UPTEC F05 085
Examensarbete 20 p
December 2005

Automatic Content-Based Filtering of Television News

Johan Ljung

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Automatic Content-Based Filtering of Television News

Johan Ljung

With the ever-increasing flow of information, the need for computer-automated tools for handling information becomes greater and greater. News shows and other television broadcasts carry vast amounts of information, but generally in forms that are not reachable through conventional information retrieval techniques. During the last couple of years, researchers around the world have given considerable attention to the problem of extracting semantic information from television and re-representing it in a form suitable for automatic indexing and searching. Detailed content information in a television broadcast is typically found in teletext subtitles (if such are available), text embedded in the video image, and in the spoken dialogue. Extracting it involves using techniques associated with signal processing, image analysis, artificial intelligence, speech recognition, etc. A reliable filter system for use in e.g. Scandinavia, where only a few of the television broadcasts are teletext-subtitled, must take advantage of information in all three forms, or modalities. The presented report contains a survey of current research projects in this area and a theoretical design of a modular, multi-modal, content-based television filter, based on findings in the survey. A prototype of a module for extracting and recognizing image-embedded text has been designed and implemented in MATLAB. The prototype operates on DCT-compressed video frames (e.g. JPEG, MPEG) and uses statistics, heuristics, image processing and neural network technology. When applied to subtitles embedded in the image, the prototype outputs recognized text with a total character error rate of 5%.

Handledare: Daniel Grönquist & Taquin Ho
Ämnesgranskare: Ewert Bengtsson
Examinator: Tomas Nyberg
ISSN: 1401-5757, UPTEC F05 085
Sponsor: Observer Sverige AB

Preface and Acknowledgements

This report describes a thesis project for the degree of Master of Science in Engineering Physics with Systems Engineering at Uppsala University. The project was carried out during an internship at Sifo Group IT&T in Solna/Stockholm, with funding from the Observer group. The project has been supervised by Daniel Grönquist and Taquin Ho, then at IT&T. The original academic supervisor / examiner was Dr. Torsten Jarkrans, a role later taken over by Professor Ewert Bengtsson, both with the Centre for Image Analysis, Uppsala University.

The original internship spanned 20 weeks during the second half of 1997 and was extended by a couple of weeks into the early spring 1998. During those weeks, the subject area was researched, the prototype was developed and evaluated, the results were presented to Observer management, and all of the thesis report, but portions of some background paragraphs, was written. However, the report was not quite completed until 2005. Sections 1-7 of this report present the project as it was carried out in 1997, based on current research and development at that time. A new section, 8, has been added in 2005, discussing the development within the subject area in the intermediate time, and its implications for the findings of this project.

The total amount of work performed on the project corresponds to approximately 27 full-time weeks. Of these, approximately 10 weeks were spent on literature studies, 10 on prototype design and implementation, and 7 on documentation. The first 25 were spent in 1997-98, one week was spent in May 2005, tying up the loose ends of the "original" report, and one in the following summer, researching and writing the new section on development 1998-2005.

I would like to thank *Dr. Jarkrans* for his helpful suggestions at the on-set of this project, and *Professor Bengtsson* for his kind support facilitating its conclusion; *Hans Strand*, formerly managing director of Observer Sweden, for making the project possible; *Ann-Sofi Korsman* and *Lars Pyk*, then heads of Observer's RTV department, for providing me with valuable information, video material and equipment; *Annica Forss* at Sveriges Television, Stockholm, for providing me with material on teletext subtitling in Sweden; *Professor Yasuo Ariki* at Ryukoku University, Seta, Japan, and *Mr. Andrew Merlino* at Mitre Corp., Bedford, MA, for kindly answering my questions and sharing their research results with me. Last, but not least, I would like to express my deepest gratitude for the help and support of my supervisors *Daniel Grönquist* and *Taquin Ho*, my "co-intern" *Magnus Karperyd* and all the others then at IT&T: *Tomas Elison, Mille Eriksson (Bessö), Anna Lagerås, Thomas Mattisson, Jens Millgård, Georg Sandholm* and *Tobias Stenälv*.

Contents

1 Introduction

Imagine a computer system which would be able not only to record TV programs for you, but to actually "watch" and "listen" to them. You would tell the system what issues or topics you are interested in, and the system would automatically analyze news programs, documentaries, etc., as they are broadcast, and present you with only those stories or clips that match your interest.

The purpose of the presented project has been to find out how, if possible, to create such a system: Which are the associated techniques? What are the problems and how can they be overcome? Who are working on this and what have they achieved? Two specific problems, representing segmentation and content extraction, have been given extra attention: shot boundary detection and video text extraction and recognition. For the latter problem, a prototype system has been developed.

The outline of this report is as follows: Section 2 gives the general background of the project, including a presentation of the funder, Observer Media Intelligence, and a short introduction to general and multimedia-specific information filtering. The section concludes with a survey of related research projects and commercial applications around the world. In section 3, a theoretical design of a content-based television filter is presented, together with a survey of techniques associated with the parts of such a filter, as well as other findings concerning the possibilities and restrictions implied by the local conditions in Scandinavia. Section 4 contains a more detailed survey of shot boundary detection techniques. Section 5 presents a prototype system for the detection, extraction and recognition of text embedded in the video image, especially subtitles. In section 6, the prototype's performance is evaluated. Section 7 presents conclusions and suggestions for further work, concerning both the prototype and the overall project.

Sections 1-7 reflect the project as it was carried out in 1997-98. A new section 8 gives a brief overview of development within the subject area since then.

2 Background

2.1 Observer Media Intelligence

Observer Media Intelligence is a multinational division of *Sifo Group*, with corporate headquarters in Stockholm, Sweden. It consists of companies in Sweden, Norway, Denmark, Finland and Germany, where *Observer Sweden* is the largest. Observer provides thousands of clients among companies, organizations and government agencies with media intelligence services. Services range from newspaper clippings and radio/televison transcripts, via press summaries to advanced media coverage analyses. Speed, coverage and accuracy are extremely important properties of the services.

Much of the work is performed manually. Employees read newspapers or magazines and cut out articles which are considered to match a client's information interest. Similarly, radio and television programs, such as news shows, documentaries and other non-fiction programs, are taped and staff members listen through them, searching for portions to transcribe for clients. With the emergence of electronic media, and of more and more sophisticated tools for computerized information retrieval, Observer has initiated several projects for modernizing the process. One project, called *TMM* ("Tell Me More") has been going on for a couple of years, and concerns digital handling of textual media [Karperyd98]. The purpose is to create a robust system in which newspaper articles, newswire telegrams, etc, are automatically searched for keyword combinations describing clients' interests, quality controlled, and delivered electronically to clients and to Observer's editorial and analysis departments.

The question Observer asks is if it is possible to extend the TMM system to non-textual media, such as television. Specifically, are there any computer systems for filtering television programs, so that instead of listening through all broadcasts in their entirety, Observer staff can concentrate on quality control, thereby increasing speed, coverage and accuracy? If no such systems exist today, will they emerge in the near future? Is it at all possible to create such a system? Those are the questions, which the presented thesis project are intended to answer.

2.2 Information Filtering

The system Observer is interested in, and which would mimic its currently manual services, can be described as an *information filtering system*. It has been suggested that such a system can be characterized by the following features:

- The system is designed for *unstructured* or semi-structured data, e.g. news articles rather than database records.
- Such systems deal primarily, but not exclusively, with textual information.
- They involve large amounts of data.
- Filtering applications typically involve *streams of incoming data*, broadcast or sent directly point-to-point.
- Filtering is based on *profiles*, descriptions of individual or group information preferences, typically representing long-term interests.
- Filtering often implies *removing* data from the incoming stream, rather than finding data in it.

Information filtering is closely related to the comparatively more researched field of *information retrieval*, or IR. There are some differences – one is that in IR, the information base is fairly static and the information need is dynamic, whereas the relation is reversed in filtering; another is that for filtering, the timeliness of a text is typically much more significant than for IR. But in terms of representing texts and queries, and of comparing these, the similarities are such that information filtering and information retrieval can be considered "two sides of the same coin" [BelkinCroft92].

Central to information retrieval is the concept of *relevance* The purpose of an IR system is to retrieve all those documents that are relevant to a particular user and his particular question, while retrieving as few as possible of the non-relevant. Determining relevancy is thus a key task for the IR system, and one of potentially great complexity, however straightforward this task may appear to a human. A particular IR system's successfulness can be expressed as its *efficiency* – measured in terms of the amount of computer resources used to perform its task, and its *effectiveness* – measured with regards to the relevance of retrieved documents. Effectiveness is very often expressed by the metric pair *recall* and *precision*, where

$$recall = \frac{\#\,retrieved\ relevant\ documents}{\#\,all\ relevant\ documents, retrieved\ and\ not\ retrieved}$$

and

$$precision = \frac{\#\,retrieved\ relevant\ documents}{\#\,all\ retrieved\ documents, relevant\ and\ not\ relevant}$$

Recall and precision are inversely related, i.e. measures to raise one will typically lower the other. [vanRijsbergen79] The filter Observer (or any other user with similar needs) requires is thus one which is efficient enough to allow *real-time* or near-real-time filtering, for a "reasonable" investment and running cost, and effective enough to yield "acceptable" recall and precision.

It should be emphasized that the distinction between relevant and irrelevant information is not trivial. For example, the syntactic structure of a statement may in itself be irrelevant information, but it may give clues as to the source (speaker, author) of the statement, whose identity may be relevant information. Relevance varies at different levels or stages of the filtering process. If information relevant to the user typically answers the question, "who said what about this, when, where and in which context?", the problem for the filter is to identify documents where the condition "about this" is fulfilled. Thus any type of information which can be used to evaluate "about this", is relevant to the filter.

2.3 Multimedia Information Filtering

Although the purpose of filtering may be the same, the methods vary with the type of *media* in which the information is contained. In *text* documents, such as newspaper articles, almost all the relevant information lies in the semantic and lexical meanings of the words. Thus filtering generally involves analyzing the words, comparing them to certain key words describing the information interest, etc. At the lexical level, this can be computer-automated fairly straightforwardly as a process of, ultimately, comparing character strings. There exist numerous search engines which, with varying complexity, index and search text documents. A well-known example is Digital Equipment's search engine for the World Wide Web, AltaVista.

Multimedia documents combine different media types, such as text, images, audio and video. A television program can be described as a multimedia document. One of the properties of sound sequences and still or moving images, is the vast amount of information they contain. Consider for example a photograph. The expression "a picture says more than a thousand words" is a gross understatement. A complete textual description of every object in the image, their exact spatial relationships, every visual feature of the objects, every detail of the background, etc, would be inconceivably long. Of course, in any given situation, most of the information would be irrelevant.

The first problem of multimedia information filtering lies thus in *extracting* those features in the multimedia document, which carry information relevant to the filtering process. For a content-based television news filter, such information typically concerns the subject of a news story, people, places, organizations mentioned, etc.

The second problem concerns the *representation* of the extracted information. To allow for automation of the filtering process, the information must be represented in a format suitable for computer handling. It would be advantageous if this format was the same for all relevant information, regardless of in which media it is contained, since this would reduce the number of components needed in the filtering system.

Text is one, but not the only, format in which the extracted information can be represented. One advantage of using text is that, as mentioned above, there are many easily available software applications, which index, categorize and search text documents. Therefore, in designing and implementing a television news filter, one does not have to spend resources on creating special applications for the actual filtering, ie comparing the extracted features with the user's interest profile, but rather, one can focus on the information extraction. Another advantage is that in a typical multimedia document, such as a television news program, at least some of the information is already in text form, and hence does not need to be rerepresented. This advantage is perhaps more evident if one considers an extension to a "complete" content-based news filter, combining input from television as well as other media types, such as newswire telegrams, newspaper articles, etc.

The obvious drawback of representing multimedia documents with text, is the large amount of information discarded. As it is impractical, perhaps impossible, to represent all the visual information in a picture with text, we need prior knowledge of what information is relevant. Otherwise, there are better ways to compactly represent visual information, used for example in IBM's QBIC system [Flickner&al95, QBIC97] and MIT's Photobook system [Pentland&al95]. Another problem is that transforming image, video and audio data to text, is a process of interpretation, so not only may relevant information be discarded; it may also be incorrectly interpreted (assuming that there is one correct interpretation, which is not necessarily the case).

However, for a content-based television news filter, we do have some prior knowledge of what information is relevant, namely information on such high-level semantic entities as people, places and organizations. Extracting semantic features always involves interpretation, regardless of representation, so the advantages of using text seem to outweigh the inconveniences.

After deciding to extract certain features from a multimedia document and represent them with text, the question remains how to actually do it. One way is of course to manually annotate eg. video documents by writing down key words describing the content. Although researchers have developed computer tools for facilitating the process, for instance the Video Annotator of the Norwegian Institute of Technology's VideoSTAR project [Hjelsvold&al95], manual annotation is still too time consuming to be useful in a content-based television news filter. Therefore, tools which automatically extract content-bearing information are needed.

2.4 Automatic Content Extraction and Segmentation

The main sources of content information in a television program are the *spoken dialogue*, *text* appearing in the image, e.g. captions and subtitles, and, if present, *teletext subtitles* or *closed captions*. The latter are transmitted in text format together with the audio and video signal. Extracting them involves *signal processing*, to separate the text signal from the video, but the process is standardized and in practice performed in low-cost hardware. *Text processing* techniques can then be employed to filter out formatting codes and generally to prepare the text data for automatic searching.

Text embedded in the video image requires substantially more processing. *Image analysis* is used to localize and to extract text characters, which are recognized as a process of *pattern recognition*, associated with *artificial intelligence*. The same techniques can be used to extract and recognize human faces and other objects, such as logotypes, as well. *Natural language processing* and *understanding* (NLP/NLU) techniques can be used to further enhance the output of a character recognizer. The result of these operations would be a computer-searchable text file containing the text found in the video and descriptions of recognized objects.

To make the spoken dialogue searchable, it must be transcribed, using *speech recognition* technology. SR, in itself, encompasses audio signal processing, pattern recognition and NLP/NLU. Speaker-independent, open vocabulary, continuous speech recognition is still an unsolved problem. However, the problem is currently given considerable research attention, and there are systems that produce transcripts, which are, although incomplete and erroneous, sufficiently correct to allow for text retrieval of dialogue content.

So far, this discussion on content-bearing features has focused on the problem of deciding whether a multimedia document is relevant to a user's information need. However, if the multimedia document is a TV news broadcast, this is generally not enough. A filter which outputs an entire 45 minutes long news broadcast because a ten second news telegram matched the user's search profile, is clearly of little use. An equally important problem lies thus in identifying those "sub-documents" which correspond to single news stories, so that only the ones of interest may be returned. This problem is generally referred to as (story-level) *segmentation*.

Most approaches to segmentation involve using different forms of image analysis to identify *shots* – uninterrupted sequences of frames captured by a single camera. These shots are then used as the basic building blocks of which story segments are formed. Image analysis can also be used to identify specific objects, eg. text captions, or to recognize certain generic types of shots, which may give clues as to the program's story-level structure. A common assumption is for instance that news stories begin and end with studio shots [Ariki&al96, Furht&al95 ch 14].

Another approach to low level segmentation is to use audio signal processing to distinguish between different types of audio content, eg. silence, music and speech, and to detect speaker changes. High level segmentation often involves using natural language processing to detect probable topic changes in teletext or SR-generated transcripts.

Clearly, content-based manipulation of television programs is a multidisciplinary problem, finding solutions in techniques associated with image analysis, signal processing, speech recognition, NLP, information retrieval, and so on. An excellent overview of these techniques and their use in an extensive digital video project is given in [Christel&al96].

2.5 Related Applications and Projects

During the last three to four years, the problems of automatic content-based filtering of television news, and related issues, have been addressed by numerous research groups around the world. In this sub-section, some applications and research projects are described briefly. The descriptions here are mainly at a procedural level or from a user's point of view. Some of the algorithms and techniques used are described in more detail in sections 3-5.

The complexity of the applications varies substantially, ranging from "simple" systems which rely entirely on closed captions (or teletext subtitles) to complex systems using a combination of image analysis, speech recognition, natural language processing, and other techniques, to extract information from several different sources within the television program. Most of the applications are aimed at content-based retrieval, rather than filtering, but as stated before, this distinction is not fundamental. Some other projects focus on generating abstracts or some sort of hierarchic/logic representations of the program content, to facilitate quick browsing. In general, the projects are at an early developmental stage and it is uncertain whether and when they will result in openly available products. However, at least one commercialized application exists today.

2.5.1 Closed Captions / Teletext analysis for content-based filtering or retrieval

AccesTV is a family of commercial off-the-shelf software products for content-based television filtering, developed and marketed by *Televitesse Systems Inc.* in Canada. The accesTV "assistant" runs on a Pentium PC, equipped with a

video tuner/digitizer and a closed captions decoder.[1] It continuously monitors incoming closed-captioned text for keywords specified in the user's search profile. When a match occurs, a video clip of fix, user defined, length is saved, recording a part of the television broadcast. Optionally, the system alerts the user, who can watch the broadcast online. Over 100.000 keywords can be specified, divided into groups representing different subjects or topics (”searches” in Televitesse's terminology), and interrelated with the Boolean operators AND, OR, NOT and NEAR. The search engine allows misspellings in the closed-captioned text by tolerating a variable character error rate. It also has a ”sounds like” feature which compares the phonetic similarity between keywords and words in the transcript. The accesTV assistant can be used as a stand-alone application or in conjunction with one or more accesTV ”servers”, which retransmit television broadcasts to multiple assistants over a LAN. Each server and stand-alone assistant can monitor one television channel at a time. The filtering system is entirely dependent on concurrently transmitted text transcripts, specifically in the North American closed caption format. Current list prices are USD 699 for the accesTV assistant and USD 6.995 for the accesTV server. [Televitesse97]

Although accesTV seems to be the only currently available COTS-system devoted to filtering television news, similar applications have been developed as research projects around the world. The ”Multimedia Applications in Telecooperation” (*MAT*) research group at the German National Research Center for Information Technology, has for instance built a prototype system simply called *News on Demand*, aimed at retrieval of TV news stories over the World Wide Web. Their prototype automatically records the fifteen minutes long ”ARD Tagesschau” at 8 PM every day, by using hardware to encode the audio/video to a MPEG1-file and storing it in a video server (a Sun SPARC-station connected to a RAID array). Concurrently, a teletext decoder extracts the teletext transcript in ASCII format. Timestamps are added and the resulting text files are stored on a webserver. The user interface is implemented with frames and plug-ins in a web browser (Netscape Navigator). When searching for names or topics, the user enters keywords and Boolean operators in an HTML form field. A CGI-script at the webserver invokes a full text retrieval of the teletext text files, using the University of Arizona's search engine ”Glimpse” and interface toolset ”GlimpseHTTP” [Glimpse97]. The webserver returns a list of search results to the user, with links to corresponding text documents and video segments. By following a video link, the user instructs an MPEG decoder/player plug-in in the browser to start playing the stored news video, at the time where the search hit occurred. [Tenzler&al96, MATNoD95] MAT's prototype is an interesting example of how an application can be built with easy-to-get components. However, in Germany, only one news program per day is accompanied by a teletext transcript [Jonas&al95], which limits the practical use of the prototype system.

An approach very similar to MAT's has been taken in the *Video on Demand* project at the Northeast Parallel Architectures Center (*NPAC*) of Syracuse University, NY. NPAC's prototype, which also is accessed through a web browser, lets the user search CNN News videos, as well as motion pictures and other video documents, for keywords in the closed caption transcript. [Kurmanowicz96, NPACVoD97] In addition to these examples, there are several projects in which closed captions or teletext are used in conjunction with other sources of content information. Some of them are presented in sub-section 2.5.3. Teletext is discussed further in sub-section 3.4

2.5.2 Analysis of embedded text

In many cases, especially outside the United States, teletext transcripts are not available. Some research groups have therefore instead focused on the extraction and recognition of text *embedded* in the picture. Such text typically includes *captions* with information regarding people, places and topics, and, especially in Scandinavia, *subtitles* carrying translations from a foreign language or sent as a service to the hearing-impaired audience.

As a part of the *MoCA* (”Movie Content Analysis”) project at the University of Mannheim in Germany, Rainer Lienhart and his colleagues have developed a semi-automatic system for the retrieval of video clips, eg. news programs, using the text embedded in the video images. They digitize and encode incoming video as a sequence of

[1] In ”stand-alone” mode.

JPEG images, discarding the audio. Through a series of image processing manipulations, text character regions are segmented from the surrounding picture. These regions are run through an optical character recognition (OCR) module. The resulting text output from each image frame is stored, together with a timestamp, in an index. The index can then be searched through an interactive query, in which the user specifies a combination of search words or substrings. These strings are compared to the index text and where they match (with a certain user defined character error tolerance) the corresponding video frames are displayed in the user interface. At the user's request, the retrieved videos are played in an external video browser. The MoCA group first used a gray-scale approach to character segmentation [LienhartStuber95] and later developed a more accurate, but slower, algorithm for handling color images [Lienhart96]. The source code for their applications and example video clips are available for download at their website [MoCATR97].

In a similar project at the *NTT Human Interface Laboratories* in Yokosuka, Japan, researchers Kurakake, Kuwano and Odaka have implemented a multistep algorithm for indexing video clips by their textual content. First, video frames containing text are identified. Then, text regions are extracted from the frames and text lines are identified. Finally, characters are recognized and stored with a timestamp, indicating the frame from which they were extracted, in a searchable index. The NTT system also includes a visual feature matching algorithm for comparing image regions surrounding the identified text. Frames with text of similar visual appearance are assumed to indicate topic changes and are used to create a conceptual segmentation of the video. It is the researchers' present ambition to speed up the system to enable real-time processing. [Kurakake&al97]

Another Japanese research project, at *Rykoku University*, is more specifically aimed at indexing and classifying TV news articles. As in the NTT system, text frames are identified and text regions extracted. Through morphological analysis, nouns are identified in the OCR output and used for indexation. [MotegiAriki96]

Apart from the ones mentioned, there are other projects in which text frames are identified and/or text images extracted, but no character recognition is performed. One example is a system developed by Hae-Kwang Kim at the Computer Science Research Institute of Toulouse, France (*IRIT*), in which text images are extracted from video frames at fixed intervals, stored with timecodes in a database, and displayed in a user interface for quick browsing and fast access to different parts of the video. [Kim96]

2.5.3 Combining multiple information sources, including speech

The applications described in sub-sections 2.5.1-2 have all in common that they depend on *one* type of descriptive *text*, either embedded in the video image *or* transmitted in a separate stream. As such text is not always available, or does not carry sufficient information by itself, some major research projects integrate the analysis of *multiple* information sources within the video document, especially the *spoken words*.

