be used for querying the multimedia database.

We extracted the metadata and associated it with multimedia documents based on outputs from dynamic Bayesian networks presented in the previous sections, and text detection and recognition algorithms. Therefore, we enable two types of multimedia querying tasks to be fulfilled: (1) based on a recognized superimposed text, and (2) based on DBNs output.

Metadata extracted from the superimposed text is stored like explained in section 5.1. Therefore we enable browsing of multimedia documents based on the list of types and names extracted from Formula 1 race. This metadata will enable a user to query database for textual information, or for a specific part of the race. Example textual retrieval queries are: race winner, classification in the *i*th lap, driver position in the *i*th lap, relative positions of two drivers in the *i*th lap, etc. Examples of the video retrieval based on recognized superimposed text are: specified driver in the pit stop, race finish, final lap, all segments where driver is signed, etc.

The types of the superimposed text and names extracted from the Formula 1 races are listed in Table 5.9.

To show the querying power of our system, in the sequel we will give some query examples (in descriptive form):

1. "Retrieve all video sequences with Michael Schumacher"
2. "Retrieve all video sequences with Michael Schumacher leading the race"
3. "Retrieve a video sequence where Michael Schumacher is first, and Mika Hakkinen is second"
4. "Retrieve the video sequences where Barrichello is in the pit stop"
5. "Retrieve the sequences with the race leader crossing the finish line"
6. "Retrieve all fly outs with superimposed text"

These are only several of a large number of possible queries, which give an impression of great possibilities of our powerful system.

Table 5.9. *Types of superimposed text, and names extracted from the superimposed text*

Types	Names	
0 – no type associated 1 – classification of first eight drivers 2 – number of remaining laps (with the name of the race leader) 3 – safety car 4 – race stopped 5 – pit stop 6 – final lap (with the name of the race leader) 7 – winner (race finish) 8 – driver signature	0 – no names 1 – Michael Schumacher 2 – Barrichello 3 – Hakkinen 4 – Coulthard 5 – Montoya 6 – Ralf Schumacher 7 – Raikkonen 8 – Irvine	9 - Trulli 10 – Panis 11 – Villeneuve 12 – Verstappen 13 – Bernoldi 15 – Zonta 16 – Fisichella 17 – Burti 18 – Alesi

Metadata obtained from DBNs is stored as a vector of probabilistic values (0 to 1) for each segment of 0.1s duration in the race. Therefore we enabled a query example that has form like this:

Segments = FindSegments{ raceName, outputType, probabilityLevel, durationLevel, boundLevel },

where ***Segments*** will contain the starting and ending frame numbers that represent the query output. The ***raceName*** parameter specifies the name of a race. The data about the information the user is interested in is stored in ***outputType***. This parameter is used to distinguish witch outputs from different DBNs we want to use, i.e. the DBN for highlights, for excited announcer, for start, for fly out, or for passing. We set the threshold from 0.1 to 0.95 for probability in ***probabilityLevel*** parameter. The ***durationLevel*** parameter specifies the minimal duration of a selected segment (from 1s to 20s), and we specify the duration of retrieved part of the multimedia document using the parameter ***boundLevel***. We can retrieve only the part that was selected by a DBN, or the whole shot based on performed shot segmentation.

Based on the metadata obtained from DBNs and text recognition, we can construct many more query examples, which can be easily fulfilled. However, to make the most appropriate query set we must investigate the statistics about the information that people want to receive from such systems, but this is a direction for the future work.

7. Conclusions

After all experiments with the audio/video feature extraction and finally DBNs, we have to summarize the benefits and usefulness of our work. Therefore, we will explain shortly what we have done, and what are the benefits and drawbacks of our system.

First, we must exert that this is the first time to the best of our knowledge that dynamic Bayesian networks are used for indexing and characterization of multimedia documents. A very important aspect of our work is that we extracted different multimodal clues from a multimedia document and fused them using DBNs. As can be seen in the previous sections, the benefits of using these multimodalities, together with DBNs are priceless.

We can clearly see that the usage of three different clues from multimedia documents led us to much better characterization of Formula 1 races. The audio subsystem was able only to detect half the number of all interesting segments in the race, while the integrated audio/video system was able to correct the results obtained from the audio part, and detect almost 80% of interesting segments in the race. However, this audio part is still useful if we want only the segments with excited announcer speech, where it showed satisfying recognition accuracy. By integrating the superimposed text, audio and video subsystems we built a powerful tool for indexing the Formula 1 races videos.