The most ambitious project is probably *Informedia*, carried out at *Carnegie Mellon University* (CMU) in Pittsburgh, PA, within the framework of the U.S. national Digital Libraries Initiative. While the project is currently at a developmental and evaluational stage, it is an expressed ambition to commercialize developed products. Among the project's industrial partners are DEC, Intel and Microsoft. The Informedia project combines a wide variety of technologies in developing tools for accessing large (thousands of hours) video databases. Techniques include identification of text and faces in images, scene cut detection, camera motion analysis, audio content type analysis, speech recognition (SR) and natural language processing of SR-generated or closed caption transcripts. [Wactlar&al96, Informedia97]

An application within the Informedia project is the *News-on-Demand* system, which automatically indexes and stores television and radio news broadcasts in the Informedia Digital Video Library (IDVL). The video data is digitized and compressed to MPEG-I format using basic COTS hardware. The audio is fed through CMU's continuous speech recognizer "Sphinx-II" [CMUSpeech97], which generates a time-indexed transcript of the spoken words. If closed

captions exist, the output from the Sphinx system is used in conjunction with them, to create a corrected, time-aligned transcript. Otherwise, the SR-output is used as transcript. The audio signal is also analyzed for periods of silence. This information is used, together with structural information in the CC-text if available, to segment the news program into ”paragraphs” – ideally corresponding to separate news stories. Image analysis is then performed to further segment the paragraphs, by identifying scene transitions, and to choose a single representative frame for each scene. Finally, the content-bearing words of the transcript, together with information from the segmenting steps, are indexed in the IDVL catalogue. Shortly after the program's conclusion, users can retrieve portions of interest through interactive spoken or written natural language queries. The system extracts keywords from the query and searches for them in the index catalogue, using the *Pursuit* search engine (which is developed at CMU and used in the *Lycos* WWW search service). Documents in the IDVL are ranked for relevance to the query and the best results are listed in the News-on-Demand user interface. If the user chooses to examine a search result, a set of representative frames from the video paragraph is shown, together with the corresponding keywords. The video can be playbacked from any of the frames, together with the scrolling transcript. Except for the speech recognition, which is done on a separate platform, the system runs on a Pentium PC. The system's, especially the speech recognizer's, performance is far from perfect. In retrieval tests conducted with actually transmitted news broadcasts, the Sphinx system's output had a ”word error rate” of approximately 60%, yielding values around 50%[2] for recall and precision, compared to retrieval using a perfect transcript. Improving SR performance is one issue for further research in the Informedia News-on-Demand project. Another is adding OCR capabilities for reading captions and other text in the image. [Hauptmann&al95, HauptmannWitbrock97]

In a major European project, researchers at *Cambridge University* and Olivetti and Oracle Research Laboratory (*ORL*) in the U.K. have developed a prototype system for automatic content-based retrieval of broadcast news. In a process similar to MAT's (sub-section 2.5.1), their system automatically records and digitizes the daily 30-minute BBC1 evening news broadcast. During recording, the video stream is analyzed for image movement. Times when movement is high, indicating possible scene changes, are recorded for later use. Concurrently, a teletext decoder extracts subtitles, which are converted to a suitable text format and stored, together with timestamps, on disk. When the user enters a search query, keywords are matched to the archived teletext files, using standard techniques for text retrieval. A list of the best matching news broadcasts is returned for further exploration by the user. To allow retrieval of only the relevant portions of a certain news program, the teletext transcript is divided into overlapping segments of variable length. A relevance score for each segment is computed, and displayed in a video browser with a color code, in a timeline corresponding to the entire video clip. The user can use this timeline to choose portions of the video for playback. Subtitles are concurrently displayed as a scrolling transcript. [Brown&al95]

At present, the news retrieval prototype relies solely on teletext subtitles for content information. However, it is the developers' intention to integrate a speech recognizer, developed within the *VMR* (”Video Mail Retrieval”) project at Cambridge and ORL, with the application. From a user's point of view, the VMR prototype is very similar to the news retrieval prototype, the major difference being that VMR is concerned with the retrieval of video mail messages, rather than broadcast news. The fundamental underlying difference is however that the output from a speech recognizer is used instead of teletext subtitles for content retrieval. The VMR approach differs from Informedia's, in that the speech recognizer does not output a text transcript, but an intermediary representation of ”phone lattices”. Query words are phonetically transcribed and matched against these lattices. Therefore, no transcripts appear in the user interface. [Young&al97]

The Cambridge news and video mail retrieval systems are built on top of the general, modular, architecture of ORL's *Medusa* project, concerned with sending and processing multimedia over a high-speed ATM network [Brown&al95]. A similar project is *ViewStation* at *TNS*, the Telemedia Networks and Systems Group at MIT. The ViewStation distributed multimedia system consists of *VuNet* – a gigabit-per-second ATM-LAN, connecting Unix workstations and multimedia devices – and *VuSystem* – a software environment for developing multimedia

[2] In a later paper, [HauptmannWactlar97], figures of 85%-95% are mentioned.

applications. In ViewStation, audio and video is passed through a number of modules, performing various operations on the multimedia data. So far, over 50 software modules have been written and integrated in various TNS test applications. Some applications perform real-time (image) processing of live video. Others are designed for content-based retrieval of pre-recorded television programs. One such application is the *News Browser*, which permits the user to search for keywords in the closed caption transcript of TV news programs, such as CNN Headline News. Matching news stories are displayed in a news browser window. Other, technically more advanced, applications are the ”Joke Browser”, which uses NLP and heuristics to segment talk show monologues into separate jokes, and the ”Sports Highlight Browser”, which uses image processing and template matching to extract highlight sequences. [Lindblad&al95, LindbladTennenhouse96] The most interesting aspect of the ViewStation project is perhaps not these test applications, but the architecture upon which they were built.

Researchers at *Virage Inc.* in San Mateo, California have addressed the issue of video representation and retrieval at a rather abstract level. The *Virage Video Engine* (VVE) is a platform independent architecture for processing, specifically indexing, multiple synchronized streams of data, such as the image sequence, audio and possibly closed captions of a television program. In the VVE, a set of *primitives* operate on *media objects*, which are standardized representations of eg. video clips, where the actual data format (mpeg, avi, etc) is hidden from, and irrelevant to, the primitives operating on them. There are four ”default” primitives in the VVE: The ”keyframe primitive” operates on image sequences and extracts a set of key frames, intended to give an ”adequate” representation of the visual content of the video. The keyframes are identified using shot boundary detection and other image analysis techniques. The ”motion primitive” analyzes the image stream and outputs different index values, giving a qualitative and quantitative description of motion in the video. The ”audio primitive” performs audio analysis to index times of speaker changes, silence and transitions between speech and music. indicating possible topic changes. Finally, the ”caption primitive” analyzes closed caption transcripts for speaker or topic changes and for information on specific subjects discussed. More advanced primitives can be added to the VVE, such as audio primitives for word spotting or speech recognizer transcription, giving a finer-grained semantic index when closed captions are not available. [Hampapur&al97]

To which extent the VVE architecture has been implemented in actual applications is not entirely clear from Hampapur et al’s paper. However, Virage Inc. does market two seemingly related products: the *Video Cataloger*, which can ”watch, listen, read, and automatically extract information from the video”, in a ”real-time process result[ing] in a video asset fully indexed by its textual, visual, and audio components”, and the *Video Browser*, with which users can search the indexed videos. Search results are presented as a set of keyframes in a user interface. The keyframes act as links to corresponding locations in the video files, which can be streamed from a video server and displayed on screen. [VVCB97] The Virage Video Cataloger and Browser are not designed specifically for content-based retrieval of broadcast news, but could probably be used satisfactorily for that purpose, provided either that the news broadcasts are accompanied by closed captions, or that a word spotter or speech recognizer is added to the system.

A system with similar features, but specifically designed for content-based retrieval of television news, is being developed at *Mitre* Corp. in Bedford, Massachusetts. The *Broadcast News Navigator* (BNN) is a web-based navigation tool for searching and viewing TV news. Broadcasts, live or pre-recorded on analog video tape, are divided into video, audio and closed caption streams, which are analyzed in parallel. The video is digitized and stored in a video database. At the same time, it is analyzed for scene changes and keyframes. The audio stream is analyzed for speaker changes and the information thus extracted is used together with the scene change information to automatically detect story segments. For these segments, summaries are generated with information from the closed captions analysis. The CC-stream is especially analyzed for *named entities*, ie persons, organizations, locations, dates or time values. These entities are used as *tags* associated with each news story. The user can either search for specific tags or instruct the system to perform a free-text keyword search of the whole transcripts. It is also possible to search for certain topics, such as ”Chinese Leaders” or ”Swiss Banks”. Another feature of the BNN is to display tag

frequencies for a collection of news stories, giving an indication of what the media's attention is focused on during a certain time period. [MerlinoMorey97, BNN97]

Yet another system for TV news retrieval is being developed at Professor Yasuo Ariki's laboratory at *Ryukoku University* in Seta, Japan. The system uses character recognition and speech word spotting to extract keywords from image-embedded captions and the spoken dialogue. These keywords are used to classify video segments into different categories and can also be used as an index for content-based retrieval. TV news programs are segmented into news articles (corresponding to separate stories), using image analysis to identify scene breaks and studio scenes, and a heuristic model of articles being separated by studio scenes. An additional feature of the system is an "interactive query function", allowing the user to ask the system about unfamiliar words that are spoken during the broadcast. The system monitors speech from the user and recognizes words that have recently appeared in the broadcast speech. It then retrieves information on the unfamiliar word from an electronic dictionary and presents it through a speech synthesizer. [ArikiSugiyama96, ArikiSugiyama97]

In addition to the described projects and applications, aimed at content-based filtering or retrieval, there are a number of projects dealing with *abstracting* TV programs, or presenting them in a structured form, facilitating fast and easy *browsing*. Within both the Informedia project and the MoCA project, mentioned above, systems have been developed which automatically create "skims" or abstracts – video documents of typically a tenth of the original video's length, intended to give a short but adequate summary of the key contents [SmithKanade97, Pfeiffer&al96a].

3 System Design

The applications described in the previous section give interesting examples of how a content-based television filter can be constructed. However, none of them can be used to completely solve Observer's problem, as suggested in sub-section 2.1. Primarily, of course, because almost all of them are research prototypes, not openly available at present, but also because they are not designed for: a) television *filtering*, b) the local conditions in *Sweden* (or Scandinavia). In this section I therefore discuss how a system specifically aimed at automatic content-based filtering of Swedish television news programs, documentaries and debate programs, could be designed.

3.1 General

An ideal system should keep an easily modified record of the user's search interests, divided into several different topics or profiles. It should monitor all relevant television channels for stories matching these profiles. It should be accurate, with high levels of, most importantly, recall, but also precision. It should be completely automatic, from the input of raw antenna feed to the output of structured pointers to logically coherent portions of digitally stored video documents. Furthermore, it should be fast, giving access to relevant stories within a reasonably short time, preferably a couple of minutes, after the broadcast. And in addition to all this, it should be designed as cost- and resource-efficiently as possible.

These requirements imply a multi-modal, modular design. *Multi-modal* means that information in *multiple* different forms within the television broadcast, such as teletext, audio and video, is *combined* in the filtering process. The reasons for this are two-fold. First, without information about what is *said* in the program, too much of the high level semantics of news stories would be missed, making it virtually impossible to determine whether the "about this" condition of the generic expression of the user's information need, "who said what about this, when, where and in which context?", is fulfilled. As table 3.1 shows, only a small percentage of Swedish news broadcasts are subtitled when Swedish is spoken. Furthermore, with one exception, these broadcasts are shorter news summaries, covering only a subset of the day's news stories. Although the figures are somewhat "better" for documentaries, a significant percentage of the broadcasts are not subtitled. The situation is similar in the neighboring Scandinavian countries, as can be seen in tables 3.2 and 3.3, and in Germany, where approximately 10% of the news broadcasts are subtitled[3]. The conclusion must be that the ideal filter needs a speech recognizer.

But, secondly, today's state-of-the art speech recognizers still output erroneous transcripts and it is reasonable to assume that to some extent, future implementations will too. It would thus be wasteful not to use text transcripts if such are available. Speech in languages other than Swedish is almost exclusively translated in subtitles. Sometimes teletext subtitles and image-embedded text carry other types of information, not given in the dialogue. Therefore, speech recognition would simply not be enough by itself.

Modular means that the system is built with several separate software or hardware modules, which in different ways process the data. Although practically all the systems presented in sub-section 2.5 can be described as modular, perhaps the most illustrative example is MIT's ViewStation system, discussed in 2.5.3. The obvious advantage of a modular design is that different functionalities of the system can be designed and implemented independently. This facilitates parallel development and makes it easier to add or improve functionalities over time. Furthermore, not all modules would be of use for all television programs; a teletext-module would of course be redundant for most Swedish news shows. Maybe some programs would be most fruitfully analyzed with a customized version of a certain module. With a large set of independent modules, designed to interact with a common protocol over a common network, and easily accessed and controlled within some common software framework, it should be fairly

[3] Of 106 news broadcasts on 24 German TV channels, on Wednesday, September 17th 1997, 11 were announced to be subtitled [OnlineTV97].

Table 3.1: Swedish Television

Type of broadcast	Number of broadcasts per week	Total length of broadcasts per week (min.)	Percentage of broadcasts			
			Not subtitled when Swedish is spoken	Teletext-subtitled	Subtitled in image	Not subtitled but later rerun with subtitles
National and International News	186	3075	84	9	7	-
Regional News	704	7590	100	-	-	-
Sports News	28	290	96	4	-	-
Community "Bulletin Board"	5	25	-	-	100	-
In-depth Coverage and Debate	3	160	67	33	-	-
Special Interest Shows (Science, Health, etc.)	21	785	67	28	-	5
Documentaries	14	560	71	29	-	-
Other non-fiction (subset only)	18	730	89	11	-	-

*Table 3.1. Percentage of Swedish TV broadcasts which are subtitled. It should be noted that speech in languages other than Swedish is generally subtitled (in the image), regardless of whether and how the rest of the broadcast is subtitled. Data collected from the five major TV channels, SVT1, SVT2, TV3, TV4 and Kanal5, between September 1st and 7th, 1997.(*Dagens Nyheter*, Stockholm, August 28th & September 4th, 1997, [SVT97])*

Table 3.2: Norwegian Television

Type of broadcast	Number of broadcasts per week	Total length of broadcasts per week (min.)	Percentage of broadcasts			
			Not subtitled when Norwegian is spoken	Teletext-subtitled	Subtitled in image	Not subtitled but later rerun with subtitles
National and International News	76	2325	91	9	-	-
Regional News	225	6495	100	-	-	-
Sports News	11	130	100	-	-	-
In-depth, Debate & Special Interest	16	435	50	50	-	-
Documentaries & other non-fiction	25	1040	76	24	-	-

*Table 3.2. Percentage of Norwegian TV broadcasts which are subtitled. Speech in languages other than Norwegian is generally subtitled (in the image), regardless of whether and how the rest of the broadcast is subtitled. Data collected from the five major TV channels, NRK1, NRK2, TV2, TV3 and TV Norge, between September 5th and 11th, 1997. (*Dagbladet, *Oslo, September 4th 1997, [NRK97])*

Table 3.3: Danish (State) Television

Type of broadcast	Number of broadcasts per week	Total length of broadcasts per week (min.)	Percentage of broadcasts			
			Not subtitled when Danish is spoken	Teletext-subtitled	Subtitled in image	Not subtitled but later rerun with subtitles
National and International News	32	525	59	41	-	-
Sports News	6	65	100	-	-	-
In-depth, Debate & Special Interest	20	660	85	10	-	5
Documentaries & other non-fiction	28	915	79	21	-	-

Table 3.3. Percentage of Danish (state) TV broadcasts which are subtitled. Speech in languages other than Danish is generally subtitled (in the image), regardless of whether and how the rest of the broadcast is subtitled. Data collected from the two channels of the Danish Broadcasting Corporation, DR1 and DR2, between September 1st and 7th, 1997. [DR97]

uncomplicated to use *exactly* the modules needed in a given situation, with hopefully an improved performance and a minimized use of costly resources (eg. external software licenses, hardware) as a result.

In summary, a system for content-based filtering of TV programs, tailored for Observer's needs, should contain modules for extracting information from the video stream, as well as the audio stream and possible teletext subtitles. It should also contain some module which creates intelligently segmented searchable documents from the extracted information, and a module for the actual information filtering. A schematic design layout is shown in figure 3.1.

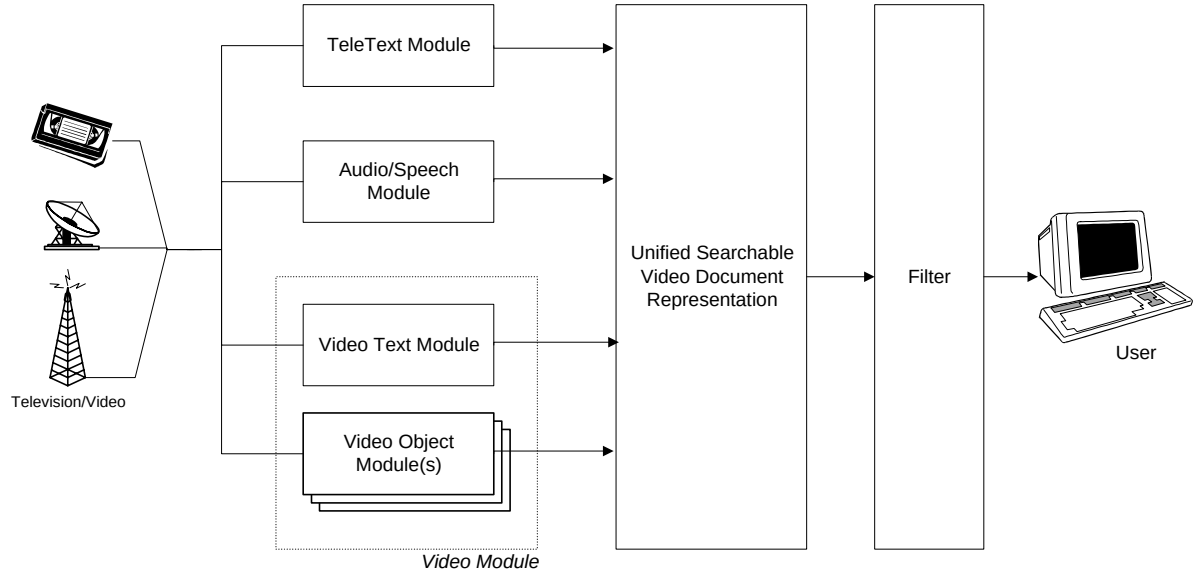

Fig. 3.1. Basic design of a content-based video filter. In an actual implementation, some of the modules would probably be divided into several sub-modules, and other modules would be added to the system.

The remainder of this section discusses, rather briefly, some design issues. Emphasis is on image processing. Some fundamental properties of digital audio and video are presented, as well as relevant research results from around the world, in terms of design choices and algorithms for existing implementations of the functionalities of the different modules. In the concluding sub-section, the overall design questions of this sub-section are revisited, and a more detailed design of a proposed system is presented. Two functionalities or modules are discussed in greater detail in the following sections: Section 4 contains a survey of shot boundary detection techniques suggested in literature. Sections 5-7 describe my own implementation of a prototype module for the extraction and recognition of image-embedded text.

3.2 Digitizing and Compression

The first step to analyzing and manipulating TV programs is to capture and digitize the audio and video streams, and to represent them in a format suitable for storing and transmission.

3.2.1 Analog and digital video

Traditional analog video is perceived as a two-dimensional representation, semi-discrete in time and space, of a continuous 3-D world. Technically, though, it consists of one or more parallel one-dimensional continuous signal(s), which the TV camera creates through *raster scanning* of the image in view. The camera sensor scans the image from left to right along (almost) horizontal *lines*, registering at all times the brightness, and possibly color, of the point currently in view. Each line slants slightly downwards, and when the sensor reaches the end of a line, it jumps back

to its horizontal starting point, where it continues to scan a new line, slightly below the previous. When the scanning reaches the bottom of the image, the sensor jumps back to the top and starts scanning a new *frame*. The time it takes for the scanning point to move back from the bottom right to the top left corner, is called the *vertical blanking interval*, VBI, during which the sensor is switched off. In the PAL television standard, used in most of Western Europe, 25 frames are transmitted every second, with 625 lines per frame, of which c. 40 lines are in the VBI. [Luther92]

One standardization issue is which base vectors to use in *color space*. The three base colors for transmitted (as opposed to reflected) light are red, green and blue. With a combination of them, any color can be formed. Thus, by assigning levels of red, green and blue to all points in an image, an arbitrary color image is obtained. This is e.g. done in the tube of a color television set. However, there are other possible color base vectors than RGB. Due to the fact that the human eye is more sensitive to black/white intensity than to color, most TV and video standards use a color scheme with one *luminance* component, representing the gray scale intensity, and two *chrominance* components, representing the color. The latter are allocated lower bandwidth in the combined video signal, with corresponding lower resolution.

One luminance/chrominance standard, used in PAL, is *YUV*. It has an approximate relation to RGB where

$$Y = 0.3R + 0.6G + 0.1B\,; \quad U = B - Y\,; \quad V = R - Y$$

which in the case of pure grayscale images, i.e. $R = G = B$, means that $Y = R = G = B$ and $U = V = 0$. [Fuhrt&al95]

The two competing TV standards to PAL are NTSC, used in North America and Japan, and SECAM, used in France and Eastern Europe. The SECAM standard uses the same frame rate and size as PAL, but a different encoding technique for the chrominance components, whereas NTSC is more similar to PAL in this respect, but uses 30 frames per second, with 525 lines per frame. All standards use a method called *interlacing* to reduce flickering and enhance the perception of smooth motion, without increasing signal bandwidth. With interlacing, each frame is recorded, and transmitted, in two scanner runs – one for all the "odd" lines in the frame and one for all the "even". This means that the image is updated 50 times per second (60 for NTSC), with one half of a frame at a time. [Luther92]

So much for analog video. To be able to store and manipulate the video on computer equipment, we need to *digitize* it, i.e. make the representation truly discrete. This is done through digital sampling and quantizing and possibly interpolation and other manipulation. Like analog video, digital video has a frame rate, but each line is divided into a fix number of *pixels*, each of which carries a discrete value representing the intensity and color at that point. Like analog video, digital video can have different color encodings and different bit depth (corresponding to bandwidth) for the color component values. A typical digitized PAL video has 25 frames per second, where each frame is 576 x 720 pixels. The intensity and color of each pixel is typically represented by an 8-bit value (0-255) for each of the three color components, e.g. RGB.

Digital video formats allow interlacing, but digital video displays, such as computer displays, are typically *non-interlaced* (a.k.a. *progressive scan*). And for the type of image analysis discussed in this report, still frames that can be treated individually are required. Video digitization may therefore involve a process called *deinterlacing*. Without deinterlacing, objects in motion, captured with interlacing, appear with jagged edges. This is because the object will have moved between the two scanner runs, and is thus placed differently in the two subsets of lines. Deinterlacing uses motion estimation to guess where objects move to on reconstructed scan lines, but this process is computing-intensive and prone to errors. [LiebholdHoffert91]

For practical reasons, digital video is typically *compressed* (cf. sections 3.2.3-5) before it is stored or transmitted. The choice of compression method also determines the file format of the video data file. In practice, video is digitized using a *video capture board* installed in a computer. Such a board may include a TV tuner, which separates the video

signal from the carrier wave on which it is broadcast. It often includes hardware for compressing the video signal according to some standard, and it may also be able to extract teletext (cf. section 3.3) from the TV signal and store it separately. There are several suppliers of such hardware, where the U.S. company Hauppauge claims to be market leaders in TV capture and digitization boards [Hauppauge97].

3.2.2 Digital audio

Audio, like video, must be digitized prior to computer storage and analysis. In a device known as an analog-to-digital converter (ADC), which is a standard component in a PC sound card, the continuous analog audio signal is made discrete in time through sampling and in amplitude through quantization. According to the Sampling Theorem, the highest signal frequency that can be reconstructed properly is half the sampling frequency, also known as the Nyquist frequency. Common sampling frequencies for digital audio are 44.1 kHz and 48 kHz, which allow for correct representation of the full 20-kHz range of (normal) human hearing. The amplitude range of human hearing is assumed to be near 120 dB, which is not quite covered by the common quantization level of 16 bits per sample, as this translates to 96 dB ($\approx 20 \log_{10} 2^{16}$). Nonetheless, the Audio-CD standard prescribes 16-bit sampling at 44.1 kHz, with *PCM* encoding. PCM, or pulse-code modulation, is the generic name for the digital representation of an analog signal, where each sample is translated into a digital code. [Strawn94]

3.2.3 Compression[4]

Digitizing audio and video means creating vast amounts of data. An hour of CD-quality audio is 44.1 kHz × 16 bits × 2 (stereo) channels × 3600 seconds = 620 MB. Similarly, an hour of video, captured with the settings of section 3.2.1, is 104 GB. To allow efficient handling, both in terms of volume and speed, data is typically *compressed* prior to storage or transmission and *decompressed* for playback or analysis. And as will be shown later in this report, some analysis can actually be performed better on compressed data, than on uncompressed.

Data compression algorithms make use of *redundancy* in the uncompressed data. Redundancy is roughly stated the difference between the *data* and the *information* contained in the data. Information theory was pioneered by Claude Shannon, who in a famous paper ([Shannon48]) introduced the concept of *information entropy* as a measurement of the amount of information in a discrete signal, based on the probabilities of the different possible signal values. Through Shannon's first theorem, this entropy defines the lowest average number of bits[5] needed to accurately code and transmit the different signal values, thus indicating the potential for compression.

Such compression, which allows perfect reconstruction of the original data, is called *information preserving* or *lossless* compression. Lossless compression, which is one of the two major groups of compression methods, utilizes various forms of *statistical* redundancy in the data. One basic form of lossless compression, *variable length coding*, is based on the assumption that certain data values are statistically more probable than others, and therefore coded with fewer bits than less common values. Morse code is a variable length code, where the letter E – the most common letter in the English alphabet – is represented with a single dot, whereas the more uncommon letters Q and X are represented with longer sequences of dots and dashes [ibid]. A binary counterpart to Morse is *Huffman coding*, which is the optimal coding with regards to Shannon's first theorem, when the values are coded one at a time.

Other basic forms of lossless compression are *run-length coding* and *differential* (or *lossless predictive*) *coding*, which utilize the *temporal/spatial* redundancy of typical images, audio and video, i.e. the phenomenon that the difference between two adjacent pixels (or sound samples or video frames) is typically small. With run-length coding, rather than storing e.g. the value '255' 720 times in a row for full line of white pixels, one could store only the values '(255,720)'. With differential coding, one would store the difference between a value and its neighbor, rather than the full value. With

[4] Except where otherwise stated, the background information for this section was found in [Fuhrt&al95], [GonzalesWoods92] and [Luther92].

[5] The term *bit*, short for "binary digit", was in fact also introduced in [Shannon48], although Shannon attributes it to J. W. Tukey.

this difference statistically likely to be zero or low, differential coding in combination with variable code lengths can provide significant savings. In fact, many of the commercially implemented lossless compression algorithms use combinations and modifications of these basic algorithms.

The other major group of compression methods is the *lossy* group. Lossy compression methods can achieve higher compression ratios than lossless, and are therefore often used for multimedia applications. However, for each compression and decompression operation, some information is lost. But the idea is to target the *psychoacoustic* and *psychovisual* redundancy of audio/video, i.e. information which due to limitations in the human ear and eye is not easily perceived.

One type of lossy compression already discussed is the bandwidth suppression (corresponding to *sub-sampling* in the digital case) of the chrominance channels in the major television standards (cf. section 3.2.1). Other types include *(lossy) predictive coding* and *transform coding*. In the latter, data in the spatial or temporal domain is transformed to the frequency domain, where frequency components with small magnitudes are assumed to be less perceivable to humans, and are therefore more coarsely quantized or simply discarded.

The de facto standard transform within image compression is the *Discrete Cosine Transform* (DCT). Like its close relative, the Discrete *Fourier* Transform, it outputs a sort of frequency spectrum. (In the continuous case, if the input sequence is real, the Cosine Transform's output is simply the real part of the Fourier Transform's.) The transfer function of the two-dimensional DCT is

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad , \quad u,v = 0,1,2,\ldots,N-1$$

where

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & , \quad u = 0 \\ \sqrt{\frac{2}{N}} & , \quad u = 1,2,\ldots,N-1 \end{cases}$$

and $f(x,y)$ is the (eg. intensity) value of pixel (x,y) and N is the number of pixels in each direction. In image compression, DCT is typically performed on blocks of 8×8 pixels, yielding 64 different DCT coefficients. The first of these (corresponding to "frequencies" $u = v = 0$) is simply the *mean*[6] of all 64 values and is often called the *DC* coefficient (or component). The remaining 63 are called *AC* coefficients. It should be noted that the transformation in itself does not provide any compression; this is achieved in the subsequent quantization (or discarding) of the transformed coefficients. [GonzalezWoods92]

Fourier-type transforms are used in audio compression as well. Fast Fourier transforms can be used to distinguish more sinusoidal, or "tonal" frequency bands from more noise-like, as the first step of a *subband coding* algorithm. Subband coding is a popular audio compression method, used e.g. in the MPEG standard (cf. section 3.2.5, below). It exploits the psychoacoustic phenomenon of "masking", which means that the human ear is not perceptible to certain sounds, because they are temporarily masked by other sounds. In subband coding, the audio frequency range is divided into N subbands. For each "time frame" of M samples, M/N samples are registered per subband. However, only those subbands are actually encoded, which are least masked in the particular time frame, whereas the rest are dropped. This provides significant data savings, with little or no audible difference. [Strawn94]

3.2.4 JPEG

One of the most commonly used compression standards for grayscale and color images is *JPEG*, named after the Joint Photographic Experts Group, who conceived it. JPEG does not prescribe *one* compression method, nor *one*

[6] Well, actually 8 times the mean, but that doesn't make any principal difference.

combination of methods, but rather a standardized framework for choosing between a number of specific methods. The full JPEG standard supports four modes of operation:

- sequential DCT-based encoding, where each image component is encoded in a single left-to-right, top-to-bottom scan;
- progressive DCT-based encoding, where the image is encoded in multiple scans, with subsequent greater detail;
- lossless predictive coding; and
- hierarchical DCT-based encoding, where the image is encoded at multiple resolutions.

To be JPEG-compliant, an encoder or decoder does not need to handle all these modes, but it must support the *baseline* sequential encoder, which is illustrated in fig. 3.2. The baseline encoder divides the image into 8×8 pixel blocks from top left to bottom right. For color images, this is done separately for each color component. Each block is transformed using the Discrete Cosine Transform. The result is an 8×8 matrix of DCT coefficients in the range [-1024, 1023]. These are quantized by the equation

$$F_q(u,v) = Round\left[\frac{F(u,v)}{Q(u,v)}\right]$$

where F is the original coefficient, and Q is a corresponding value in a quantizing table, which is not predefined but supplied in the JPEG file. After quantization, many of the 63 AC coefficients are typically zero or low, and they are ordered in the zig-zag pattern shown in fig. 3.3, to improve the effectiveness of the final encoding step. This is an entropic (or statistical) encoding, using a combination of run-length coding and Huffman coding. The DC coefficient is treated slightly differently; since it is usually strongly correlated to the DC coefficients of adjacent blocks, it is encoded with predictive coding, based on the previous block, before it is also Huffman coded.

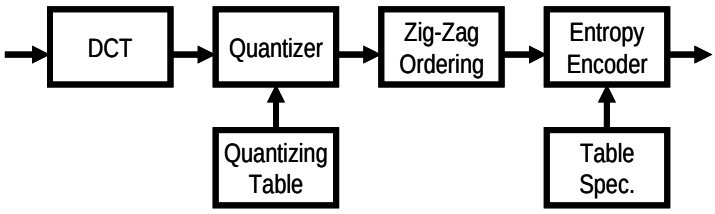

Fig 3.2.The encoding process of the baseline JPEG encoder

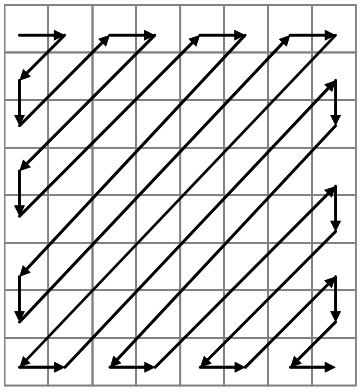

Fig 3.3. Zig-Zag ordering of the DCT coefficients from (0,0) to (7,7)

The decoding process is essentially the inverse of the encoding. The standard allows for different parameter settings, e.g. through the quantizing table, with which various degrees of image compression and corresponding degradation can be achieved. For a typical photographic image, with average to good image quality after compression, a typical file size is 1/15 of the original uncompressed file. [Fuhrt&al95, Luther92]

3.2.5 MPEG

A sequence of JPEG images can be used to represent video. This rather common format is usually referred to as ”Motion-JPEG” or ”MJPEG”. However, Motion-JPEG does not consider the great temporal redundancy of video, and it is not an official standard. The corresponding video standard is *MPEG*, as set forth by the Moving Picture

Experts Group. MPEG defines a video compression standard, an audio compression standard, and a standard for combining video and audio in a data stream, through a process known as *interleaving*.

MPEG video compression uses three types of image frames, with different coding algorithms:

- *I* frames ("Intra-images") are encoded without reference to other frames, with a DCT-based technique similar to JPEG. They provide random access points into the video.

- *P* frames ("Predicted" images) are encoded using *motion-compensated forward prediction* based on the previous I or P frame. P frames are significantly more compressed than I frames.

- *B* frames ("Bi-directional" or interpolated images) are prediction coded with reference to the previous I/P frame, or the next I/P frame, *or both*, whichever provides the best compression. B frames are thus the most compressed frames.

Motion-compensated prediction assumes that the current image can be modeled as a set of local translations of the previous image, where different sub-images can move in different directions, and with different amounts of movement. This *motion vector* is calculated for each 16×16 pixel block (known as a "macro-block") in P frames and uni-directionally predicted B frames, whereas interpolated B frames have two motion vectors per block. Motion vectors are encoded as the difference from the motion vector of the next adjacent block and represented with a variable length code. The difference between the actual and the predicted image, known as the *prediction error image*, is DCT-transformed, quantized and encoded with a variant of Huffman coding, much like I frames and JPEG images.

The MPEG standard allows for different sequences of I, P and B frames, but a common sequence is the one shown in fig. 3.4, below.

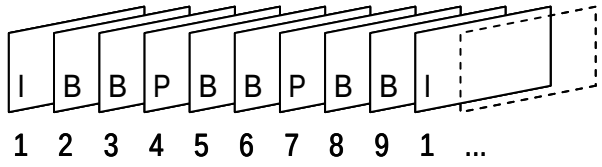

Fig. 3.4. A typical sequence of I, P and B frames in MPEG video.

The MPEG *audio* standard is based on subband coding (cf. section 3.2.3) and allows for three different *layers*, or degrees, of processing. The simplest layer, layer 1, uses 32 subbands and frames of 384 samples, whereas the most complex, known as "MPEG audio layer 3", uses 1152 samples per frame and much more complex filtering and quantization, yielding the highest compression ratio. MPEG audio allows three different sampling frequencies – 32, 44.1or 48 kHz – of mono or stereo signals.

Like JPEG, MPEG is a "standards toolbox" rather than one specific method, and several parameters are open – e.g. frame size, frame rate, frequency of I frames (which affects editability, among other things), bit rate (through e.g. quantization tables), etc. To guarantee a minimum level of interoperability between systems using MPEG, the group defined a "Constrained Parameter Set" of maximum resolution, etc, targeting a bit rate of 1-1.5 Mbit/s, which is equivalent to standard CD data transfer speed. This constrained parameter set was incorporated in the first phase MPEG-1 standard. The second phase, MPEG-2, targets higher resolutions and bit rates between 2 and 80 Mbit/s. MPEG-2 also addresses a number of issues not addressed in MPEG-1, such as interlaced video support and multi-channel audio. The next phase, MPEG-4, is intended to target very low bit rates. As with JPEG, the standard does not in detail define an encoder, whereas it strictly defines the functionality of a decoder. This opens up for several competing implementations of coder/decoders (or *CODECs*), which can be implemented in software or as hardware chips on e.g. TV capture cards. [LeGall91, Fuhrt&al95, Strawn94]

3.2.6 Design implications

As table 3.1 shows, Swedish news broadcasts alone sum up to approximately 150 hours per week. Only the visual data would amount to as much as 15 *Tera*bytes in digital form, with reasonable color resolution and no downsampling or compression. Storage issues alone clearly suggest the need for efficient compression. MPEG is the most efficient video compression standard. Using compressed video data in all stages of the filter furthermore limits the required bandwidth of the communication channels, and, as will be shown in sub-section 3.4 and section 4, it can decrease the number of computations necessary in several image processing steps. Hardware which encodes real-time video to MPEG is available at moderate costs. Therefore, it is proposed that one of first modules in the filter design should be an MPEG encoder and that all video data transferred between and used within the modules, should be MPEG-compressed, except, naturally, in those situations where uncompressed data is unavoidably needed, for instance when video clips are displayed in the user interface.

3.3 Teletext

Having thus digitized and compressed the TV program to monitor, it is time to start analyzing its contents. Of the three general modes of information in fig. 3.1, teletext subtitles, where available, are the simplest to handle. Once captured, they require hardly any further processing, before they can be searched for profile keywords. They are also the least error-prone.

Teletext subtitles are provided as part of the teletext information service, offered by most Scandinavian TV channels. Teletext *pages* are sent as text signals embedded in the VBI part of the television video signal. Decoders within TV sets or on PC capture cards convert the signal to lines of text, which are displayed on top of the TV image. In the case of a capture card, the text can also be saved in a separate text file. [Schneider94]

There are capture cards which capture teletext only, and cards which capture teletext in addition to capturing audio and video. In both cases, the cards are typically bundled with software for programming the card, saving the text in various formats and dynamically transferring it to other applications, etc. There is an abundance of commercial products available, e.g. "OPT-III", which offers a Windows Developer's Toolkit; Philips Semiconductors' "PC Text" software and several different decoder chips; the Australian "Unitext" system; and the previously mentioned Hauppauge products. [Opt97, Philips97, Pelican97, Hauppauge97]

Teletext subtitles are intended to contain the essence of the spoken dialogue, but they are no exact transcripts. For reasons of space and legibility, the subtitles are typically briefer, and sometimes corrected or simplified in language, than the speech. As shown in table 3.1, a fairly limited amount of Swedish TV programs are subtitled. According to Annika Forss, who works with subtitling issues at Swedish National Television, SVT aim to increase the number of subtitled programs, but no dramatic changes are expected for the coming years.

By comparison, the North American equivalent to teletext subtitles, Closed Captions, provide an even better platform for content based retrieval or filtering. First of all, they are much more prevalent, due to U.S. federal law, which stipulates that virtually all new television programs must be captioned. They are also generally more verbatim than teletext subtitles. Like teletext subtitles, Closed Captions are transmitted in the vertical blanking interval of the video signal and they can be captured with COTS TV capture cards. [CapFAQ97]

3.4 Image Analysis

Image and video analysis, in a wide sense, can provide much useful information to a content-based TV filter. This information can be both of a syntactic nature – how the program is structured, and semantic – what is said and what "happens" in the program.

3.4.1 Shot Boundary Detection

Shots – uninterrupted sequences of frames, captured by a single camera – are generally used as the basic building blocks of which coherent video segments are formed. Detecting the boundaries between shots is thus a fundamental function in any system in which story-level segmentation is to be attempted. Shot boundaries appear where shots are joined during video editing. There are two major types of boundaries: abrupt changes or *cuts*, and gradual transitions, eg. *fades*, *wipes* and *dissolves*. The latter are often referred to as edit effects.

Shot boundary detection quite naturally involves comparing video frames over time. In general, frames from different shots are ”more different”, in some sense, than frames from the same shot. Several difference metrics have been suggested in literature. In some proposed methods, the differences in individual pixel values are used, in others, overall features are compared, eg. color histograms. Algorithms have been developed for detecting shot boundaries both in regular, uncompressed, video, and directly in MPEG- or JPEG-compressed video. A survey of shot boundary detection techniques is presented in section 4.

Which detection technique to choose, when designing a content-based television filter, is in part, of course, a question of performance. In an independent test of five different detection algorithms, techniques based on histogram differences generally gave best performance in terms of recall vs. precision values, whereas using DCT coefficients in compressed video gave significantly poorer results, especially in terms of precision [BoreczkyRowe96].

Other design issues are computational cost and memory requirements. The algorithms described in section 4 are likely to vary substantially in time consumption, and some may not be suitable for real-time segmentation. Performing multipass algorithms in (pseudo) real-time would require heavy buffering, in addition to fast computations. One way to significantly speed up the computations, regardless of algorithm, is to perform the operations in dedicated hardware. *Dubner International*, of Westwood, NJ, market a frame capture card for PC’s, called the *Scene Stealer*, which, in addition to digitizing video, detects shot boundaries [Dubner97]. The plug-in by itself is only capable of digitizing monochrome video, but interfaces to common off-the-shelf color video capture cards. It is not clear which algorithm the Scene Stealer employs or how well it performs, but the card is used in the BNN system at Mitre (cf. 2.5.3).

3.4.2 Text extraction and recognition

Identifying text embedded in video images is mainly a means of extracting searchable content information, although the presence of certain types of text can also give some segmentation clues. In designing a TV filter, as discussed in this section, the most important features of a video text subsystem are speed and recall in finding and re-representing embedded text in (computer) text format. As the resulting output from such a subsystem is to be computer handled, rather than read by humans, precision is less important. More specifically, it hardly matters if some non-text regions in a video image are accidentally interpreted as text, yielding some garbled character strings in the output, as long as the actual text strings are retrieved and interpreted correctly.

Contrary to the case of shot boundary detection, not very much work has been done on video text extraction and recognition. In addition to the system prototypes mentioned in sub-sections 2.5.2-3, there has been some research on locating and recognizing text in photographs and other complex still images, the findings of which can be applied on video images as well. But compared to the vast amount of research and development in the field of conventional document OCR (recognition of hand-written or printed text on paper), ”Video OCR” is relatively untrodden ground. To my knowledge, there are no commercial systems available.

The process of Video OCR has three distinctive steps: *detection*, *extraction* and *recognition*. Detection means deciding if a video frame, or a larger block thereof, contains any text, without exactly locating it. This first step, which is omitted in many of the described systems, has in literature been handled as a problem of *local* shot boundary detection. In

the NTT system [Kurakake&al97], video frames are divided into subregions and text frames are identified by comparing intensity histograms for corresponding regions in adjacent frames. In a similar manner, Yeo and Liu apply their own "DC sequence" method for compressed video boundary detection (cf. section 4) and consider a text caption appearance or disappearance detected when a scene change is indicated in a subregion, but not in the remainder, of a frame [YeoLiu96]. Both methods assume that a text object does not stay in the picture over a shot boundary, which is typically not true for e.g. subtitles. The approach is *inter-frame*, in that differences *between* frames are used. In the other described systems, the methods for localizing and extracting text are also, implicitly, used for detection. These methods are typically *intra-frame*, i.e. features *within* single frames are used. A new, intra-frame, method for explicit detection is suggested in sub-section 5.3.

The step which has been given the most attention in research involves localizing and extracting text segments. In [Zhong&al95], two common methods are described: the *connected component* method, by which text character segments are distinguished from the background through color analysis; and the *spatial variance* method, whereby text regions, or rather their *bounding boxes*, are identified through variance analysis and edge detection (cf. fig. 3.6). Variants of these methods are used in all systems described in this report. Zhong et al found that the spatial variance method generally works better than the connect component method, although it sometimes has problems with characters extending below the baseline or above the other characters (e.g. 'p' and 't'). Their solution to this problem is a hybrid approach, wherein they apply the connected component method in a second step, to refine the results of the first spatial variance step.

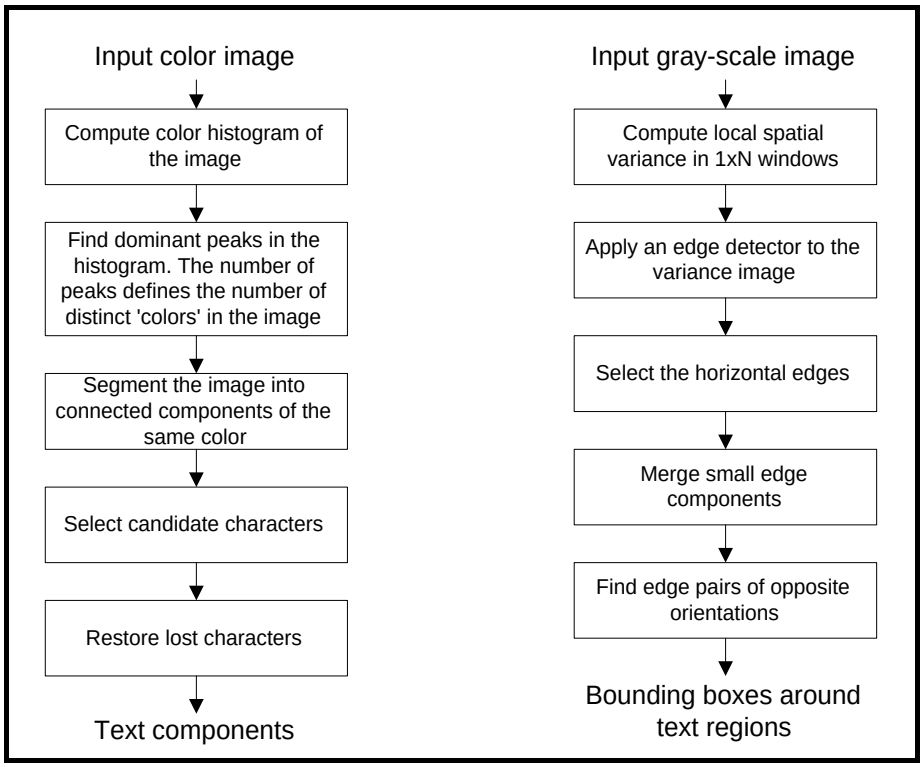

Fig. 3.6. Block diagrams of the connected component method (left) and the spatial variance method (right), as presented in [Zhong&al95]

3.4.3 Face extraction and recognition

Automatic detection and recognition of human *faces* in video images has applications within e.g. security and surveillance and within human-computer interaction. It could also be useful in the context of a TV news filter. For instance, a company interested in a TV monitoring service from Observer, would most likely want to know when their CEO appears on television, even if his or her name for some reason is not mentioned.

As with text in video images, there are several steps in the identification process. The first is to detect video frames containing faces, and to extract only the sub-frames which contain the actual faces. Several methods have been

suggested. Some of these have been developed primarily for identifying faces in collections of images, rather than video, and presuppose that the image contains one and only one face. They are not suitable in this context.

In one intra-frame approach, presented in [Rowley&al95], a neural network is used to detect faces in a sliding 20×20-pixel window. The process is repeated for subsequent downsamplings of the original image, to detect bigger faces. Another intra-frame approach, suggested in [YowCipolla97] aims to detect facial features by searching the edge image for curved shapes, representing eyes, eyebrows, mouth, etc. Where several face shapes are detected with reasonable spatial relationships, these are grouped into face bounding boxes. This approach, which has some similarity with the "spatial variance" method discussed for text extraction above, is more robust to varying sizes and perspectives, than the previous. Other intra-frame approaches include analyzing the image for certain textures or colors, and there are inter-frame approaches based on motion-detection, under the assumption that human faces move more than the background [ibid].

A popular approach to the next step, recognition, is the "eigenfaces" approach, first presented in [TurkPentland91], by which eigenvector analysis is used to extract the most significant features of face images. The distance between two face images in the thus created "face space", is used as a measure of similarity. There are of course other approaches, and an extensive survey can be found in [Chellappa&al95].

3.4.4 Extraction and recognition of other objects

Recognition of text and faces in video images can be considered special cases of *object recognition*. It is possible to extract and recognize other specific objects as well. If a content-based filter is to find TV clips concerning e.g. computers, one way would be to extract those clips where images of computers appear. Object recognition has been a field of research for a couple of decades. A popular presentation of the subject, in the general context of digital image searching, is given in [Forsyth&al97]. One of the authors is currently involved in a research project at Berkeley University, concerning recognition of objects, particularly humans and horses [Berkeley97]. Object recognition based on "eigenfeatures" is implemented in MIT's Photobook system [Pentland&al96]. In IBM's QBIC system, objects are matched to examples based on color, shape and texture [Flickner&al95]. Trial versions of the system can be downloaded from the QBIC homepage [QBIC97]. The system is intended to be included in future versions of IBM's commercial Digital Library product [IBMDL97].

One specific type of object, which would be particularly interesting to recognize in a TV news filter, is the *logotype*, as company logos often are used to illustrate news stories concerning the corresponding companies. To my knowledge, there is no material published on logo recognition in video images. However, there are (research) systems for logo recognition in printed documents, developed e.g. by the Document Processing Group at the Univeristy of Maryland. Their methods for segmentation, feature extraction and classification resemble those for text extraction and recognition, discussed in 3.4.1 [Doermann&al95].

3.5 Audio Analysis

Shot boundary detection, shot classification and semantic content extraction can all be performed through *audio* analysis, instead of, or rather as a complement to, video analysis. It is beyond the scope of this project to present in detail the techniques associated with audio analysis, but there are clearly some similarities to video analysis in the general strategies used.