In this work we also presented some novel algorithms, as well as improvements of the existing algorithms and their modification. These algorithms are used for the feature extraction and text recognition from multimedia documents, and for the inference and learning in DBNs. In particular, we developed a robust method for the audio feature extraction, based on the speech endpoint detection, and predetermined frequency bounds for each audio feature. We analyzed different spectrum of the audio signal for the speech endpoint detection and feature extraction, and find the most appropriate thresholds. We found that Mel-frequency cepstral coefficients can be very useful for the detection of emphasized speech.

For the video analysis we developed several simple algorithms for extraction of specific content from video, such as motion, semaphore, dust, send, etc. We transformed these parameters into probabilistic values and use them as one of the clues in audio/video DBNs. We also developed a high accuracy shot change detection algorithm.

For the text detection and recognition task, we presented a new technique for text detection based on properties of Formula 1 race videos, and modified the existing text recognition algorithm for our purpose. We implement double vertical refinement for better text extraction. We are able to extract characters, and form the word connecting them, and recognized these words with almost 100% accuracy.

Finally, we made numerous experiments with different DBN and BN structures, and compare two different DBN learning algorithms. We showed the advantage of DBNs over BNs for our application. In our experiments we also showed the unstable nature of DBNs.

We also explored the influence of different atemporal and temporal connections within the network, as well as clustering techniques. We found that the exact representation of temporal dependencies is the most powerful for DBN learning and inference. We showed that the structure and connection within a DBN has strong influence on the learning and inference procedures. The learning sequence also showed to be structure dependent. Therefore, each DBN has its own peak for segments/duration ratio.

As a result of all these experiments we pick one model for extraction of excited announcer speech segments, and the other for highlights, start, fly out, and passing segments in a race.

6.1. Problems encountered

During our work we became aware of many problems that are present in the task of multimedia document indexing. Our domain was very complex. Furthermore, we can conclude that we had to process one of the most difficult sport videos among all. This is mostly because of the realization and capturing process of Formula 1 race.

As one of the most difficult problem that we had to overcome is the complex and ambiguous sound in Formula 1 race. As the biggest problem we will isolate the roaring of car engines, especially when the "in car" camera is on. Also, the problems are: variation of volume level during the race, numerous on-line pit reports with low quality sound, on-line driver's comments during the race, etc. This was the main reason for lower DBN inference accuracy for "Spa" and "USA" race.

For video analysis main problems are related to the complex camera motion and various digital effects used during the race. For capturing Formula 1 race more that 10 cameras are usually used. All of them capture the race from different angles and distances from the track. Also, there is a lot of camera movement (such as zoom, pan, etc.).

We have another problem of different nature during our work. It was the limited time. Every sub process in our system (audio analysis, video analysis, text recognition, DBN inference, and DBN learning) consists of several smaller actions, and generally is time consuming, and computationally demanding. Just to see how demanding each of these processes is, we will present the estimated time needed for processing the multimedia files for video and audio feature extraction, text recognition, and DBN learning and inference in the Table 6.1.

Table 6.1 *Processing time for audio and video feature extraction, text recognition DBN learning and inference for 1 minute duration of multimedia file*

Platform **Pentium II** **800MHz** **256MB RAM** **Windows 2000**	**Audio features**	**Video features**	**Text recognition**	**DBN learning**	**DBN inference**
	180 min. (3 hours)	**40 min.**	**120 min. (2 hours)**	**60 - 120 min. (1 – 2 hours)**	**10 – 15 min.**

Even the video analysis is the most complex and time consuming process it took less time than audio analysis in our work, since we decided to use less computationally demanding video clues. However, this resulted in low accuracy in audio/video DBNs for passing, as we mentioned in the previous section.

6.2. Future work

Although we have already presented significant amount of work done to enable indexing and characterization of multimedia documents of Formula 1 race, we state that there are still many areas where improvements can be done. In the sequel we will list most of them.

As for the audio part of our system we can improve the filtering technique by employing finer frequency filter bounds, and employ more powerful algorithms for extracting pitch feature. There are some audio features that need to be investigated (spectral energy, band energy ratio, atc.), which we did not use in our experiments. We can also utilize the outputs of the keyword spotting system by grouping the recognized words into sets for each interesting event in the race, and associated them with each query node in the DBN structure. This would help us to categorize the events better. We could even train the keyword spotting system for a particular announcer for Formula one race, which will increase the accuracy of the keyword spotting system.

In the video subsystem we only use the simplest features. There is a large area in computer vision, which needs to be investigated and possibly integrated in our system. The problem of detecting and tracking moving objects supplemented with a fast camera movement is challenging and still not resolved problem in computer vision. Since it take a lot of time for investigation, testing, and implementation, we leave it for future and computer vision experts.