3.5.1 Content-based segmentation

There are several academic and commercial projects dealing with segmenting audio based on its content, as well as some commercial products. In the MoCA project described in section 2.5, researchers have developed algorithms for distinguishing between silence, speech, music and noise in an audio stream. Silence is detected in the temporal

domain, where *loudness* (the root mean square of the audio signal amplitude) below an adaptive threshold is defined as silence. The other sound types are distinguished in the frequency domain, by looking at the frequency spectrum's *width* (where speech covers a more narrow spectrum than music and noise) and its *"orderliness"* [Pfeiffer&al96b].

A similar, but more formalized, approach has been taken by the developers at Muscle Fish, Inc., a Californian company selling audio analysis software. For each class of sounds, they compute a *feature vector* of mean, variance and autocorrelation values of e.g. loudness, pitch, bandwidth and harmonicity, and classify new sounds by comparing their vectors with the class types'. Classes can be coarsely defined as "speech", "music", etc, or more finely defined as e.g. "close-miked speech" or even speech from different *speakers*. In addition to classifying audio into different categories, the feature vectors can be used to identify *transitions* as sudden changes in the measured features, as a means to segment e.g. news programs [Wold&al96].

In addition to providing syntactic information, speaker recognition can be used for semantic analysis. If a user is interested in e.g. President Clinton, he may want to retrieve video clips in which Mr. Clinton's particular voice was identified. Patel and Sethi have experimented with speaker recognition for video indexing [PatelSethi97a] and a speaker recognition functionality is intended to be added to Mitre's BNN system [Merlino&al97]. However, in the context of this report, speaker recognition is perhaps more useful for *classifying* clips, than for content-based filtering.

The same is true for *word spotting*, by which a system looks for particular words in the spoken dialogue. In a Japanese experiment, 183 different keywords were used to classify TV news stories into ten different categories, e.g. politics, economics and science. Through word spotting, just below two thirds of the stories were correctly classified [ArikiSugiyama97].

3.5.2 Speech recognition

Word spotting, as described above, is a case of *continuous speech recognition* with a *small to medium-sized vocabulary*. Continuous speech recognition differs from *isolated word* recognition, used e.g. for voice-command systems, and the size of the vocabulary determines which type of applications the technology can be used for.

There are many different approaches to speech recognition, but the first step is always to parameterize the speech signal, i.e. to isolate and represent speech segments on a format suitable for analysis. A common representation is *cepstral coefficients*, computed as the log of the transformed frequency spectrum, for time frames of 10-30 ms. Speech segments are then *mapped* to *phonemes*, and eventually *syllables*, *words* and *phrases*. Mapping can be done through e.g. best-fit matching to stored templates or via neural networks. The most common method, however, is the use of statistical *Hidden Markov Models*. An HMM is a finite-state machine, with probabilities associated with the transition from one state to another, but where the states of the machine cannot be directly observed (hence "hidden"). However, it is possible to compute the probability that a certain sequence of observations was generated by a certain sequence of states, and thus the probability that a certain sequence of speech segments was the signal output representing a certain sequence of phonemes (or syllables, words, etc.) [Strawn94].

The complexity of the models increases with the number of different speakers and the size of the vocabulary. To be useful for broadcast news monitoring, a speech recognizer must be *speaker-independent* and handle a *large* (> 10,000 words) vocabulary. It must also handle issues such as spontaneous speech, speech with background music or noise and speech over the telephone. To promote the development of such systems, the U.S. Defense Advanced Research Program Agency (DARPA) invited research groups to join the "1996 DARPA Hub 4 Broadcast News Evaluation". The training and evaluation material for the benchmark comprised 55 hours of speech from U.S. news broadcasts, spoken by 2,000 different speakers, and 140 million words of text material for language modeling – building up a vocabulary of c. 65,000 words. Nine SR systems participated in the evaluation, including the Sphinx system used as part of the Informedia project (cf. 2.5.3), a system developed at the Californian company SRI, and two systems developed at Cambridge University – the ABBOT system using a hybrid approach with neural networks, and the

HTK system, based on a more "traditional" HMM approach. The latter was one of the most successful systems in the test, achieving an overall *word error rate* of 27.5%, meaning that three out of four words were correctly recognized. [Stern97, Pallett&al97, Cook&al97].

These systems are speaker-independent, but not *language*-independent. In fact, they all depend on NLP language models for the only language they can recognize speech in – American English – and on access to large amounts of American English training data. An SR system for use at Observer in Stockholm must be able to recognize speech in Swedish – a language for which there has been significantly less SR research than for English – and must somehow cope with the fact that the dialogue in a Swedish TV news broadcast typically switches between Swedish, English and often other languages as well. As non-Swedish dialogue is normally accompanied by subtitles, an overall system which could recognize these would only need an SR component that could identify language, though, and "switch off" for non-Swedish. But there are examples of multi-lingual speech recognizers. SRI, mentioned above, and the Swedish company Telia Research, have developed a system which can recognize speech in Swedish and English freely mixed, although with a rather limited vocabulary [Weng&al97]. An example of Swedish-only speaker-independent, continuous speech recognition, is the WAXHOLM project at the Royal Institute of Technology in Stockholm. The system currently handles a medium-sized vocabulary of c. 1000 words [Ström96].

A problem with the large vocabulary approach to broadcast news monitoring is that the vocabulary is still finite. Names or words that were not in the vocabulary when training the system, will not be recognized. To circumvent this problem, attempts have been made at *open-vocabulary* recognition, through *phonetic*, rather than literal, *transcription*. Researchers at the Swiss Federal Institute of Technology have built a retrieval prototype, which creates a phonetic representation of the spoken audio in radio news broadcasts. Rather than using this to create a text transcript, which can be searched for plain-text keywords, the keywords are also phonetically transcribed, using a phonetic dictionary, and matched with the phoneme recognition output. In addition to addressing the problem of finite vocabularies, this approach also acknowledges the fact that German is a more difficult language than English to recognize, due e.g. to inflection and noun compounding [WechslerSchäuble95]. A similar approach was taken in the Cambridge/ORL VMR project (cf. 2.5.3), in which the HMM output of *phone lattices*, i.e. multiple hypotheses as to which phonemes have been uttered, is stored and matched with phonetic representations of queries. This provides an open-vocabulary and language-independent system which is insensitive to word recognition errors. As can be expected, this has a positive effect on recall, but at the expense of precision, which is less than 50% [Young&al97].

Both these systems were built partly with the "Hidden Markov Model Toolkit" (HTK), mentioned above. Several of the speech recognition systems in the DARPA challenge are openly or commercially available. E.g. the HTK is marketed by the Cambridge company Entropic, and SRI's speech recognition technology is available from Nuance Communications in Menlo Park, California [Entropic97, Nuance97]. The Sphinx platform is openly available for non-commercial use [CMUSpeech97], whereas the ABBOT system is both freely available in a demo version, and commercialized via another Cambridge company, SoftSound [SoftSound98].

3.6 Other Design Issues

So far, we have explored various means of digitizing TV programs and processing their audio and video streams to extract information which can be useful in the context of content-based filtering. And this is essentially the theme of the remaining sections of this report as well. For a full system, equal consideration could and should be given to the "rest" of the process, which goes beyond the scope of the present project. However, we must have some idea of what this "rest" consists of.

The information extracted in the various analysis modules must be *parsed* into a coherent, time-aligned representation, ideally segmented into separate news stories. These segments must be *filtered*, i.e. matched with client profiles, using search engine technology. The filter output, as well as parameter settings for the various modules,

must be presented and handled through *user interfaces*. And beneath everything, we need a *platform* of hardware, operating system, database and network equipment and protocols.

For all of these, there several alternatives to choose between. Some issues to consider when making these choices are:

- user-friendliness
- system transparency and predictability of its behavior
- speed in processing and data transmission
- cost for hardware, operation and maintenance
- robustness and reliability
- scalability.

3.7 Proposed System

Based on the findings of this section, and on ideas from the systems reviewed in 2.5, I present a more detailed, theoretical, design of a content-based television filter. Figure 3.7 shows the component modules and the data flow in the proposed system. The remainder of this sub-section contains short descriptions of the modules. Two remarks should first be made, though.

First, this version contains a speech transcriber. Ongoing research may prove that the "phone lattice" approach, mentioned in section 3.5.2, is a better solution, in which case the SR module of course would not output text strings. Instead it should send phone lattices to a special search engine, where they are matched against phoneme representations of the search words in the profile database. A new module, which, as automatically as possible, extracts phoneme representations of search words, must be added, as well as a second, phoneme-based, search word database. The phoneme search engine should, when matches occur, send time-coded messages to the parser.

Secondly, in the context of Observer's radio and television coverage services, the "user" in this design should not be interpreted as a client, but as an employee of Observer, performing quality control before results are delivered, electronically or otherwise, to clients.

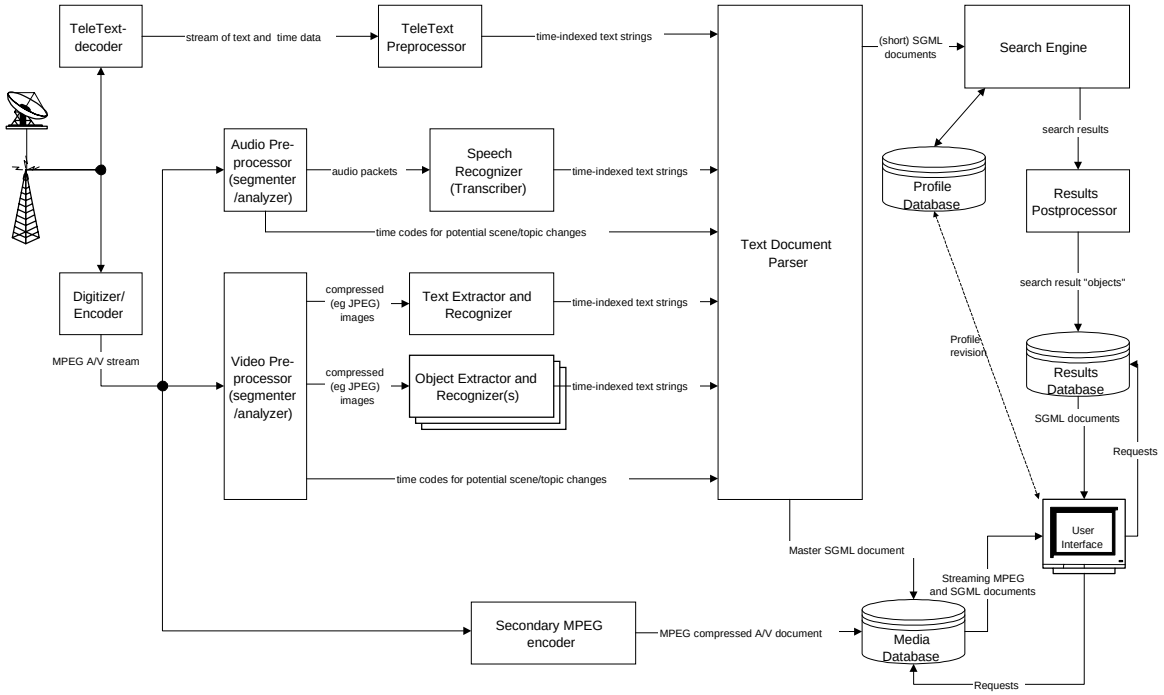

Fig.3.7. A theoretical design of a multi-modal, modular system for atuomatic content-based filtering of television material.

The *TeleText Decoder and Preprocessor* takes television RF-signal as input and outputs strings of ASCII-text, corresponding to subtitles, metatagged with time information.

The *Digitizer/Encoder* consists of hardware TV-tuner and MPEG encoder/accelerator. It takes television RF-signal as input and outputs (”broadcasts”) streaming MPEG A/V through some communications protocol (presumably UDP). The TV-tuner may be integrated with a hardware Teletext-decoder.

The *Audio Preprocessor* analyzes incoming streaming MPEG Audio for content-type (music, speech, silence, noise). It segments and retransmits speech signals in appropriate format to the speech recognizer, and sends time and type information of content-type transitions to the parser.

The *Speech Recognizer* generates a transcript of incoming speech and sends strings of text, meta-tagged with time information, to the parser.

The *Video Preprocessor* analyzes incoming streaming MPEG Video for scene changes and sends time information of detected shot boundaries to the browser. It also identifies frames, or sub-frames, that probably contain text and/or objects of interest (e.g. faces) and sends them as digital images to the appropriate extractor/recognizer.

The *Text Extractor and Recognizer* extracts regions of incoming images containing text and recognizes the extracted characters. It outputs strings of ASCII-text, corresponding to captions and subtitles, metatagged with time information.

The *Object Extractor and Recognizer* extracts regions of incoming images containing objects of the relevant type and compares them to a set of training objects. If a match occurs, the module generates and sends a string with time and object information (e.g. ”00:03:42.15 Face: Göran Persson” or ”00:14:10.50 Logo: Astra”) to the parser.

The *Secondary MPEG encoder* further compresses the incoming MPEG stream and stores it as a file in the Media Database. The idea is that the higher resolution needed for image analysis is not needed for playback and can be exchanged for less storage space and playback bandwidth.

The *Text Document Parser* receives text-strings from various modules and parses them into documents, with aid from Preprocessor analysis. Ideally, the documents thus created should correspond to separate news stories. A very simple parser uses a fix length window to create overlapping documents. A somewhat more intelligent parser uses time data to determine an interval in which to break between documents and locates the exact break at a scene/topic change detected by the preprocessors, within that interval. This would intuitively give better results than the simplest version, but probably not eliminate the need for overlapping documents. An advanced parser uses Natural Language Understanding of the incoming text, as well as the other information, to form coherent stories. The module continuously sends parsed documents in some SGML (e.g. HTML) format to the Search Engine and also stores a master script of the entire program in the Media Database.

The *Search Engine and Results Postprocessor* compares incoming documents to search profiles in the Profile Database. Matches over a certain relevance score threshold (or equivalent, depending on implementation) generate search results documents, containing a program identifier, name of matched search profile, search score and time codes corresponding to the beginning and end of the document. Results documents for the same profile are combined if they are overlapping or adjacent in time. Results are stored in the Results database. Since the text documents generated by the speech and text recognizers are likely to have a higher error level than normal text, the search engine must allow for a certain error percentage (e.g. match 'Pnarmacla' with 'Pharmacia').

The *User Interface* is preferably implemented to be used within a standard HTML browser, e.g. Netscape Navigator. Immediately (or as soon as possible) after the conclusion of the program, the GUI displays a list of new "hits" derived from the result documents in the results database. If the user chooses to view a news document, an MPEG player starts playing the program file at the given start time, while the corresponding portion of the master text document is displayed in another window. Since the individual documents are not physically segmented, it is possible to rewind/forward beyond the bounds given by the result documents. The user can mark the result as "read", which excludes it from the list of new hits, but makes it stay in the database for later retrieval, or "irrelevant", which removes it from the results database permanently.

4 Shot Boundary Detection Techniques – A Survey

It has been noted in section 2 that two fundamental problems of content-based television filtering concern *extraction* of content information and temporal *segmentation* of the data. The problem of extraction has been given the most attention in this project. In this section, however, one of the questions primarily concerning segmentation is addressed. The section contains a survey of shot boundary detection techniques. This survey is by no means complete; several other papers addressing the issue have been written, including other surveys.[7] Hopefully, it does, however, present in sufficient detail some different algorithms.

4.1 Methods for uncompressed data

Otsuji and Tonomura proposed a pixel-based approach to detection. For each frame, they computed "IDarea": the number of pixels whose intensity differed from the corresponding pixels' in the following frame, by more than a given threshold. Cuts would result in large values of IDarea, but so would high degrees of camera and/or object motion. Slow motion, animation and other visual effects would furthermore yield discontinuities in IDarea. To avoid these problems, they designed a "projection detecting filter", $P_{T1,T2}$, which performs a series of operations choosing maximum and minimum values in sliding temporal windows, with the effect that only peaks in IDarea which last less than *T2* frames and are at least *T1* frames apart, pass the filter. Using a $P_{4,2}$ filter and declaring cuts where the filter output was above a threshold of 24%, resulted in detection of almost all (97%) cuts in a 24 minute news video, with no false detections, in Otsuji's and Tonomura's own experiment. Their work focused on abrupt shot transitions (cuts) only. For gradual transitions, they proposed that their method be used with an extended time scale [OtsujiTonomura93]. It should be noted that the aforementioned threshold seems to have been chosen empirically from the evaluation data, which probably to some extent explains the algorithm's extremely good performance.

Corridoni and Del Bimbo proposed a method of statistically evaluating differences in image histograms. For *cut* detection, they divided rgb-frames into 6×4 blocks and computed the first three statistical moments (mean, variance and "skewness") of the histograms for each block and color. As a partial distance measure for each block, they computed a weighted sum of absolute moment differences between adjacent frames[8], and added together those sums corresponding to the three color components. After discarding the blocks with the highest differences, in order to make the system more robust for object motion, they summed the partial distance measures of the remaining blocks. The resulting metric was computed for each frame and divided with the second highest in a sliding temporal window of five frames. If the quota exceeded a certain, empirically determined, threshold, a cut was considered detected. [CorridoniDelBimbo96, CorridoniDelBimbo95].

For *fade* detection, Corridoni and Del Bimbo used the luminance/chrominance color scheme, instead of RGB, and a pixel-wise approach, based on the assumption that fades result in gradual, continuous increases or decreases in luminance, but only very small changes in color. For each frame they computed a metric based on the number of pixels whose luminance could be described as a monotonous function (over a 16 frame window) and whose color components did not differ by more than a certain threshold between consecutive frames. This metric was penalized if the spatial distribution of "fade-behaving" pixels was not uniform. High values were considered to indicate fades. Similarly, a *dissolve* was considered detected by the presence of two consecutive maxima in the number of monotonous luminance pixels, with a maximum in the number of near-constant chrominance pixels occurring in the valley between them, again with a spatial uniformity constraint imposed. When testing their algorithms on approximately 20 hours of video, Corridoni and Del Bimbo reportedly achieved 84-99% recall and 96-100% precision in detecting cuts, fades and dissolves. [CorridoniDelBimbo96]

---

[7] Sections 12.2 and 13.1 of [Furht&al95] contain a rather extensive and detailed survey. [BoreczkyRowe96] presents a performance evaluation of five common algorithms. In [AhangerLittle96], shot boundary detection techniques are surveyed in a greater context of video segmentation and indexing.

[8] In equation (2) of [CorridoniDelBimbo96], the absolute bin-wise histogram differences are added to this sum. However, the text of both papers gives the impression that only the moment differences were used.

Corridoni and Del Bimbo based their gradual transition detector on the ”Uniformity Measure” presented in [Hampapur&al96]. The authors of this often referred-to paper, claim to introduce a novel, ”top-down” rather than ”bottom-up”, approach to digital video segmentation. Based on ”production models” of video edit effects, such as cuts, fades, dissolves and wipes, they proposed a number of feature detectors that would indicate the presence of such effects. For instance, they concluded that fades and dissolves should be indicated by the presence of a (near-) constant ”chromatic image”, defined as the output video, differentiated over time, divided (pixel-wise) by the original video being scaled. Unfortunately, Hampapur et al's and Corridoni and Del Bimbo's papers have in common that, although the ideas behind their proposed algorithms are comprehensible and seem reasonable, it is unclear which exact algorithms they actually used. The presented formulae are mathematically inconsistent and erroneous, and contain references to functions not defined or explained.

In a recent paper, Hanjalic et al proposed yet another cut detection method based on comparing relative histogram differences within temporal windows. They defined the ”frame-to-frame difference time function” *FFD*(*k*), as a sum of bin-wise absolute histogram differences between adjacent frames. Over a sequence of frames, *FFD* should have an approximately Gaussian distribution,[9] with mean and variance depending on the degree of camera and/or object motion. Abrupt shot changes, ie cuts, yield peaks in FFD. Their solution was thus to detect those peaks, by examining *FFD* values within a sliding window. A cut was considered detected if the center value in the window was: a) the largest value within the window, and b) larger than a locally computed threshold. The threshold Hanjalic et al used was computed as the mean of the *FFD*-distribution in the window plus a factor α times the standard deviation. They reported very good performance for $\alpha = 5$. The described algorithm handles cuts only; gradual shot transitions ”require more elaborate detection mechanisms.” Hanjalic et al noted that their proposed method could be used fruitfully on the DC sequence only of DCT-compressed video (such as MPEG or ”Motion-JPEG”). It is however unclear if they have actually implemented such a solution. [Hanjalic&al97]

4.2 Methods for compressed data

The term *DC sequence* was introduced by Yeo and Liu to describe a sequence of *DC images* – downscaled images where each pixel represents an average of a number of original pixels. From JPEG-compressed frames and I-frames of MPEG video, DC images are easily obtained by downscaling the DC coefficient of each block by the block width, usually 8 (cf. sub-section 3.2.3 and footnote). For MPEG P- and B-frames, Yeo and Liu proposed a method to reconstruct approximate DC images, using block motion vectors and the DCT coefficients of preceding I- and P-frames. Given a DC sequence, they performed shot transition detection, using a pixel-based approach. While still more motion sensitive than histogram differences, pixel differences between DC images are more robust than pixel differences between uncompressed frames, since each DC-pixel represents an average over 64 original pixels. Yeo and Liu defined a difference metric as the sum of absolute differences between the luminance values of each pixel of two consecutive frames and then used the same evaluation method as Corridoni and Del Bimbo; ie, a cut was declared at a certain frame if the corresponding difference metric was both the largest within a sliding window and n times larger than the second largest. Yeo and Liu achieved best performance for a window length of 10 frames and $n = 2$. For detecting gradual transitions, they used a similar approach, but based the difference metric associated with each frame on the pixel differences k frames apart. A gradual transition was considered detected if the metric reached a ”plateau” of at least $2m$ frames length and of a height l times the difference value outside the plateau. Best performance was reported for k=20, m=5 and l=3, yielding 86% recall and 60% precision on an 11 minute video with seven actual gradual transitions. [YeoLiu95]

The advantage of performing image analysis on DC images lies mainly in the reduction in image size, with a corresponding reduction in the number of required computations. However, not very much information from the

[9] According to the *Central Limit Theorem*, under general conditions, the sum of a large number of random variables is approximately normally distributed [RådeWestergren90, p 375].

computations performed in DCT-encoding is used. One of the first shot boundary detection methods using DCT-compressed data was proposed by Arman et al. In contrast to Yeo and Liu, they used a subset of the *AC* coefficients. As they are 2-D "frequency" components, they contain information on the finer-grained spatial structure of the image. Arman et al formed a feature vector for every frame, composed of α randomly chosen AC-coefficients for each of ρ pixel-blocks. The ρ blocks, which were a small subset of all blocks, were chosen to form n connected regions in each frame, either randomly or with some a priori knowledge of which parts of the image were most likely to carry good shot-discriminating features. Once subsets of AC coefficients and blocks were chosen, the same sets were used for all frames. To detect cuts, Arman et al computed the normalized inner product between feature vectors of frames, φ frames apart. If this inner product was close to zero (within a certain threshold), a cut was declared in the interval between the frames. The algorithm was tested on "Motion-JPEG" and MPEG videos; in the latter case, only I-frames were examined. Unfortunately, Arman et al's paper, does neither state which values of α, ρ, n and φ were used, nor the algorithm's performance in terms of recall and precision [Arman&al93]. A very similar algorithm was tested in the survey [BoreczkyRowe96], however, and gave considerably poorer performance than methods based on histogram comparisons between uncompressed frames.