We made a lot of experiments with DBNs. However, there are still some matters that have to be investigated and improvements that can be done. One is concerned with learning algorithms in DBNs, such as boosting or error feedback DBN learning algorithms ([28]). Also we did not made experiments with more complex DBN structures. But, we can say that the structures and algorithms we used for DBN learning and inference, are near optimal.

Although we did not employ the most powerful algorithms for every segment in our robust F1HD system, it yielded very good results. With the mentioned improvements of the video part, our algorithm would even gain more power that it already does. Therefore, we hope that our work give a clear direction for the future researchers in the multimedia retrieval community.

8. References

[1] C. Berzuini, "Representing Time in Causal Probabilistic Networks," Uncertainty in Artificial Intelligence vol. 5, pp. 15-28, Elsevier Science Publisher B.V., 1990.

[2] A. Singhal, C. Brown, "Dynamic Bayes Net Approach to Multimodal Sensor Fusion," Proceedings of the SPIE – The International Society for Optical Engineering, 3209-32020, October 1997.

[3] M. Petkovic, W. Jonker, "Overview of Data Models and Query Languages for Content-based Video Retrieval", *SSGRR International Conference*, L'Aquila, Italy, 2000.

[4] S. Intille, A. Bobick, "Visual Tracking Using Closed-Worlds", *Tech. Report No. 294*, M.I.T. Media Laboratory, 1994.

[5] Y. Gong, L. T. Sin, C. H. Chuan, H-J. Zhang, M. Sakauchi, "Automatic Parsing of TV Soccer Programs", In *Proc. of IEEE International Conference on Multimedia Computing and Systems*, Washington D.C., 1995, pp. 167-174.

[6] N. Haering, R.J. Qian, M.I. Sezan, "A semantic event-detection approach and its application to detecting hunts in wildlife video", *Circuits and Systems for Video Technology, IEEE Transactions on*, 10(6), Sept. 2000, pp. 857-868.

[7] Y. L. Chang, W. Zeng, I. Kamel, R. Alonso, "Integrated Image and Speech Analysis for Content-Based Video Indexing," Proceedings of 3rd IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan, pp. 306-313, June 1996.

[8] M. Naphade, T. Kristjansson, B. Frey, T.S. Huang, "Probabilistic multimedia objects (multijects): A novel approach to indexing and retrieval in multimedia systems", In *Proc. of the IEEE ICIP*, Chicago, IL, 1998, vol. 3, pp. 536-540.

[9] N. Vasconcelos, A. Lippman, "Bayesian Modeling of video editing and structure: Semantic features for video summarization and browsing", In *Proc. of the IEEE ICIP*, Chicago, IL, 1998, vol. 2, pp. 550-555.

[10] A.M. Ferman, A. M. Tekalp, "A Unified Framework for Probabilistic Analysis and Extraction of Video Content," Proceedings of International Conference on Image Processing, vol. 2, pp. 91-95, Kobe, Japan, October 1999.

[11] T. Syeda-Mahmood, S. Srinivasan, "Detecting Topical Events in Digital Video", In *Proc. of ACM Multimedia*, Los Angeles, CA, 2000, pp. 85-94.

[12] M. Naphade, T. Huang, "A Probabilistic Framework for Semantic Indexing and Retrieval in Video", In *Proc. of the IEEE Intl. Conf. on Multimedia and Expo (ICME)*, New York, 2000, vol. 1, pp. 475-478.

[13] Y. Rui, A. Gupta, A. Acero, "Automatically Extracting Highlights for TV Baseball Programs," Poceedings of the 8th ACM International Conference on Multimedia, pp. 105-115, Los Angeles, CA USA, October-November 2000.

[14] V. Kobla, D. De Menthon, D. Doermann, "Identifying Sports Videos Using Replay, Text, and Camera Motion Features," Proceedings of the SPIE Conference on Storage and Retrieval for Media Databases, vol. 3972, pp. 332-343, January 2000.

[15] M. Petkovic, W. Jonker, "Content-Based Video Retrieval by Integrating Spatio-Temporal and Stochastic Recognition of Events," IEEE International Workshop on Detection and Recognition of Events in Video, Vancouver, Canada, July 2001.

[16] B. Arons," Pitch-Based Emphasis Detection for Segment Speech Recordings," Proceedings of International Conference on Spoken Language Processing, vol. 4, pp. 1931-1934, Yokohama, Japan, September 1994.

[17] L. Rabiner, B.H. Juang, "Fundamentals of Speech Recognition," Prentice Hall, Englewood Cliffs, New Jersey 07632, 1993.

[18] M. Petkovic, "Content-Based Video Retrieval," 7th Conference on Extending DataBase Technology, Ph. D. Workshop, Konstanz, Germany, March 2000.

[19] Y. Wang, Z. Liu, J. C. Huang, "Multimedia Content Analysis Using Both Audio and Visual Clues," IEEE Signal Processing Magazine, pp. 12-36, November 2000.

[20] F. Idris, S. Panchanathan, "Review of Image and Video Indexing Techniques," Visual Community Image Representation, vol. 8, pp. 146-166, June 1997.

[21] Y. Rui, T. S. Huang, S. F. Cheng, "Image Retrieval: Current Technologies, Promising Directions, and Open Issues," Visual Community Image Representation, vol. 10, pp. 39-62, March 1999.

[22] S. Srinivasan, D. Petkovic, D. Ponceleon, "Towards Robust Features for Classifying Audio in the CueVideo System," Proceedings of the 7th ACM International Conference on Multimedia, pp. 393-400, Orlando, FL USA, October-November 1999.

[23] J. Christie, "Completion of TNO-Abbot Research Project," Cambridge University Engeneering Department, Cambridge, England, December 1996.

[24] N. Babaguchi, Y. Kawai, Y. Yasugi, T. Kitahashi, “Linking Live and Replay Scenes in Broadcast Sport Video,” Proceedings of ACM Multimedia 2000 Workshops, pp.205-208, Marina del Rey, Celifornia, November 2000.

[25] T. Sato, T. Kanade, E. K. Hughes, M. A. Smith, “Video OCR for Digital News Archives,” IEEE International Workshop on Content-Based Access of Image and Video Databases, pp. 52-60, January 1998.

[26] S. Lauritzen, D. Spiegelhalter, “Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems,” Journal of the Royal Statistical Society Series B, 50, pp.157-224, 1988.

[27] F. Jensen, S. Lauritzen, “Probabilistic Networks,” Algorithms for Uncertainty and Defeasible Reasoning, vol. 5, pp.289-320, Kluwer Academic Publishers, 2000.

[28] R. Cowell, A. David, S. Lauritzen, D. Speigelhalter, “Probabilistic Networks and Expert Systems,” New York, NY, Springer.

[29] M. J. Mayo, “Bayesian Student Modeling and Decision-Theoretic Selection of Tutorial Action in Intelligent Tutoring System,” PhD Thesis, University of Centerbury, 2001.

[30] F. Jensen, S. Lauritzen, K. Oleson, “Bayesian Updating in Causal Probabilistic Networks by Local Computations,” Computation Statistics Quarterly, 4, pp. 269-282, 1990.

[31] X. Boyen, D. Koller, “Tractable Inference for Complex Stochastic Processes,” Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, 1998.

[32] http://robotics.stanford.edu/~xb/uai98/uai98slides/slide_first.htm

[33] V. S. Sbrahmanian, “Principles of Multimedia Database Systems,” Morgan Kaufmann Publishers, San Francisco, California1998.

[34] V. Mihajlovic, M. Petkovic, “Dynamic Bayesian Networks: A State of the Art,” Technical Report, TR-CTIT-01-35, October 2001.

[35] J. Droppo, A. Acero, “ Maximum a Posteriori Pitch Tracking,” Proceedings of ICLSP’98, pp. 943-946, 1988.

[36] V. Pavlovic, A. Garg, J. M. Rehg, “Multimodal Speaker Detection using Input/Output Dynamic Bayesian networks,” International Conference of Multimodal Interfaces, Beijing, China, October 2000.

Appendix A List of abbreviations

List of abbreviations	
AI	Artificial Intelligence
“avi”	Audio-video format
BNs	Bayesian Networks
CC	Connected Component
CPT	Conditional Probability Table
DAG	Directed Acyclic Graph
DBNs	Dynamic Bayesian Networks
DVEs	Digital Video Effects
EM	Expectation-Maximization
F1HD	Formula 1 Highlight Detection
FFT	Fast Fourier Transformation
FSG	Finite State Grammar
FSGD	Finite State Grammar Decoder
HMMs	Hidden Markov models
HSV	Hue, Saturation, and Value
MAP	Maximum A Posterior
MFCCs	Mel-Frequency Cepstral Coefficients
MPEG	Moving Pictures Experts Group
NP	Non-deterministic Polynomial
OCR	Optical Character Recognition (OCR)
RGB	Red, Green, and Blue
STE	Short Time Energy
VOCR	Video Optical Character Recognition