Ariki and Saito also formed feature vectors with DCT-coefficients for each frame. However, they used significantly larger image blocks and, more importantly, they approached the segmentation problem from the "opposite direction" of the algorithms previously described. Instead of trying to identify shot transition points by their own features, they set out to form clusters of frames based on similarity and then declare these clusters different shots. They divided each 8-bit gray level frame into 6×8 blocks (of 40×40 pixels) and computed the first 3 DCT coefficients for each block. From these they formed feature vectors of dimension 144 (6×8×3). Under the assumption that a shot lasts at least one second (at a frame rate of 30 fps), they took the feature vectors of the first 30 frames of a video and computed mean and standard deviation (σ) vectors. Then for each new frame they counted the number of elements in its feature vector that deviated from the corresponding mean by more than 3σ. If more than half the elements deviated thus, the frame was (momentarily) considered outside the cluster. When more than ten consecutive frames were declared outside, they formed a new cluster, with new mean and σ vectors. Otherwise, they were considered to belong to the original cluster, whose mean and σ vectors were updated with the new values. The algorithm was suggested to be more robust towards short abrupt image changes not associated with cuts, than algorithms based on finding large differences between consecutive frames. Ariki and Saito tested the algorithm on 45 minutes of news videos with 455 actual cuts and reported results of 88% recall and 91% precision. Their paper does neither discuss gradual transitions nor computational cost. They did use a hardware JPEG encoder to compress the images, but as mentioned before, the DCT blocks used in the clustering algorithm were 25 times larger than the normal 8×8 pixel blocks used in standard JPEG compression. [ArikiSaito96]

4.3 Other algorithms

In addition to the algorithms presented, a number of other techniques have been proposed in literature and implemented in various digital video retrieval or editing systems. A method based on the rate of change in color correlation between frames was used in Hitachi's IMPACT system [Ueda&al91]. IBM's QBIC system uses a combination of pixel correlation and histogram difference measurements [Flickner&al95]. In the Singapore News Parser, and related projects, a hybrid method for MPEG segmentation has been employed, using a combination of DCT coefficient differences and motion vector information in B- and P-frames [Furht&al95, Zhang&al97]. Other approaches to MPEG segmentation have involved comparing the number of backward-predicted or intra-coded motion blocks to the number of forward-predicted blocks in B-frames [MengChang96], and computing overall, row and column histograms for intensity DC images from adjacent I-frames, using the χ^2-test[10] to determine if the histograms were from the same distribution [PatelSethi97b]. A very fast and simple, but not very accurate, method to detect cuts in Motion *JPEG* video is to compare the compressed image file sizes, as adjacent frames from the same shot should have approximately the same content and consequently the same size [AhangerLittle96].

[10] cf. e.g. sect. 18.5 of [RådeWestergren90]

Recently, Wei et al presented a multi-pass algorithm for segmenting MPEG-compressed video, integrating some of the above mentioned techniques. They defined four classes of video sequences, based on their visual content, and organized in a tree-like structure. Two were considered ”boundary classes”: *breaks* (cuts) and *gradual transitions*, and two ”non-boundary classes”: *large motion* and *noise*. In a three-step classifier (cf. fig. 4.1), they used different feature metrics to discriminate between the classes. First, they separated the breaks from the other classes by counting the number of motion blocks in MPEG B-frames that were either bi-directionally interpolated or skipped. In the general case, most of the motion blocks in a B-frame should be bi-directional. However, if there is a cut between two reference frames, ie. I- and P-frames, the MPEG encoder is forced to unidirectionally predict or intracode the motion blocks of the intermediate B-frames. Therefore, Wei et al declared a break detected if, for all B-frames between consecutive reference frames, the number of bi-directional or skipped blocks was lower than a certain threshold. Having thus initially segmented the video stream, the next step was to separate noise sequences from the two remaining classes. This was done by computing two difference metrics between DC-images of consecutive I-frames: the sum of absolute bin-wise intensity histogram differences and the sum of absolute pixel-wise intensity differences. Local maxima in these metrics indicate gradual transitions and/or large motion, and sequences were classified as to belong to one of these classes if either they contained a simultaneous local maximum in both metrics (over a certain threshold to filter out local maxima occurring in valleys), or the pixel difference had a ”local abrupt change” – a local maximum at a significantly higher level than the surrounding values. Finally, to separate between gradual transitions and motion, Wei et al applied a modification of the ”constant (chromatic) image” approach proposed by Hampapur et al. Using Yeo and Liu's method, they reconstructed approximate DC-images of all frames in the remaining sequences and computed relative difference images[11] between consecutive frames. In the case of an ideal fade, these difference images should be constant, and Wei et al used the pixel value variance of the difference images as a feature detector, classifying sequences with low variance as gradual transitions. [Wei&al97]

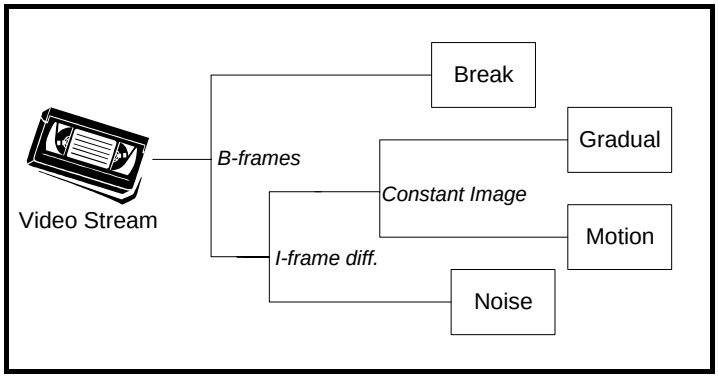

Fig. 4.1. The three-step classifier of [Wei&al97].

[11] (Frame 2 – Frame 1) / Frame 1

5 Image Text Extraction & Recognition – Prototype

The three, beyond comparison, most important sources of content information in a television program are teletext subtitles (or closed captions), text imbedded in the image, and the spoken dialogue. Teletext subtitles are the easiest to handle, but as tables 3.1-3 show, they only cover a smaller subset of the Scandinavian programs. Speech Recognition makes it possible to use the greater part of the available information, but speaker independent, open vocabulary, continuous SR is yet an unsolved problem. Although SR is successfully used in some of the systems described in sub-section 2.5.3, there are so far no available Swedish speech recognizers which could be used in a TV filter, and building one from scratch would be a far too arduous task for this thesis project. The image-embedded text may perhaps only carry fragments of the information, but they are important fragments. Since it is my firm opinion that a content-based TV filter for use in Scandinavia must contain a video text module (cf. fig. 3.1), I have built a prototype for such a module, primarily to prove that it can be done. This section presents design and implementation. Performance results and suggested improvements are discussed in sections 6 and 7.

5.1 Embedded Text

Text embedded in the video image appears in several different forms with different functions. I distinguish between five types of text:

1. *Subtitles* – containing a condensed representation of the dialogue, either translated from a foreign language or provided as a service to the hearing-impaired audience. (In eg. the United States, a voice-over would most oftenly be used in the first case and closed captions in the latter.)
2. *Captions* – short information on the name of a person speaking, a location, a topic, etc.
3. *Tables* – tabulated information, eg. sports results, opinion poll numbers, stock prices, etc.
4. *Credits* – usually appearing at the beginning and/or end of a program, containing the names of people involved in the production, their functions, and other related information.
5. *Other* – eg. incidental text appearing in the image, on cars, t-shirts, signs in the background, etc.

Fig. 5.1. Examples of Captions and Subtitles (left) and Other text (right). Images from "Aktuellt". © SVT, 1997.

This implementation has been focused on subtitles, mostly because they are widely used in Scandinavia and often complement the information available from a teletext decoder or a (mono-lingual) speech recognizer, but also, admittedly, because they are the "easiest" form of text to extract and recognize. Subtitles do not vary much in font and are almost invariant in size. Furthermore, since they are considered of vital importance to the audience's comprehension of the program, they are often optimized for visibility, in a way that also facilitates computer processing. For example, Swedish National Television (SVT) always displays subtitles with white text on solid black bars.

The order of the text types in the list above primarily reflects my opinion on their relative relevance in a content-based TV filter, but this order fortunately also happens to reflect the ease with which they may be extracted and recognized.[12]

I have made the following observations, which are true for most text of the three first, ”most relevant”, types:

- Text consists of several characters on horizontal rows, beginning at the left side of the image.
- All text on one row is of the same size and font.
- The text size (x-height) varies between 7 and 20 pixels (in a 288×360-pixel frame).
- Text has high contrast to the image background.
- Text is faded in and out rapidly and is stationary in the meantime.
- Every text stays in the picture for at least two seconds.
- Text may remain in its place over shot boundaries. (Is especially true for subtitles.)

Most of these observations are used in my implementation of the prototype. (The property of being stationary, however, makes no difference whatsoever.) For the last processing steps of the extractor/recognizer, I have imposed some additional constraints, which have the effect that at present, only subtitles are handled correctly. I believe that it should be fairly easy to remove these constraints in possible future versions:

- Text is bright (white) on a dark background.
- The x-height is exactly 9 pixels and the maximum character size is 19×13 pixels.

5.2 General Approach

As mentioned in sub-section 3.4.2, the problem of extracting embedded text can be approached mainly from an ”intra-frame” (every video frame is treated independently) or an ”inter-frame” (text is identified in differences between consecutive frames) point of view. My approach is strictly *intra*-frame, in part because the assumptions on which inter-frames algorithms are built are not valid for text across shot boundaries (eg. subtitles), in part because it was easiest to start with and turned out to work well.

I work with gray scale images, converted if necessary from color, which simulates processing of the luminance signal (cf. 3.2.1) only. I subsample PAL-images from 576×720 to 288×360 pixels. This gives savings in data amount and computations time with a factor of four.[13] The reduction of image resolution is compensated by a corresponding noise reduction, and preliminary investigations indicated that using full resolution images would not lead to better performance. It also circumvents the problem of deinterlacing.

The presented prototype consists of several sub-modules, which, in order: identify larger image blocks which may contain text, extract the actual text segments, segment them into separate character images, extract feature vectors for each character image, and, finally, identify each character. This multi-step approach is similar to the one taken in eg. NTT's system (cf. sub-section 2.5.2 and [Kurakake&al97]), although the algorithms for the individual steps are different. The extraction and segmentation steps use a combination of image processing techniques, statistics and heuristics. Character recognition is performed using a computer-simulated artificial neural network (ANN). The process flow within the video text module and the interactions between the module and the ”outside world”, or a larger system, is shown in figure 5.2. With reference to the overall system of figure 3.7, the first sub-module, the ”pre-filter”, should be considered part of the Video Pre-processor, rather than the Text Extractor and Recognizer.

---

[12] Assuming that: a) what is important to the filter is also important to the TV audience, b) more important information is intentionally made more visible, c) better visibility means easier computer handling, maybe it's not such a coincidence...

[13] On the computer I used, a Pentium166 with 32 MB RAM running Windows 95, the actual savings were even higher, since full sized images caused the system to swap data frantically between RAM and HD.

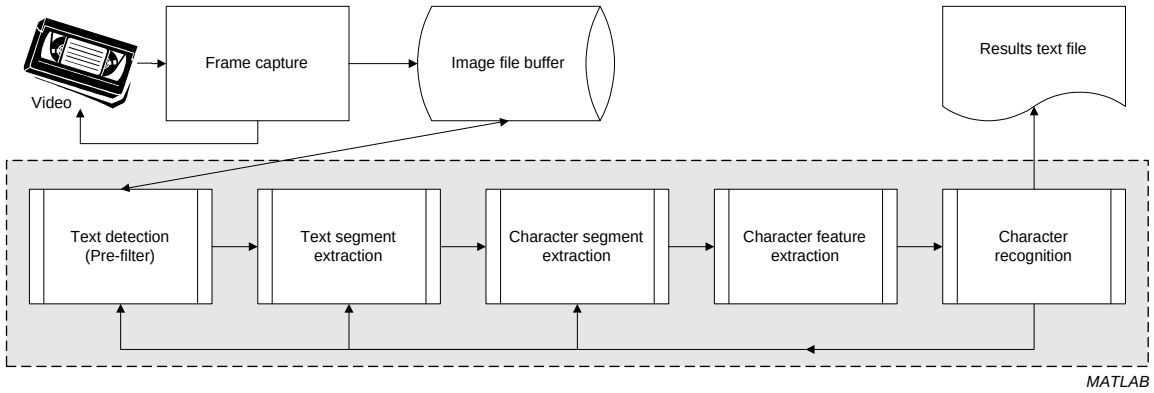

Fig. 5.2. Process flow in the Video Text Module Prototype. The bottom row contains the five sub-modules.

The sub-modules are implemented as *MATLAB* functions [ML97]. MATLAB provides simple and computationally efficient handling of large sets of structured numerical data, eg. digital images, and it is the de facto standard development environment in a wide variety of scientific computing contexts. Over other, dedicated image processing, software, MATLAB has the advantage that, as images are treated as ordinary matrices, any type of mathematical operation can be performed with them, and every single pixel is easily accessible.

In addition to giving complete access to data for low-level manipulation, MATLAB allows advanced high-level functions to be built in a C-like programming language. There are add-ons to MATLAB, ”toolboxes”, containing libraries of functions for different computational tasks. One such add-on is the *Image Processing Toolbox* [MLIPT97], which contains several functions for image manipulation used in this prototype.

MATLAB functions are stored in ”m-files”, text files that are interpreted at execution. This results in fast prototyping, but ”slow” execution. In terms of computational speed, MATLAB is optimized for vector and matrix operations, while iterations of scalar operations are notably time-consuming. MATLAB can call compiled C-functions (”mex-files”) which execute up to ten times faster than the corresponding uncompiled functions. The makers of MATLAB, The MathWorks Inc., market a ”MATLAB Compiler”, which converts MATLAB code to C [MW97]. It has not been tested in this project, but would most probably help speed up the prototype. It is also possible to build stand-alone applications, using the compiler.

5.3 Text Detection / Pre-filter

The first step of the text extraction/recognition process is to quickly identify those sub-frames in the entire video material which contain text, so that the more time-consuming image processing steps later in the process are only performed on relevant data. This step, which has the function of a ”pre-filter”, is designed on the premise that the complete TV filter system is to work with MPEG-compressed video (cf. 3.2.5) and the idea is that the sub-module should perform its filtering *without decompressing* the data. I present a novel algorithm, based on a suggestion in [Arman&al93], that edges and other image features could be detected by examining certain AC coefficients of DCT-compressed image blocks. It differs from the compressed-domain text detection algorithm presented in [YeoLiu96], in that features *inherent to text* are detected directly in the compressed data stream, whereas Yeo and Liu looked for local scene changes in partly decompressed video. Arman et al did *not* mention text detection as a possible use of their algorithm, but it seems reasonable to assume that text, which clearly contains numerous sharp edges, should yield detectable features in the frequency domain. Comparing pixel rows from text and non-text regions indicates that the assumption holds, as is shown in figure 5.3.

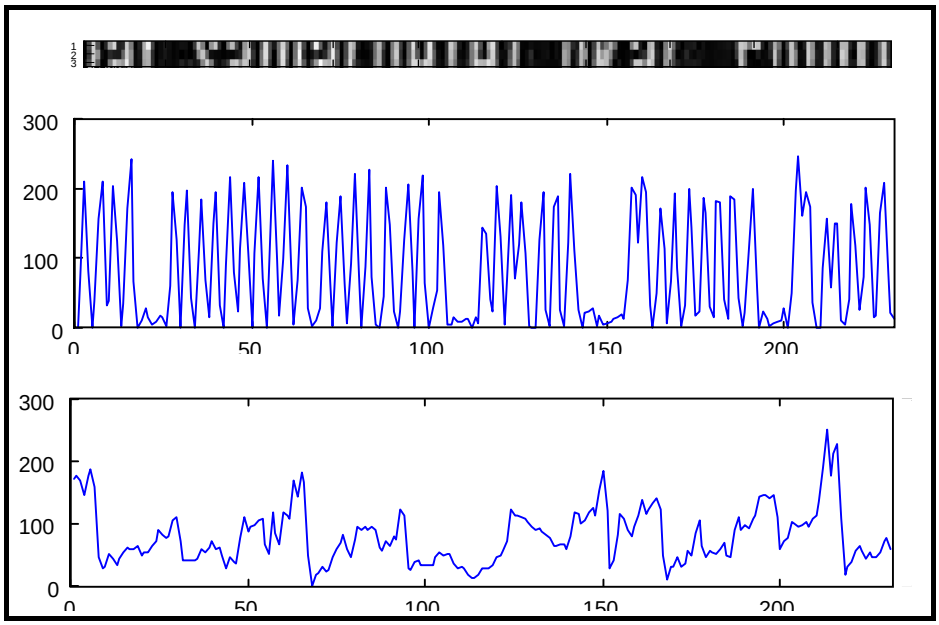

Fig. 5.3. Top: 3 pixel rows from an image segment containing text (subtitle).
Middle: pixel intensity values for the topmost of these rows.
Bottom: pixel intensity values for a row from a segment high in content, but not containing text, in the same image.
Clearly, the text segment has greater high frequency energy than the non-text segment.

Assuming thus that text means high frequency, the key part of my algorithm is to sum those AC coefficients in every 8×8 DCT block which correspond to mainly horizontal frequencies (cf. figure 5.4) and to add together these sums for each row of DCT blocks in the image. Whenever such row sums exceed a threshold, text is considered to have been detected. Figure 5.5 (next page) visualizes the algorithm applied on the first image of figure 5.1.

d	A	A	A	A	A	A	A
a	a	A	A	A	A	A	A
a	a	a	A	A	A	A	A
a	a	a	a	A	A	A	A
a	a	a	a	a	A	A	A
a	a	a	a	a	a	A	A
a	a	a	a	a	a	a	A
a	a	a	a	a	a	a	a

Fig. 5.4. The predominantly horizontal AC coefficients of each block, marked with A*, are summed in the text block extractor.*

The complete algorithm, step-by-step, is as follows:

1. For each 8×8 block, sum the 28 ”horizontal” AC coefficients.
2. Sum these sums for the *left* half of each row of blocks.
3. Divide the image into four horizontal macro blocks.
4. Mark a macro block as a text block if any row sum in it exceeds an empirically determined threshold (I used 15 000, but the figure is likely to depend on the implementation of the DCT-encoder.)
5. Decompress text blocks and pass them on to the next processing step.

The algorithm was implemented as a MATLAB m-file function, `JLDCTX`, the source code of which is listed in appendix A. In a ”real” system, the idea is to extract luminance I-frames directly from the MPEG stream. This functionality has however not been implemented in the prototype. Instead, the process is simulated by computation of DCT coefficients in decompressed images, converted to gray scale. Although the actual filtering algorithm is very fast, block-wise DCT transformation in MATLAB is time consuming (approximately seven seconds per 288×360 image, on a 166 MHz Pentium), and so the pre-filter, in its present form, is not suitable for real time video

processing. Therefore, an alternative pre-filter has been designed and implemented. It is based on the DCT-version, but instead of AC coefficients, a set of standard deviations are computed and summed for the left half of every pixel row.[14] Using a different threshold, the rest of the algorithm is the same as the one above. In terms of performance, the ”compressed” and ”uncompressed” versions are virtually interchangeable. At present, the ”uncompressed” version runs faster, since no DCT coefficients need to be computed, although the core of the algorithm, the decision-making, is more time consuming in the standard deviation case. As stated before, in a ”real” system, DCT coefficients would be computed anyway, most likely in hardware, as part of a general MPEG encoding.

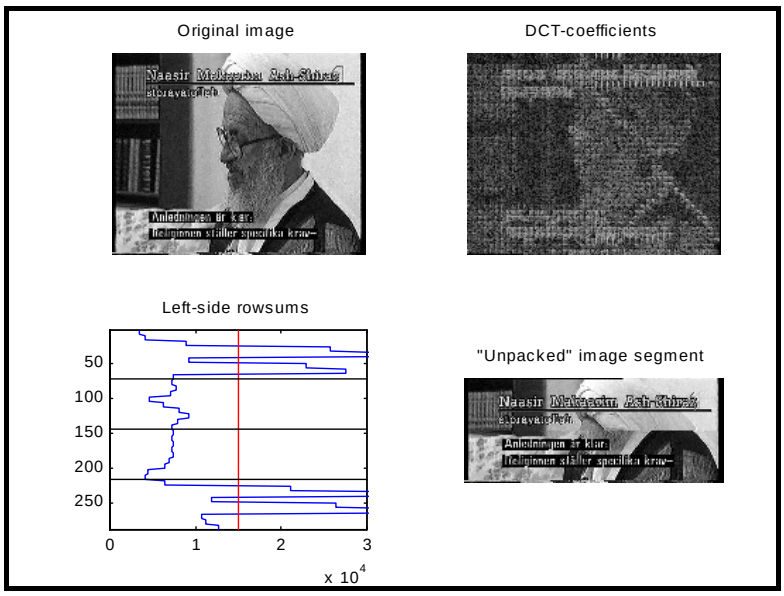

Fig 5.5. Pre-filtering (DCT version) of the first image in figure 5.1. The bottom right image is passed on to the next step of the text extraction process. (TV image from ”Aktuellt”. © SVT, 1997)

5.4 Text Segment Extraction

In those image blocks, hopefully containing text, which pass the text detector, the next step is to extract exactly the segments that consist only of text, ie words and the spaces between them. In sub-section 3.4.2, two standard methods for text segment extraction were presented. However, they both seem overly time-consuming. The processing times reported in [Kim96] and [Zhong&al95] for both methods[15] are far too long for real time or near real time systems. I have therefore developed an adaptation of the ”projecting edges” algorithm suggested in [MotegiAriki96]. The algorithm is based on the description of text regions as ”horizontal rectangular structures of clustered sharp edges” in [SmithKanade97]. It bears resemblance to the ”spatial variance method” as described in sub-section 3.4.2 and eg. [Zhong&al95], but instead of detecting edges in a variance image, the algorithm detects edges directly in the gray scale input image, and operates on the binary image formed by them. The binary edge image is dilated and then eroded with a 1×7 horizontal structuring element, in order to fill the gaps between text edges (”close” is the common name for this morphological operation).[16] Then the original edge image is subtracted from the result (”bottom hat”, [MLIPT97]), with the effect that all edges which are not closely located in horizontal direction, are completely removed. Figure 5.6 shows the result of edge detection and morphological operations, performed on the text blocks extracted in figure 5.5.

[14] The process is similar to the first steps of the ”spatial variance method” in fig. 3.6.
[15] Approximately 6–11 seconds per image (smaller than 288x360 pixels) on a SPARC 20 workstation.
[16] cf. eg. [GonzalezWoods92, sect. 8.4]

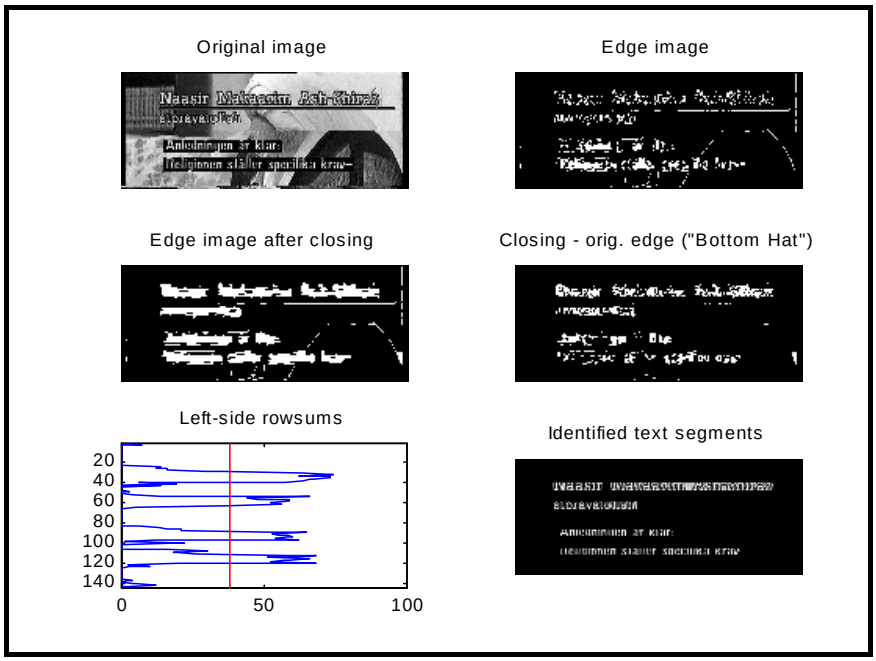

Fig. 5.6. Edge examination and text segment extraction, performed on the text blocks extracted in figure 5.5.

The next step in the algorithm is to identify text rows. It bears some resemblance to computing the ”y-signature”, as described in [Kim96]. For each pixel row in the ”bottom hat” image, the number of set pixels from column 40 to the middle column (180 in a 288×360 image), is counted. An adaptive threshold is defined as twice the mean of all such sums, constrained to at least 1/40 and at most 1/8 of the total image width. An example of row sums and corresponding threshold is displayed in the bottom left sub-image of figure 5.6. When at least seven consecutive row sums are above the threshold, they make up a text row.

Within the detected text rows, text segments are identified in a very similar manner. For each text row in the ”*closed*” image, the number of set pixels is counted for every column. One half of the largest such sum is used as a threshold. The largest possible segments, beginning and ending with column sums above the threshold, and where the sums are not below the threshold for more than 16 consecutive columns, are extracted as text segments. Segments that are at least 30 columns wide are indexed by their corner positions. This index is later used to identify those portions of the decompressed image, on which character segmentation should be performed. The bottom left image of figure 5.6 shows extracted text segments in their original positions.

An additional step, between row and segment extraction, is currently implemented in order to facilitate character segmentation and recognition. It is designed for use with subtitles (especially the ones used by SVT) and is rather ad hoc. The purpose is to make sure that only and exactly the text portion containing the lower case character bodies (the ”x-height”) is extracted. This is achieved by summing the intensity values in every pixel row of each detected text row. The pixel rows with the smallest sums are removed, so that exactly 9 (the sought-for x-height) rows remain. Then a new text row is pointed out by the top- and bottom-most of the remaining pixel rows. If the processed text row contains SVT subtitles, the new text row will in almost all cases contain only the desired x-height portion. For other lower case text, the result of the algorithm step is most often also that only the x-height portion (but not necessarily of height 9, though) is extracted. When text rows of height lower than 9 are detected, a number of pixel rows are added at the top and bottom before the described computations are performed. This means that the new text rows will always have a height of at least 9 pixels. Naturally, this is a problem if the actual x-height of the detected text is smaller than 9. Hopefully, it should not be too difficult to either make this algorithm step more general, or improve the character segmentation/recognition submodules so that it can be omitted entirely. (Currently, omitting this step degrades performance by approximately 20%.)

The complete, step-by-step, text segment extraction algorithm is as follows:

1 Perform edge detection in the original image and make a binary image, where pixels corresponding to edges are set to one and the rest to zero.
2 Morphologically close (dilate + erode) the binary image, with a structuring element of 1×7 pixels.
3 Create a new binary image by subtracting the original edge image from its closing ("bottom hat").
4 For each pixel row in the bottom hat image, count the number of ones between column 40 and the middle column.
5 Compute a threshold as twice the mean of those sums, constrained to at least 1/40 and at most 1/8 of the number of pixel columns in the image (9 and 40, respectively, for a 288×360 image).
6 Where at least seven consecutive sums are above the threshold, let the corresponding pixel rows constitute a text row.
7 If a text row consists of any other number of pixel rows than *nine* ("Ad hoc" algorithm):
   7.1 If the number of pixel rows is *less* than nine, expand the text row by one or two pixel rows each above and below.
   7.2 Sum the original *gray scale* values in each pixel row of the text row.
   7.3 Remove the pixel rows with the smallest sums, so that exactly nine rows remain.
   7.4 Let the top- and bottom-most of the remaining pixel rows point out a new text row.
8 For each (new) text row in the *closed* image, count the number of ones in each column.
9 Compute a threshold as half the largest such sum.
10 Mark as text segments the largest portions of each text row, which begin and end with a column sum above the threshold and where not more than 16 consecutive column sums are below the threshold, discarding segments that are less than 30 columns wide.

This algorithm has been implemented in the MATLAB function JLTEX, which is called with a gray scale image matrix, or the name of a file containing it, as input parameter, and returns an index with the corner positions of extracted text segments. Several of the numerical values in the presented algorithm are merely the defaults used in the prototype. When calling JLTEX, it is possible to supply other values.

5.5 Character Segment Extraction

The text segment extractor outputs rectangular image regions containing several text characters and background. To allow the characters to be recognized correctly, they have to be individually segmented, while as much as possible of the background is discarded. This is the function of the *character segment extractor*.

As mentioned before, for each step of the process, more and more heuristics based on *subtitle* features are used. In this step, text is assumed to be white on a uniform dark background. All characters in a text segment are assumed to be of the same font size and placed on a common baseline. Every individual character is assumed to be horizontally cohesive, but always disjunct from other characters. It is further assumed that the segments provided by the text segmentation step contain only the x-height portion of the text, which implies that the text consists mainly of lower case letters.

There are also some demands imposed by the remaining steps in the recognition chain. The character feature extractor expects to receive character images of exactly 19×13 pixels size, and the neural network is trained only with characters that are centered horizontally in the image and placed vertically on a specific baseline (between the third and the fourth pixel row from the bottom, to be exact). In order to identify word boundaries correctly, the network is trained to recognize blanks, rather than to ignore them. Therefore, character images representing inter-word spaces should be output from the character segmenter, as well as images containing characters, and they must all conform to the size and alignment requirements of the succeeding processing steps.

Given these assumptions and demands, the chosen solution is to identify the background space between characters, by examining the maximum gray-level value in every pixel column of the text segments. Narrow valleys in this metric indicate inter-character space, while wider valleys should be encountered in the spaces between words. Two dynamically calculated thresholds are used to identify the background. Another, user-defined, threshold for the width is used to decide whether a portion of the identified background should be output, representing a blank. In the general case, the background pixel columns are discarded and the columns between them are extracted as character images, which are padded with zero (black) columns evenly at the sides. Applying this algorithm to the first subtitle row of figures 5.5-6, results in the output displayed in figure 5.7.

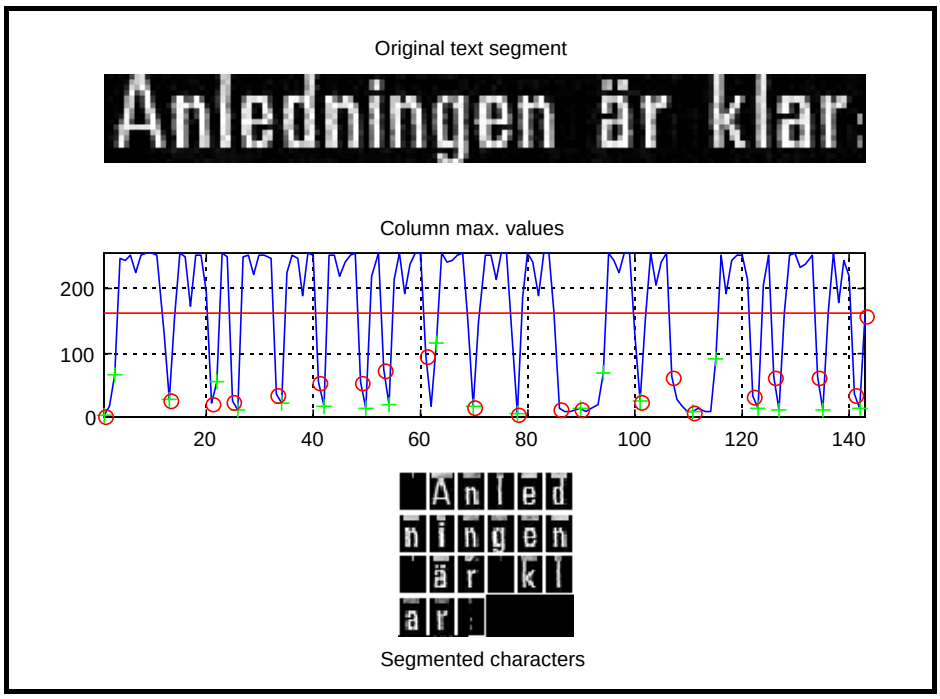

Fig. 5.7. Character segmentation in the first subtitle row of fig's 5.1a and 5.5-6. The middle graph shows the maximum pixel values in each column. The straight line marks the mean of those maxima and the +'s and O's mark the beginnings and ends, respectively, of detected character segments.

Identifying the background (which, in a sense, is the direct opposite of computing the ”x-signature” of [Kim96]) is actually easier than identifying the character foreground, since the background has more unambiguous features. This is, of course, heavily dependent upon the assumption that the background is black. The presented algorithm includes a check step, which inserts forced character breaks at the deepest valleys in identified ”characters” that are too wide. The effect is that the algorithm works with negative (bright on dark) text that *does not* have a uniform black background, as well, but with poorer performance. Typically, single bright pixels in the background may incorrectly be interpreted as (narrow) characters, and segments of seemingly conjoined characters may be split at the wrong places. Correctly identified characters may not be as well centered in the output character images.

Since the input text segment only contains the x-height portion of the text, the computations are performed after the segment has been expanded by four pixel rows above and three below, to include the ascenders and descenders of lowercase letters such as *b* and *j* and the upper part of uppercase letters, but before an additional three pixel rows are included at the top, to include the ”accents” of the uppercase Swedish characters *Å, Ä* and *Ö*. This is a rather ad hoc solution, based on the fact that the background bars of SVT subtitles do not extend above the top of other uppercase characters, unless there are *Å*s, *Ä*s or *Ö*s present. This would lead to problems, as discussed in the preceding paragraph, if the topmost pixel rows were included during background identification.

The presented version of the character segment extractor uses the following, step-by-step, algorithm:

1 Expand the input text image regions by four pixel rows above and three pixel rows below.

2 Ignore the text segment if the obtained height is not exactly 16 pixels[17]
3 Create a comparison array with the highest (brightest) gray level value of each pixel column in the text segment.
4 Define a pixel column as part of the background if, either:
   - both the column and at least one of its immediate neighbors has a comparison value below one half of the mean value in the array
   - the column has a comparison value below one eighth of the mean, *or*
   - the column has the lowest comparison value in an interval longer than 13 columns, between two pixel columns of the background, as defined above.
5 If more than three consecutive pixel columns are considered part of the background, split them into two groups (to mark intra-word space).
6 Expand the text segment with an additional three pixel rows above (to handle *Å*, *Ä*, *Ö*).
7 Extract groups of pixel columns of the new text segment, between background columns, as character segments.
8 Add zero-valued (black) pixel columns evenly at both sides of the extracted segments, so that each character segment is centered in a 19×13-pixel image.

This algorithm has been implemented in the MATLAB m-file function **JLCHSEG**. It takes the original gray scale image matrix and the output index from **JLTEX** as input arguments, and outputs an array of unisize character image matrices. The array is technically four-dimensional, rather than 3-D, in order to conform to some of the functions in MATLAB's Image Processing Toolbox. In practice it is 3-D, since one dimension, reserved for color information, is not used.

As in the case of the text segment extraction algorithm and **JLTEX**, numerical values are defaults, based on SVT subtitle characteristics, and can easily be replaced when calling the function. For instance, the exact numbers of pixel rows added, before or after background identification, are not significant for the segmentation algorithm in itself. It could be argued that this modification of the text segment should actually be made in the text segment extractor, rather than in this sub-module. In any case, expanding text regions by a fix number of pixel rows works very well with subtitles, i.e. text of a specific, pre-determined format, but is not a very general solution. In an earlier version of the character segment extractor, the background detection algorithm was applied directly to the output from the text segment extractor, with the result that certain characters that are wider than their x-height parts, such as *T* and *j*, were not as well segmented and centered, while other non-subtitle text was generally better segmented than in the presented version. Another, more general solution would be to identify single characters through some sort of connected component analysis with region labeling within the (expanded) text segments, in a manner similar to the "hybrid method" presented in [Zhong&al95]. This approach has not been tested, though.

5.6 Character Feature Extraction

The extracted character images are numerical matrices of 19×13 = 247 elements. The elements are integers in the range [0..255]. When the matrices are displayed graphically, with higher values represented by gradually whiter gray levels, as in the left half of figure 5.8, a human has generally no difficulty recognizing which character they contain. However, to a computer, they are still just heaps of numbers. As there are, in this case, $256^{247} \approx 10^{595}$ possible matrix combinations, it would of course be completely impossible to create a lookup table for classification. Character recognition must therefore be performed as some sort of comparison to *training examples*, where the closest similarity, by some measure, determines which character the matrix contains.

It is possible to compare the pixel gray levels of the character image matrix directly to those of stored training examples. This method, called (direct) *template matching*, is, however, very sensitive to noise and small geometrical variations and it is not invariant to character size or average brightness [Trier&al96]. Furthermore, the number of

[17] This has really nothing to do with the actual character segmentation algorithm, but is done because the following sub-modules only handle text of a certain size, and it would thus be a waste of time to segment other text.

mathematical operations needed for similarity computation is rather high, since each character image and each template is represented by a vector of, in this case, 247 elements. A better solution is to *extract* from the characters a smaller number of invariant *features*. In addition to reducing the data amount, feature extraction allows more robust character recognition. Some suggested features for video OCR have been "the local directions of contours" [Kurakake&al97] and foreground pixel density in sub-blocks [Ohyo&al94].

The feature extraction method used in this prototype is a slight variation of the method used in the MoCA project (cf. sub-sections 2.5.2, 3.4.2 and [Lienhart96]). All 2×2 pixel blocks in the character image are tested for membership in one of four *direction classes*. The character image is divided into sub-images (3×3 in the MoCA system; I use 8×6). In each sub-image, the number of blocks that belong to each direction class is counted, and these numbers are normalized to make the features size-invariant. In the MoCA system, the result is a feature vector of 36 elements (9 sub-images × 4 direction classes), wheras in the presented prototype, the feature vector has 194 elements (8×6×4 + 2 special "Swedish" features).

To avoid the problems of interpolating and scaling, my version of the feature extractor currently only accepts, as said before, character images of size 19×13 pixels. The size corresponds to the maximum total height of a subtitle text row (with e.g. an *Å* and a *g* in it), and the maximum width of a subtitle character, in the font used by SVT. Since one of the benefits of feature extraction is size-invariance, this restriction must of course be upheaved if a possible future version of the system is to work with other text than subtitles. As of now, parts of the algorithm are rather ad hoc. The first three pixel rows in the character image are removed and used for a special algorithm to detect uppercase Swedish "accents". A blank (black) pixel row is added on top of the remaining rows, and the resulting 17×13 image is divided into 8×6 overlapping sub-images of 3×3 pixels. Every sub-image contains four (overlapping) 2×2 blocks, which each are assigned to one of the four direction classes, or neither. The number of assigned blocks is counted for each class and sub-image, and then divided by four to get a feature value in the range [0..1]. An example of feature values for the letter 'A' is shown in the right half of figure 5.8.

The direction class assignment is performed as a series of boolean operations on binary matrices. In the MoCA system, which uses color segmentation to extract characters, character "images" are binary – pixels belong to either the character foreground or the background. The presented prototype, however, extracts gray-scale character images, which must be binarized prior to operations. To simplify the implementation, the character images are binarized through *global* thresholding. Generally, it would probably be better (and not very difficult) to use *local* or *adaptive* thresholding, but in the rather ideal case of white subtitles on black background, a global threshold works out well enough. This is in any case a much less significant issue than the current size restrictions.

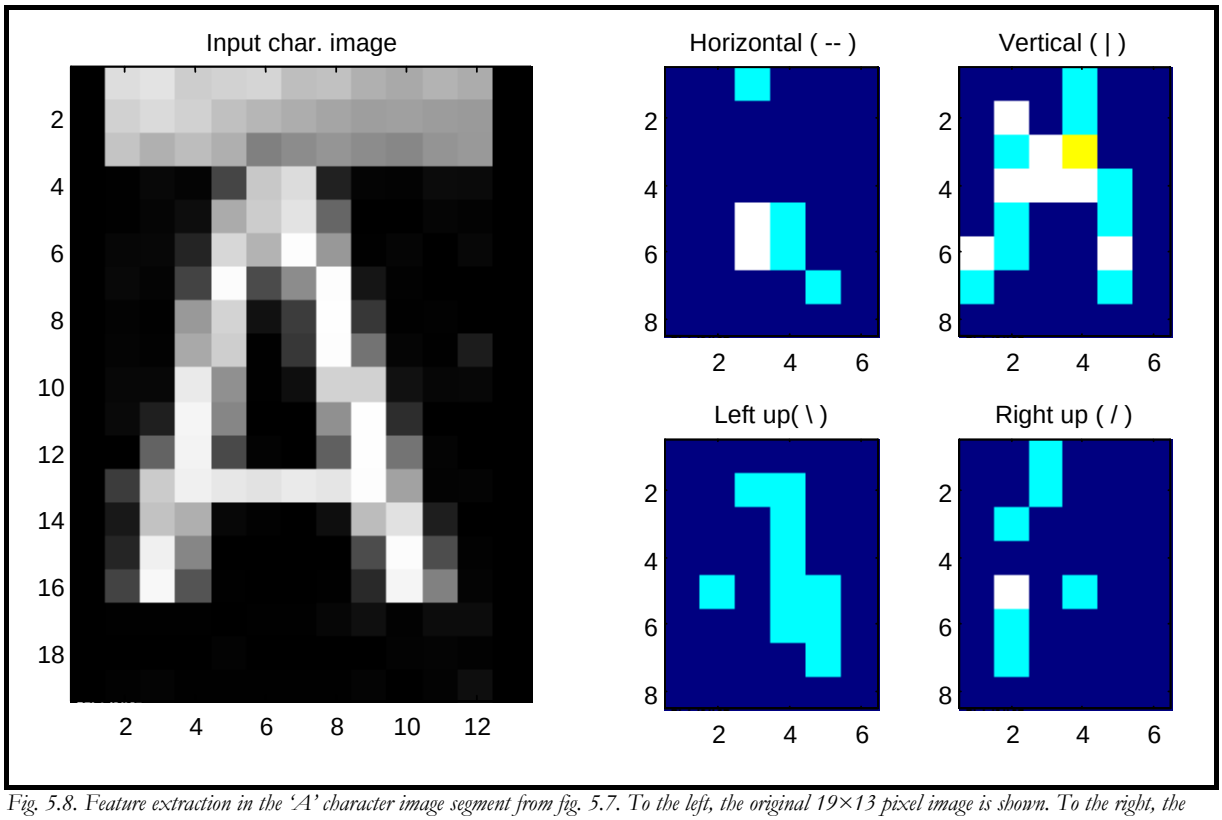

Fig. 5.8. Feature extraction in the 'A' character image segment from fig. 5.7. To the left, the original 19×13 pixel image is shown. To the right, the frequencies of the four different direction classes in the 8×6 overlapping 3×3 pixel blocks. NB! The first three rows in the pixel image are not regarded when computing direction frequencies.

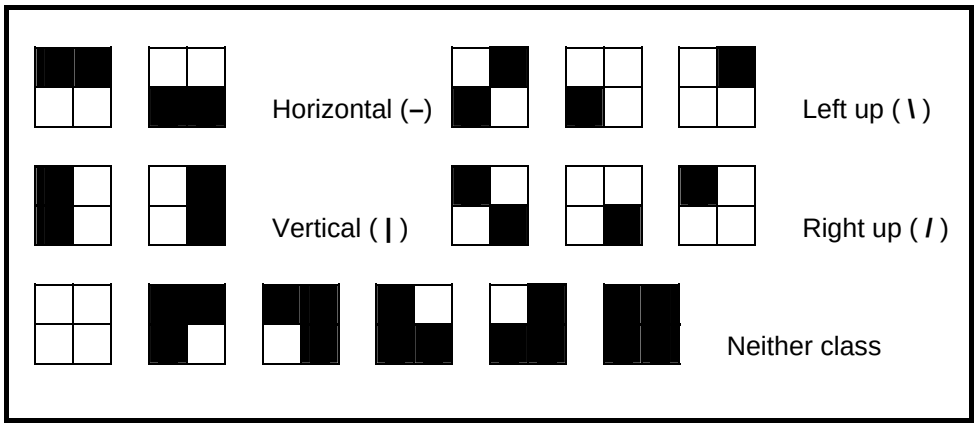

Fig.5.9. Direction classes (cf. [Lienhart96])

The direction class masks used in the prototype are the same as the ones used in the MoCA system. Figure 5.9 shows which 2×2 blocks belong to which class. In the figure, white pixels correspond to text foreground and black to background, since this is the visual appearance of subtitle text, and also corresponds with the chosen implementation. In [Lienhart96], text foreground is represented by black pixels, which means that the horizontal and vertical groups are the same, wheras the diagonal groups are, obviously, different. However, whether the character foreground is represented with ones or zeros in the binary image, makes of course no difference, as long as the system knows which is which. In other words, during binarization/thresholding, the system must know whether the handled text is positive or negative. Since this information in any case is needed for the character extraction algorithm to work, this should not be a problem.

In addition to the 192 features representing direction class frequencies in the sub-images, two special features are extracted to indicate the presence of "accents" [º] and [¨] in the Swedish characters *Å*, *Ä* and *Ö*. The simple, and *very*

ad hoc, algorithm for this feature extraction is to add together the gray level values for some of the middle pixels in the top three rows and subtract the rest. Two different sums are computed, one for each "accent", and if they are above a threshold, the corresponding feature values are set to one; otherwise, they are zero.

The complete feature extraction algorithm, implemented in the MATLAB function `JLFX1913`, is this:

1. Remove the top three pixel rows of the character image and decide by masking if they belong to the "ring" in 'Å' or the "umlaut" in 'Ä' and 'Ö'.
2. Add an empty (black) pixel row on top of the remaining, so that a 17×13 image is the result.
3. Binarize the image through bi-level thresholding at gray level 128.
4. For each possible 2×2 pixel block in the binary image, decide by masking and boolean operations which, if any, direction class (cf. fig. 5.9) it belongs to.
5. Divide the binary image into 8×6 overlapping sub-images of 3×3 pixels.
6. In each sub-image, count the number of pixel blocks for each direction class.
7. Divide by 4 (the maximum number).
8. Create a vector with the "ring-indicator" (0 or 1), the "umlaut-indicator", and the other 8×6×4=192 feature values, computed as above.

The result is a feature vector with 194 elements in the interval [0..1].

5.7 Character Recognition

It was said in the beginning of the preceding sub-section, that character recognition must be performed as a comparison to training examples or templates. That is perhaps an over-simplifiaction. It is true that several of the video text recognition systems presented in literature use the nearest-neighbor algorithm to compare extracted feature vectors to template vectors, with the Euclidian distance or the normalized inner product as a (dis-)similarity measure [Kurakake&al97, Lienhart96, Ohyo&al94]. However, there are other methods for pattern recognition, which can not, at least not intuitively, be described as similarity comparison.

I have used a computer-simulated *artificial neural network*. ANNs have been used in several "conventional" OCR systems. They can be considered combined feature extractors and classifiers, and their function can be described as creating decision boundaries in feature space, where the complexity of the network architecture determines the complexity of the boundaries [Trier&al96]. ANNs can also be described as general function approximators, which are trained by example to give a certain output for a certain (type of) input. The type of network used in the prototype, is a feed-forward perceptron network with one hidden layer.[18] The architecture, which is shown in figure 5.10, has been proven suitable for this type of application.

Since using a hidden layer is like performing a second feature extraction, but with features that are rather incomprehensible to a human, it could perhaps be argued that the hidden layer makes the feature extraction step redundant. This would maybe, although doubtedly, prove to be true, with massive training of the ANN. However, in my own experiments, using the pixel gray levels as network input, instead of the extracted feature vector, gave considerably poorer results.

[18] cf. e.g. [BealeJackson90]]

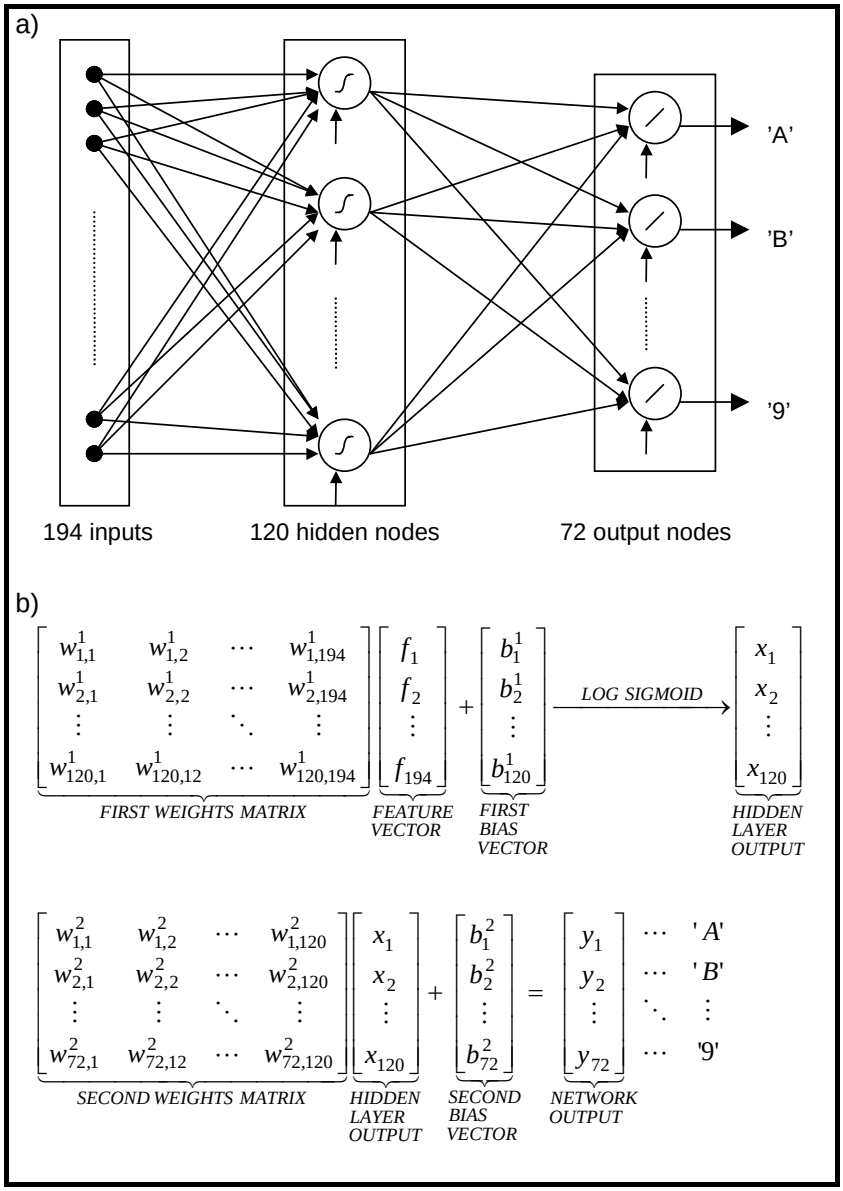

Fig. 5.10. The architecture of the ANN used for character recognition.on: a) symbol form; b) matrix form.

The network has 194 inputs, corresponding to the length of the feature vector, 120 hidden nodes, and 72 output nodes, one for each recognizable character. The number of hidden nodes was empirically determined as a trade-off between network complexity and performance. This evaluation was not very thorough, and 120 is probably not the most ultimate number. The exact number is generally not very important, though.

The ANN is trained to distinguish between 72 characters:

- All uppercase letters in the Swedish alphabet (including 'ÅÄÖ') *except* 'Q' and 'X'
- All lowercase letters *except* 'l' and 'q'
- Period, comma, colon, quote, exclamation, question mark, percent sign, dash and space (. , : " ? % - *space*)
- The digits 1-9

The letters 'Q', 'X', and 'q' are missing due to lack of training examples. The letters 'I' and 'l' are identical in the font used by SVT and can therefore not be distinguished through character recognition. The same is true for 'O' and '0'. The punctuation marks that the network recognizes are the most commonly used in SVT subtitles. Some more

infrequent ones, such as semicolon and parenthesis, have thus been skipped. The slash ('/') has been omitted from the training examples because slashes are inserted by the prototype system to mark text segment boundaries.

The network has been trained by *error backpropagation* [BealeJackson90]. Twelve different sets of characters, all from SVT subtitles, have been used as training examples. However, due to insufficient training material, some character images have had to be used in more than one set. The result of training is a set of weights and biases which are multiplied with, and added to, the inputs of each node in the network. In practice, this is done through matrix algebra, as shown in the equation of figure 5.10b.

The character recognition algorithm, which in major parts follows the standard algorithm for feed-forward perceptron networks, is as follows:

1. Multiply the input feature vector with the weights matrix for the hidden layer and add the corresponding bias-vector.
2. Pass each of the resulting sums through the log-sigmoid function $(1/(1+e^{-x}))$ so that the resulting vector, with 120 elements in the interval [0..1], is output from the hidden layer.
3. Multiply the hidden output vector with the second weights matrix and add the corresponding bias-vector, yielding an output vector with 72 elements.
4. Let the largest element in the output vector point out the recognized character.
5. Append the recognized character to a string.

In figure 5.2, the character recognizer is described as an independent sub-module. However, in the actual MATLAB implementation, the character recognition *functionality* is performed in three lines of program code, which would be rather pointless to put in a separate MATLAB *function*. Therefore, it is implemented directly in a main function JLLR, which calls the four previously described functions and outputs strings of recognized characters. At present, recognized 'I's are replaced by 'l's if any other lower case letter has been recognized earlier in the text segment. This is done only to improve human readability. If the output is to be computer searched, it would probably be better to replace 'l's by 'I's in the target words and leave the output as it is.

The complete extraction and recognition algorithm for a video frame, implemented in the main function JLLR, is thus:

1. Identify and extract those frame sub-blocks that contain text, using the text detection algorithm.
2. Extract text segments within the sub-blocks.
3. For each text segment, extract separate character images.
4. For each character image, compute a feature vector.
5. For each feature vector, perform character recognition, using the algorithm above.
6. If any lower case letter has been recognized earlier within the same text segment, change a recognition output 'I' to 'l' in the output string (for visibility only).
7. Append a '/' to the output string after each text segment, to mark (probable) end-of-line.

5.8 Demo System

With reference to figure 5.2, the developed extraction and recognition algorithms belong in the shaded, bottom, area. The prototype is intended to interact with a larger, surrounding system. However, no such system exists today. Therefore, a demo system has been developed, to demonstrate a possible data-flow, all the way from a VHS cassette to a computer text file.

The demo system comprises all parts of figure 5.2. Its hardware components are a Silicon Graphics O2 workstation with a video digitizer board, connected to a VHS tape deck via an S-video cable[19], and a HP Vectra PC with a 166 MHz Intel Pentium CPU, 32 MB RAM, running MS Windows 95, connected to the O2 over a 10 Mbit/s Ethernet LAN. The major software components are: on the O2, a tcsh-script, written by Tobias Stenälv at Sifo Group IT&T, which uses the SGI software tools *MediaRecorder* and *MediaConverter* to capture a video frame every two seconds, convert it to JPEG format and save it on disk; and on the Vectra, MATLAB 5 with Image Processing Toolbox, and the m-file functions described earlier in this section.

An additional m-file, JLPOLL, works as a main demo program. It repeatedly looks for new JPEG image files in a specified directory, in this case physically on the O2's hard disk, and runs them through the recognition main function JLLR. The output strings are appended to an automatically generated text file. A log file with problem reports is also created. When no new files have arrived during a user-defined time interval, the execution halts. Timing information is added to each string in the text files. A sample output is displayed in figure 5.11.

```
Aktuellt 18, 970909. Neural Network at training level w12fil_8
Processed 1997-10-31, 11:10:20

Time 00:00:32 :  Reseavdragen höjs med 2 krrmil./ Arbetsgivarnas sjuklöneperioO/
Time 00:00:34 :  Reseavdragen höjs med 2 krPmil./ ÅrheBgmarnas sjuklöneperiod/
Time 00:00:38 : -åtemtalU till K dagar. Dessa försiag / är resultatet av en uppgöreUs/
Time 00:00:40 : -åte talU till ? dagar. Dessa försiag / är resultatet av en uppgöreme/
Time 00:00:42 :  mellan regeringen och c / inför hudgetprupositionen./
Time 00:00:44 : -mellan regeringen och c / imör budgetpropositionen./
Time 00:00:48 : Två glada centerparthter presente/ rade uppgörelsen med regeringen-/
Time 00:00:50 : Två glada centerpartmter presente/ rade uppgörelsen med regeringew/
Time 00:00:54 : -innehållande två viktiga centerkrav: / höjda reseavdrag/
Time 00:00:56 : -och kortare sjuklöneperiod / för arbe gwarna./
Time 00:00:58 : -och kortare syukiöneperiod / för arbe gwarna./
Time 00:01:00 :  Vi vill göra det lättare för företagen / att göra nyanställnlngar /
Time 00:01:02 :  Vi vill göra det lättare för företagen / att göra nyanställningar-l/
Time 00:01:04 :  Vi vill göra det lättare för företagen / att gora nyanställningar-3/
Time 00:01:06 :  i. g:c p.eseg  ney a.getöiö,l. e cen./1et är vår primärt viktigaste åtgärd./
Time 00:01:08 : -för att pressa ner arbetslösheten./1et är vår primärt viktigaste åtgärd./
Time 00:01:10 : -för utt pressa ner arbetslösheten./1et är vår primärt viktigaste åtgärd./
Time 00:01:14 :  Det höjda reseavdraget kan betydan/ någon tusenlapp oin aret i förtyänst-n/
Time 00:01:16 : för långpendlarna. Från årsskötet / vil1 c ocli regeringen höja avdraget/
Time 00:01:18 : -för långpendlarna. Från åmskiftet / vill c och regeringen höja avdraget/
Time 00:01:20 : -från 13 kr till 15 kr per mil. Men / gränsen för när avdragen får göras-/
Time 00:01:22 :  g1 di p l,g1 /från 13 kr till 15 kr per mil. Men / gränsen för när avdragen får göras/
Time 00:01:24 : från 13 kr till 15 kr per mil. Men / gränsen för när avdragen far göras-/
Time 00:01:26 : -höh från 6 OOO kr till 7 ODO kr /
Time 00:01:28 : -höjs från 6 OOO kr till Z ODO kr /
Time 00:01:30 :  Protesterna var många den 1.1. i år/ när arbe givarnas sjuklöneperioO/
```

Fig. 5.11. The first part of an output text file, generated by the demo system main program JLPOLL. "Time" is how many hours, minutes and seconds into the program the frame was sent. The output for the first text segment of frame 00:01:06 shows what can happen if extracted characters are placed on the wrong baseline. The other errors are "normal" segmentation and recognition errors. The error rate is somewhat higher than in the total evaluation data set.

[19] In an S-video connection, the luminance and chrominance components are transmitted on separate wires, which leads to higher resolution.

6 Prototype Performance

To allow for performance evaluation, two different broadcasts of SVT's news program *Aktuellt 18* were digitized. The programs, which are 15 minutes long and subtitled in the image[20], were recorded a couple of days apart on analog VHS tape, and later digitized at one frame per two seconds. The frames were stored as JPEG images. One set was used for designing the prototype and training the recognition network. The other was saved and used for evaluation only. During performance tests, the same hardware components as in the demo system (sub-section 5.8) were used.

6.1 Text Detection

The evaluation data set consisted of 432 images, comprising 1728 image blocks. Of these, 379 contained some sort of text. When using the DCT-version of the pre-filter, all 236 blocks containing subtitles were detected, as well as 72 containing other types of text and 13 false blocks. The recall level for subtitle text was thus 100%. Recall was also high for captions and credits, but low for tables and other text. One reason is that the current version only examines the left half of the image. This limitation is very easy to remove, but since at the end, only subtitles are handled anyway, it has not been done. Overall precision was high. The results are listed in table 6.1.

Table 6.1: Pre-filter results

Frame blocks with text of type:	*Actual*	*Detected*	*Recall*
Subtitles	236	236	100%
Captions[1]	36	33	92%
Tables[2]	42	15	36%
Credits	10	10	100%
Other Text[3]	55	14	25%
Total	379	308	81%
Not containing text	(1349)	(13)	(1%)
Total precision			96%
Total number of frames / blocks	432	1728	

Note 1: Of the 3 missed blocks, two contained a single short word and one a caption being faded in (which was detected in the subsequent frame).
Note 2: Of the 27 missed blocks, 17 contained text centered in the image and 10 contained text rows with a single short word in the left half-image.
Note 3: Of the 41 missed blocks, 20 contained exactly the same line of text (which still should have been detected, though) and 12 contained a single word in a weather map. The remaining nine contained scene text.

Table 6.1. Results of applying the Pre-filter (DCT) Module to the evaluation set. 432 frames (comprising 1728 blocks) were examined. All 236 blocks containing subtitles passed the filter, as well as 72 containing other types of text and 13 false detections.

Results from using the alternative, "uncompressed", version of the pre-filter have not been quantified. A qualitative study indicated the same level of recall and a slightly lower level of precision, compared to the DCT-version. In both cases, the text detector seems to perform perfectly well.

[20] Beginning in October 1997, *Aktuellt 18* is now teletext-subtitled instead.

6.2 Text Line Extraction

It is a bit harder to evaluate the performance of the text segment extractor. In figure 5.6, the top caption has been extracted as two separate text segments. Is this correct or not? If it is, had it been wrong to extract the caption as one segment? Since it makes very little practical difference if a text row is extracted as one or more segments, as long as no words are split in halves, the number of correctly identified and extracted text *lines* has been used instead. At this step, the text category "other" has been ignored, since such text often is not horizontally aligned. 321 image blocks passed the pre-filter. The results of text line extraction within them are listed in table 6.2. Again, recall was high, if not quite 100%, for subtitles, but significantly lower for other text types. This depends partly on the fact that here, as well as in the text detector, only the left image half is used for detection. Another reason is that the thresholds used were trimmed for subtitle extraction. Precision was very high, almost 100%.

Table 6.2: Text line extraction results

Text lines of type:	*Actual*	*Extracted*	*Recall*
Subtitles[1]	460	452	98%
Captions[2]	58	29	50%
Tables[2]	19	9	47%
Credits[2]	22	19	86%
Total	559	509	91%
Not containing text of above types[3]		(7)	
Total precision			99%
Total number of blocks	321		

Note 1: Six of the eight missed text lines were very short (three instances each of "*-Ja.*" and "*att göra...*"). The remaining two were inexplicably missed in one frame and correctly extracted from the two following.
Note 2: The low recall figures for the other text types are mainly due to the fact that the system was trimmed for subtitle extraction and recognition. In earlier experiments, with other thresholds, the figures were significantly higher. The comparatively high recall value for credits is coincidental.
Note 3: Of the seven extracted "noise"-segments, all but one contained "other" text or fragments thereof.

Table 6.2. Results of applying the Text Segment Extraction Module to the 321 image blocks wich passed the Pre-filter. 452 of the 460 subtitle text lines were correctly extracted, as well as some text lines of other types and a few false detections.

6.3 Character Segmentation and Recognition

Finally, the performances of the character segment extractor, the feature extractor and the character recognizer have been evaluated. Since they are closely interdependant, they are presented together. Another reason for this is that certain standard performance metrics, such as the *total character error rate*, include both segmentation and recognition errors. In fact, some types of segmentation errors reported here, i.e. "misses" and "additions" have their origin back in the text segment extractor.

In this evaluation, only subtitle text has been regarded, and only a subset of the evaluation data. Manual evaluation is a time-consuming and pain-staking procedure, so only the first 100 subtitle rows have ben used.[21] These still contained 3178 characters, though. Segmentation and recognition results are listed in table 6.3. Extra spaces between words have been ignored, as well as isolated garbage characters that do not interfere with real words, since they do not affect the ultimate performance of the prototype. This leniency of course improves the reported results, albeit not in a significant amount. The overall performance must still be considered good, with both recall and precision at

[21] A not-as-thorough examination of the remaining 352 subtitle text rows indicates that the performance figures are as good, if not better, for the entire data set, as for the first 100 rows.

96% and a total character error rate of 5%. It would, however, be misguiding to compare these results with those of other video text recognition systems. This prototype only handles one, very specific, type of text.

Table 6.3: Character segmentation and recognition results

Actual characters	3178	
		% of actual
Segmentation		
Correctly segmented	3085	*97,1%*
Split 1 to 2	2	*0,1%*
Merged 2 to 1	88	*2,8%*
Missed completely (at extremes)	3	*0,1%*
Added "noise" (at extremes)	24	*0,8%*
Total output (CorSeg+2*Split+… ... +Merged/2+Added)	3157	*99,3%*
Recognition		
Correctly recognized	3036	*95,5%*
Substitutions due to segm.errors	44	*1,4%*
"True" substitutions	51	*1,6%*

Actual characters	3178	
		% of actual
Overall Performance		
Insertions	26	*0,8%*
Deletions	47	*1,5%*
Substitutions	95	*3,0%*
Total output (CorRec+Ins+Subs)	3157	*99,3%*
Recall (CorRec/Actual)	96%	
Precision (CorRec/TotalOut)	96%	
Character Error Rate ((Ins+Del+Sub)/Actual)	5%	

Table 6.3. Character segmentation and recognition results for the first 100 lines of subtitle text in the evaluation data set. Of 3178 characters, 3036 (95.5%) were correctly output.The segmentation algorithm accounts for missing 2.9% of the characters and the recognition algorithm for missing an additional 1.6%. Of the segmentation errors, "Missed" and "Added" errors were introduced in the text segment extraction module, and the others in the character segmentation module.

6.4 Processing time

The total handling time of a video frame ranges from approximately one second, for images without text, to almost five seconds for images with several text rows, including two full subtitle rows. One second is the time it takes MATLAB to read a color jpeg-image from disk, decompress it, sub-sample it and convert it to gray-scale. Text detection takes less than 0.01 second. Text segment extraction takes 0.5-1.0 second, depending on the size of the image block and the number of text rows. Character segmentation takes up to a couple of seconds, depending on the number of characters and the "quality" of the background. Feature extraction and character recognition together take a little more than 0.01 second per character.

Whether these figures are "good" is hard to say. Processing a typical frame with subtitles only, takes a little more than three seconds. In the demo system, frames are captured every two seconds, which is more than sufficient to cover all text in the video – an average subtitle is recognized twice or thrice. This means that, depending on the amount of text, the text extraction and recognition system's handling time for a complete TV program is 50-200% of the program's length. I consider this figure reasonably acceptable.

6.5 Search results

Speed is one of the most relevant performance issues in the general context of a content-based TV filter. Another is the usability of recognition results for searching/filtering. Simply counting the output words that were completly correctly recognized would not be very interesting, as it has been said before that a search engine for this type of material should allow for some character errors (cf. sub-section 3.7). Furthermore, the effect of recognizing the

same text more than once should be studied. Therefore a test was conducted with a search engine with a variable character error tolerance, searching for typically "relevant" words in the output of the demo system.

Members of Observer Sweden's editiorial department, who professionally create summaries of press material for Observer's clients, were given a perfect transcript of the subtitle text in the evaluation program. They marked approximately 80 different words as the most significant for the program's content. A simple text search engine, written by myself in MATLAB code, was then used to search for the words in the demo system's output text file. The search engine was implemented to allow a variable frequency of insertions, deletions and substitutions. It was not able to handle words that were split over two rows, so a few such words were excluded from the evaluation, leaving 125 instances of 79 different words. In this test, the benefits of handling the same subtitle (usually) twice, were employed. Thus, a word appearance in a subtitle was only counted as one instance, even if the same subtitle appeared more than once. Consequently, for recall evaluation, it was sufficient if a word was correctly identified in one such subtitle instance. On the other hand, multiple instances of the same subtitle do not necessarily improve precision, and *all* false matches were counted.

Two tests were conducted, the results of which are listed in table 6.4. In the first test, where no character errors were tolerated, recall was approximately 90% and precision close to 100%. In the second test, one character error was allowed per new set of ten characters in the target words.[22] Recall rose to around 95%, while precision dropped to circa 90%.

Table 6.4: Search results

Error level 0	Actual	Found	False	(False, incl. inflections)
Words	79	70	0	(0)
Instances	125	112	4	(4)
Recall & Precision, Words:		*89%*	*100%*	
Recall & Precision, Instances:		*90%*	*97%*	
Error level 10	**Actual**	**Found**	**False**	**(False, incl. inflections)**
Words	79	76	8	(14)
Instances	125	118	14	(25)
Recall & Precision, Words:		*96%*	*90%*	
Recall & Precision, Instances:		*94%*	*89%*	

Table 6.4. Results from searching for words in the prototype output, using a search engine with variable character error tolerance.

In a third test, whose results were not quantified exactly, one error was allowed per five letters. The effect was an additional decrease in precision, but not any significant raise in recall. This was mainly due to the "double curse" of segmentation errors. Since almost all splits and merges result in two errors, one insertion/deletion and one substitution, shorter words with segmentation errors were still not accepted as search matches. This could be overcome by allowing even more character errors, but probably at the expense of an unacceptably low level of precision. Still, 95% recall with 90% precision is arguably not a result to be ashamed of.

[22] I.e., in words with up to ten letters, one error was allowed; in words with 11-20 letters, two errors, etc.

7 Conclusions and Further Work

The purpose of the prototype was to prove that realtime or near-realtime video OCR can be done. I feel that the performance figures presented in the preceding section give evidence to that proof. It is certainly true that limiting the character recognition material to a single type of text in one size and font, is no small limitation. It should be remembered though, that the presented prototype is no more than that – a prototype created more or less from scratch in approximately ten weeks. There are several things that can be done to transform the prototype into a "real" system. Some suggested improvements are presented in table 7.1.

Table 7.1: Suggested Improvements

Module/Function	Short Term Improvements	Long Term Improvements
Pre-filter (JLDCTX)	• C-version of JLDCTX, compiled for speed. Either as a stand-alone process which reads JPEG-files from one place and saves MAT-files at another, or as a MEX-file, called from JLPOLL just like today.	• Stand-alone executable which "listens" to a MPEG bitstream and sends matrices to MATLAB over network (eg. via UDP), alternatively saves MAT-files on disk. (Requires a video capture card capable of real-time MPEG encoding)
Text Segment Extractor (JLTEX)	• Ability to handle centered or right-aligned text. • More sophisticated thresholding, to improve recall of eg. captions • Better handling of different character sizes (x-heights)	• Compiled version, for speed • Ability to discriminate between subtitles and other text
Character Segmenter (JLCHSEG)	• General tuning of thresholds (to lessen the number of Merge-errors, probably at the expense of some more Split-errors) • Ability to discriminate between and handle text of different height • Ability to handle oblique and underlined characters • Improved handling of text without solid background • Ability to handle positive (eg. black on white) text	• Version based on region labelling rather than column max values • Compiled version, for speed
Character Feature Extractor (JLFX)	• More general handling of Å,Ä,Ö • Ability to handle other sizes than 19×13	• Dynamic thresholding
Neural Network	• Training with subtitle-characters of other fonts • Training with characters from other text types, ie different fonts and sizes	• Optimizing training algorithm for speed, possibly through compiling
General	• Optimization for MATLAB 5 (improves speed)	• System output via network (UDP) to separate parsing module, rather than to file on disk • GUI for monitoring and tuning • Complete MATLAB-independence (?)

Table 7.1: Suggested improvements of the Text Extraction and Recognition Module (System).

Some of the "short tem" improvements are trivial, such as adding the ability to handle text which is not left-aligned. Others, like a c-version of the pre-filter, require more work. Most of the suggestions are either for making the system more general or for making it faster. More general handling is of necessity. A real system should be able to handle at least all sorts of subtitles and captions, and as much as possible of tables and other text. Ad hoc solutions must be replaced by robust methods. Higher speed is not necessarily a top priority, though. The prototype is reasonably fast and if shorter processing times are needed, maybe it is sufficient simply to run the system on a faster computer. For instance, running the prototype on a 200MHz Pentium Pro, instead of on a Pentium166, gave approximately 30% lower overall handling time.

An interesting question is how the system should interact with the user and other parts of a larger system. Complete MATLAB-independence is a related question, but not necessarily a solution. A significant contribution to the

prototype's speed comes from MATLAB's efficient algorithms for matrix manipulation. There are ways in which MATLAB can be used as a "computational engine" [ML97], called from other applications. This is in any case an area which should be investigated further.

The overall conclusions of my work with the prototype must be that it is indeed possible to perform video text extraction and recognition on a personal computer, that MATLAB is a suitable environment for such computations, and that with some, not too prohibiting, modifications and improvements, the presented prototype can be transformed into a real, working, system.

As a more general conclusion of this thesis project, based on the findings presented in sections 2-4 and the prototype results, I believe that it should be possible to create a multi-modal content-based TV filter, satisfying the demands described at the beginning of section 3. A possible design of such a system has been presented in sub-section 3.7, and although no such system is commercially available today, some of the components are. There are extensive research efforts being made in the area, and it does not seem unreasonable to think that for example the Informedia project or BNN will lead to commercial systems not entirely different from the theoretical system presented in this report.

In any case, this field of research is a fascinating one. The ever-increasing flow of multimedia information, in forms of analog and digital television, digital video on the Internet, and who knows what in the future, must somehow be tamed. Techniques for the taming are emerging.

8 Development 1998-2005

As mentioned in the preface, this thesis project was carried out during the second half of 1997, and the previous sections of the report were in all essence written in late 1997 and early 1998. This section aims to briefly describe some of the development within the subject area since then and up to mid-2005, when the report is ultimately presented.

8.1 The Problem

The company at and for which the project was carried out, Observer Media Intelligence, has undergone some changes since 1997. Then a division of the Swedish Sifo Group, Observer became a separate, listed company in 2000. In the following years, Observer grew substantially through aggressive acquisition of the leading media monitoring companies in the UK, Ireland, Portugal, Canada and the United States. With c. 2,500 employees in 13 countries, the Observer group is now by far the world's largest company within media based communication management and business intelligence [Observer05]. At the core of the company's business is still the monitoring of printed, electronic (more so than in 1997, naturally) and broadcast media. The process of TV monitoring is essentially the same as in 1997, at least in Scandinavia and Germany. Programs are to a greater extent recorded, stored and/or delivered digitally – rather than on analog video tapes – but the content analysis, i.e. the matching of certain news stories with individual clients' information needs, is still performed entirely manually, by employees listening through the recordings. Whereas media monitoring in general has become significantly faster and cheaper through the application of automatic information filtering technology to electronically available media (e.g. Internet news sites) and digitized printed media, the analog monitoring of broadcast media is still very time- and resource-consuming. Thus the need for systems and tools to automate parts of this process is as great today as at the outset of this project, if not greater.

8.2 Commercial Applications

As concluded in section 7, there were no commercially available applications tailored to Observer's needs in 1997. The only COTS system for content-based TV filtering, the *Televitesse* system described in section 2.5.1, depended entirely on North American closed captions (and went off the market already in 1998). Today the situation is significantly different. Companies within the Observer group, in particular in English-speaking countries, have had a chance to evaluate several different commercial, multi-modal systems over the last couple of years.

The most extensive of these is the "VS News Monitoring" system from *Virage* (cf. 2.5.3). Virage was acquired in 2003 by *Autonomy*, one of the world's leading providers of search engine and knowledge management technology, who also acquired the share majority in the speech recognition company *SoftSound* (cf. 3.5.2). Components from the three companies (and others) have been combined into a modular solution which captures, stores and analyzes TV programs, allowing keyword based searching and real-time filtering ("alerts"). The system extracts searchable content metadata firstly through speech-to-text transcription and from closed captions / teletext subtitles if available. Speaker recognition and other types of audio analysis are used primarily for story segmentation, together with video analysis technology such as shot boundary detection, face recognition, logo recognition and video OCR, although the output of some of these can be used for filtering/retrieval as well. The system also includes NLP modules for text analysis of transcripts, for named-entity extraction (people, places, etc.), summarization and further improved story segmentation [Virage05, Autonomy05, SoftSound05]. Several of these functionalities are made available as optional, third-party plug-ins, e.g. the "ConTEXTract" video OCR module which is licensed from SRI (who have also been developing speech recognition technology, cf. 3.5.2) and the "FaceIT" face recognition module from the US company Identix [SRI05, Identix05].

Another commercial system targeted at end-users, as well as monitoring companies such as Observer, is the "NewsScanner" from the Israeli company *Idioma*. The system analyzes the video stream of captured TV broadcasts for scene changes and stores the timestamps of these in an index, together with closed captions or teletext subtitles. An optional word spotting module, developed by Philips, offers a possibility to find topics of interest in broadcasts without subtitles. When words matching such (pre-defined) topics are detected in the audio stream, the corresponding time stamps are logged in the index. The index can be then be used to browse identified "hits" either topic by topic, or program by program, after which the user can edit, annotate and distribute stories of interest [Idioma05].

Two systems using speech-to-text transcription (rather than word spotting) for media monitoring purposes are "RadioEars" from the US company *TVEyes* and the "Media Mining Indexer" and "Media Mining Server" from Austrian *SAIL Labs*. RadioEars is designed for generating real-time alerts, i.e. filtering, whereas the Media Mining Server is primarily designed for search and retrieval. However, the Media Mining Indexer generates a searchable transcript in real-time, which can be input into a filtering/alert application as well. In addition to speech recognition, SAIL's Indexer offers speaker recognition, speaker change detection and NLP technology for named entity extraction and story segmentation / topic detection. The user interface of the Media Mining Server contains a keyword translation feature, allowing the user to enter search words in one language and retrieve stories containing the corresponding words in other languages [TVEyes05, SAIL05].

Although the SAIL system allows multiple language queries, a common aspect of all four systems – and a common drawback from a Scandinavian perspective – is that they are in part restricted in the languages they can index. Some of the functionality is not language dependent, e.g. teletext capture, video-based segmentation, and speaker identification, where available. However, some key functionality is only available for English and a few more (major) languages. E.g. in the Virage system, image text extraction and recognition is offered in English and Spanish only, and speech transcription in Italian, German and French in addition. The SAIL system currently transcribes the same languages, plus Arabic. None of the systems provides speech recognition for the Scandinavian languages.

8.3 Research Projects

Language is an area of concern for several projects, yet to result in commercial applications. TVEyes and SAIL Labs currently collaborate in a EU-funded project called *REVEAL THIS*, concerning storage, categorization and retrieval of multimedia and multi-lingual digital content across different sources, including television. To promote European research and development within "information society technology", with the expressed aim of "increasing industrial competitiveness and the quality of life for European citizens in a global information society", the European Commission sponsors a wide variety of such "IST" projects, run in collaboration between academic institutions, commercial IT developers and user organizations from several different European countries [IST05]. In REVEAL THIS, TVEyes and SAIL are joined by Xerox and universities in Greece, Belgium and the UK. The project, started in November 2004, is primarily targeted at consumer end-users, whereas another project, *PENG* ("Personalized News Content Programming"), is aimed at professional users, e.g. journalists, providing them with an "interactive and personalized tool for multimedia news gathering and delivery". PENG participants include universities in France, Switzerland and the UK, a Spanish IT development company and the Swiss (Italian) Broadcasting Corporation. Both projects are due for completion in 2007 and have not yet publicized any tangible results [REVEAL05, PENG05].

A few years ago, another EU/IST project was carried out, specifically aimed at media monitoring and similar "selective dissemination" of multimedia information. The *ALERT* project, in which Observer participated through its German subsidiary, focused on broadcast speech recognition and automatic topic detection in languages for which there had been very little previous development – French, German and Portuguese. For each of the three languages, the project produced a demo system, alerting users to news stories of interest, through the use of audio and video analysis for segmentation, speaker-independent continuous speech recognition for automatic

transcription, and statistical methods for topic segmentation and detection. Statistical methods (rather than linguistic) were used because they are more tolerant to the types of errors found in automatically generated transcripts. In the German system, a word error rate (cf. 3.5.2) of 17% was achieved, which was considered a very good performance compared to other German broadcast transcribers. The same system's topic segmentation algorithm identified c. 90% of the topic boundaries, with 65% precision. The project consortium included universities and user organizations representing the three languages – the two other users were Observer's French competitor Secodip and RTP, the Portuguese national broadcasting corporation – as well as two software companies, one French and one Portuguese, acting as system integrators. When the project was concluded in 2003, it had met several of its original goals, but not the one of creating a common, multi-lingual platform. Rather, three separate systems were created, one per language, based on the slightly different needs of the three user organizations [Iurgel&al02, Neto&al03, Gauvain&al02, ALERT03]. This division into three tracks, in combination with the lack of a German system integrator in the project consortium, was perhaps the main reason that from Observer's perspective, the project, however theoretically interesting, did not yield any concrete applications. Specifically, it did not lead to any change in the way Observer monitors TV broadcasts in Germany.

What, then, became of the projects deemed most interesting in 1997? The *MoCA* project (cf. 2.5.2) continued for a couple of years, with particular emphasis on face and object recognition, but it does not seem to have spun off any commercial applications. The project web site ([MoCA05]) is still online, but it has not been updated since 2001. By contrast, the *Broadcast News Navigator* developed at MITRE (cf. 2.5.3) has been subject to more recent development. In particular, the system has been extended with "local context analysis" technology for query expansion and interactive user feedback. This allows the user to retrieve more stories of interest through the identification of relevant named entities – other than those included in the original query – in found stories, and using these for query expansion. The system has also been "personalized" by allowing users to set up profiles of topic interests, etc., which can be used for ongoing monitoring and generating alerts. The BNN has been used internally at MITRE for research and within the US military for daily information monitoring. The system can still only handle US news broadcasts (with closed captions) and the developers acknowledge that the anticipated future research into multilingual content will pose challenges not only in language handling, but also in dealing with different information structures and formats [Maybury&al04].

Monitoring of foreign news broadcasts for military intelligence purposes, is the subject of the most recent sub-project of the extensive *Informedia* project at CMU (cf. 2.5.3). A new system for this purpose, called "ENVIE" ("Extensible News Video Information Exploitation"), dealing specifically with Mandarin and German broadcasts, will be developed, building on existing Informedia technology. Since 1997, the Informedia project has spawned several sub-projects for a wide variety of applications. Within the core system, speed and accuracy of the underlying information extraction methods have been improved. New functionality has been added for named entity extraction, for video OCR and for speaker name and face identification. Much effort has been put into new user interfaces supporting multi-modal queries and dynamically generated multimedia summaries [Informedia05].

To promote development and facilitate benchmarking of systems like Informedia, the US National Institute of Standards and Technology in 2001 extended the "Text REtrieval Conference" ("TREC") program with a video retrieval track, known as *TRECVID*. In the TRECVID program, research groups can compare their results on a common set of video documents and specific topics, in a process similar to the ARPA Hub4 Evaluation of speech recognition, described in section 3.5.2. In the TRECVID 2003 evaluation, which was the first to deal specifically with news broadcasts, 24 companies and academic institutions participated with one or more systems each. The Informedia group achieved the highest overall performance in the evaluation, with a system using a combination of speech transcription, closed captions, video OCR and a dynamically weighted vector of various low-level audio and video features [HauptmannChristel04]. A rather detailed description of the components of Informedia's TRECVID 2003 system is given in [Hauptmann&al03]. In 2004, the number of participating project groups had increased to 33, including IBM Research, the Imperial College in London and the Finnish Media Team Oulu. An overview of their approaches and the results can be found in [Kraaij&al04].

8.4 Technology Development

The results of the research projects and development of the commercial applications described in this section, seem to validate the multi-modal, modular system design of a theoretical system, discussed in section 3. In particular, the overall principles apparently hold true, although, of course, some of the detail design choices do not reflect the underlying technical development since 1997. Regarding this development, three very general, and perhaps trivial, observations can be made:

1. *The processing and storage capacity of "ordinary" computer equipment has increased significantly.* This is blatantly obvious – the PC on which this report is finalized has 20 times higher processor speed and 30 times more memory than the one on which the prototype was developed – but it means e.g. that algorithms previously discarded because they were too time-consuming to allow anything resembling real-time processing, or required prohibitively expensive equipment, are now practical.

2. *Algorithms have improved and new standards have emerged.* This is of course true within most subject areas, but one of the most obvious is compression, where e.g. the "MPEG audio layer 3" format, briefly mentioned in 3.2.5, has enjoyed quite some success under the shorter name "mp3". Another area is data interchange, where *XML* today would be a more natural format choice in the system outlined in section 3.7, than SGML (although XML is in fact a subset of SGML, so…) [XML05].

3. *Functionality previously only available in research systems is now available in commercial applications and modules.* This is in no small part a direct and indirect consequence of 1 and 2, of course.

A very recent and extensive survey of various techniques associated with multi-modal video indexing is presented in [SnoekWorring05]. The authors propose a theoretical video indexing framework, in which the *layout* and *content* of each of the three modalities – visual, auditory and textual – is constructed from (in the case of layout) "fundamental units" (e.g. video frames), transitions and special effects, and in which the layout and content in their turn map to a hierarchical *semantic index*, ranging from genre via "logical units" to "named events". With reference to this framework, the authors review various methods for segmenting, extracting and integrating layout and content information, among those methods described in sections 3-4 of this report, and in later literature. They particularly emphasize the benefits of a multimodal approach, i.e. using information in all modalities and combining it for even better results.

As for analyzing the *visual* modality in broadcasts, "observation 3" above holds particularly true. Whereas no such commercial systems seemed to exist in 1997, there are now several applications for video OCR, face detection, etc., e.g. the SRI and Identix systems that are available both as stand-alone applications and plug-ins to the Virage system mentioned in 8.2. Another company supplying such technology is *Internet Security Systems*, whose "Proventia" product suite is capable of detecting and recognizing logotypes (in addition to text and faces) in video and image collections [ISS05]. All of the systems described in 8.2 contain some form of shot detection/segmentation mechanism.

Exactly how these commercial applications work is naturally not very publicized, but it is reasonable to assume that they algorithm-wise are not substantially different from the approaches described in sections 3-4 and later literature. A fairly recent survey of low-level image analysis and pattern recognition methods can be found in [Antani&al02]. This paper also contains a comparative study of the effectiveness of various *cut* detection methods, where the authors conclude that Yeo and Liu's method (cf. 4.2) is the most successful on compressed data, but still not as efficient as color histogram comparisons in the uncompressed domain. This result is corroborated in [Lienhart01], which is an exemplarily comprehensible (rather than comprehensive) survey specifically of shot transition detection methods. However, Lienhart argues that the commonplace distinction between "compressed" and "uncompressed" methods (as e.g. in section 4 of this report) is not very relevant, as the core algorithm – color comparison, motion

tracking, etc. – is typically the same in both cases. This is especially true for Yeo and Liu's "DC images", which essentially are downsampled "regular" images. Lienhart concludes that intensity variance methods (as in most of the approaches described in 4.1 and 4.3) are the most efficient for detecting *fades*, whereas the comparatively more difficult task of *dissolve* detection is best solved with a new neural network-based approach.

The effects of "observation 1" are particularly apparent within *audio* analysis, specifically *speech recognition*. Applications which in 1997 required high-end UNIX workstations, will today run, with better performance, on Windows- or LINUX-based PCs. Two of the state-of-the-art SR systems described in section 3.5.2 are now available for free download under an open source license. CMU's *Sphinx* is currently in version 4 and completely rewritten in the Java programming language. The latest performance figures for the "Hub 4" evaluation data indicate a word error rate of 18.8%, compared to 27.5% in the 1996 evaluation [Sphinx05]. The *HTK* toolkit, which in 1997 was marketed by Entropic, is now free to download from Cambridge University, after Microsoft acquired Entropic in 1999. The HTK is a library of C code and can be used for other Hidden Markov Model applications, such as character recognition and various classification tasks, as well as speech recognition. The latest version, 3.3, was released in July 2005 [HTK05].

Of the other two commercial vendors described in 3.5.2, *SoftSound* was effectively acquired by Autonomy, as mentioned in 8.2, above, leaving *Nuance* as the biggest independent speech recognition company. This position is further strengthened through Nuance's merger with ScanSoft, announced in May 2005 [Nuance05].

Nuance's is the only system among these which comes with ready-made support for the Swedish language, but for call-center applications, which is rather different than broadcast transcription. The European systems for broadcast speech transcription, e.g. SAIL Labs' system and those developed or used within the ALERT project (cf. 8.3), do not handle Scandinavian languages either. In Sweden, academic research within speech recognition is concentrated to the Centre for Speech Technology at the Royal Institute of Technology. Based on the quarterly and annual reports published on their web site [CTT05], broadcast speech transcription does not seem to be the subject of any current or previous projects.

8.5 Video OCR

The apparent lack of broadcast SR technology for the Scandinavian languages and the specific local complexity of having different spoken languages mixed in the same news program, emphasize the virtues of a multi-modal solution, incorporating the detection, extraction and recognition of text appearing in the video images. Video OCR has been the subject of intensified research and development in the years since 1997, with research efforts focusing on various types of text for different applications. The two fundamental types of text are *scene text* occurring "naturally" in the images, e.g. signs in the background or car license plates, and *overlay text* added "artificially" in the video editing process, often referred to as "captions". Of the five types of text discussed in section 5.1, the first four belong to the overlay category. The *purpose* of video OCR may be to extract semantic content for *indexing and retrieval*, but it can also be to perform specific *image processing* tasks, e.g. to optimally encode high resolution text objects within a low bit-rate compression standard, such as MPEG-4 (cf. 3.2.5), or to remove overlaid text from the image [Lienhart03].

This thesis project deals with overlay text recognition for retrieval purposes. The same application has been considered in two fairly recent reviews, [Crandall&al03] and [Lienhart03], in which the authors present various approaches in recent literature, comparing them both in terms of algorithms and performance to their own respective approaches. From these reviews, two trends can be discerned in the development since 1997.

One is the use of standard (commercial) OCR software for the final recognition step in the process, as opposed to the development of video-specific recognition software – as e.g. in the early approaches in the MoCA project and, of course, within this thesis project. Standard OCR software, developed for recognizing text in scanned paper

documents, will run into serious problems if applied directly to video images. These problems include low resolution, non-uniform color of text and noisy, multi-color backgrounds. Therefore, current video OCR approaches employ not only sophisticated algorithms for *binarization*, i.e. the division of pixels into text and background, but also various image enhancement methods, e.g. interpolation.

Another trend is the increased attention given to the temporal dimension of video. In particular, algorithms have been developed for *tracking* text objects through video frames, from their first appearance to their disappearance (referred to as a *text event*), during which time they possibly move and change their size and/or orientation. Text tracking is essential in image processing applications, but it is also beneficial in a retrieval context, as it ensures that every text object is indexed only once, while using the temporal redundancy to increase OCR recall and precision. Apart from the relative effects on search performance of having the same subtitle typically recognized twice or thrice, as discussed in 6.5, this project has not considered the temporal dimension within video OCR.

These trends aside, there seem to be no fundamental changes in the research directions taken post-1997. The basic process steps are the same – *detection* of text, *extraction* (or localization) of text regions, *segmentation* (or binarization) of text characters, and *recognition* of these. As in 1997, there are two major groups of algorithms identified in literature – *connected component*-based and *texture*-based (referred to as "spatial-variance methods" in section 3.4.2) – as well as hybrid methods combining the two. The approach taken in this thesis project (and in most of the systems reviewed by Crandall et al and Lienhart) is mainly texture-oriented.

Texture-based text detection involves extracting some *feature metrics* from an image region and applying some *decision method* to determine whether the region contains text or not. Decision methods in recent literature range from simple thresholds (e.g. if the sum of the extracted features is higher than a certain value, text is considered detected) to sophisticated machine learning classifiers, such as decision trees, neural networks and support vector machines. Some of the suggested features are local color variance, local edge strength, edge density, edge angle distribution and edge angle symmetry.

The algorithm used in this project is a specific case of edge strength detection in the compressed domain, in which a subset of the DCT coefficients in MPEG I-frames, corresponding to "predominantly horizontal" frequencies, are used as text detection features (cf. section 5.3). The idea of using a DCT subset apparently came independently to different people and research groups in the late 1990-s. The approach is often attributed to [Zhong&al00]. However, the same idea, although for JPEG images, was proposed by researchers at Stanford University's Information Systems Lab already in 1995 [Perlmutter&al96].

The algorithms based on this idea, in particular the coefficient subsets used, vary between the applications. Perlmutter, Chaddha et al used 18 DCT coefficients and proposed an empirical method of choosing them. They computed the average absolute values of each of the 64 coefficients for two reference sets of 8×8 image blocks – one containing text and one not – and used statistical methods to determine which coefficients had the highest "discriminatory power". The resulting set of coefficients is illustrated in fig. 8.1a). Later research at the same institute indicated that detection errors could be decreased by a quarter through the use of coefficient *likelihood* measures, rather than the absolute sum of the subset coefficients, as a feature metric. Based on observed statistical distributions of all (quantized) DCT coefficient values, in text and non-text image blocks, two conditional probabilities for each possible value, given text and non-text respectively, were computed. The absolute difference between these probabilities, for the observed values, was summed over the 18 coefficients as a feature metric [Perlmutter&al96, Kalman&al01].

The same empirical approach for finding the optimal DCT coefficient subset, but with a slightly different algorithm, was described in [Crandall&al03] and used in the authors' own system. Their use of the absolute differences between "text" and "non-text" coefficient sets as discriminator favored the generally high-energy, low-frequency coefficients, as can be seen in fig. 8.1 b). This may be one reason why the algorithm reportedly performs significantly better on

larger text. The sum of this subset is used for detecting horizontal text, whereas the authors use the same set on the transpose of the DCT coefficient matrix for detecting vertical text. However, since the chosen subset, with one exception, is symmetrical along the diagonal, it seems questionable whether the difference between the "horizontal" and "vertical" sums is very big.

The subset presented in [Zhong&al00] seems to be chosen through a combination of deductive reasoning and experimental tuning. The algorithm starts detecting text using only a small number of strictly horizontal frequency coefficients and later discards some of the false positives by examining the strictly vertical coefficients. This is illustrated in fig. 8.1 c). For comparison, the coefficient subset used within this project is shown in fig. 8.1 d).

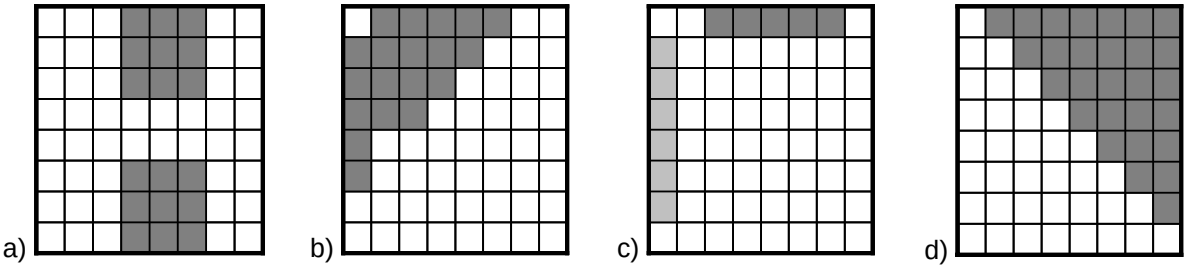

Fig. 8.1. The DCT coefficient subsets used for text detection: a) by Perlmutter, Chaddha et al (according to [Kalman&al01]; b) by Crandall et al; c) by Zhong et al (where the dark gray horizontal coefficients are used first and the light gray vertical coefficients later in a multi-step approach); and d) within this project.

Zhong et al perform morphological operations with a 1×3 structural element, in order to connect text blocks into cohesive horizontal text regions, while discarding singletons. Several of the reviewed texture-based approaches involve morphological operations on *edge* images (as discussed in section 5.4), based on assumptions on edge density and orientations, in order to more finely extract text regions. Connected-component analysis, and similar color based approaches, seem to be the most popular for character segmentation. Different measures have been taken to increase robustness, in particular size-independence, where performing the same algorithm on a series of subsequently sub-sampled input images and then weighing the results together, is a popular approach. Nevertheless, it is worth noting that in most of the described systems, as they proceed further through the video OCR process, more and more heuristic constraints regarding e.g. minimum and maximum size of text, minimum length of text rows, "reasonable" aspect ratios of text characters, orientation, etc., are used. In this respect, they do not differ from the approach taken in this project.

Performance of the reviewed systems varies with their generality as well as with the algorithms used. As expected, systems tuned for specific types of text perform significantly better on these than on other types, with more general systems' performance "in-between". For text detection and extraction, Lienhart has collated reported results for several fairly recent approaches. Recall is consistently reported within the range of 92-99%, with precision (where reported at all) between 61% and 95%. In this comparison, Zhong et al's system was the fastest and best in terms of recall, but among the poorest with regard to precision. The reported results are qualitatively consistent with and quantitatively – roughly – comparable to the results presented in tables 6.1 and 6.2.

However, one should not draw too far-reaching conclusions from reported results, as they all differ in evaluation procedures and evaluation data sets. To establish a common, independent benchmark, the 2003 International Conference on Document Analysis and Recognition (ICDAR) included a "robust reading" competition, similar to the ARPA Hub 4 and TRECVID evaluations discussed previously in this report. The competition was sub-divided into the tasks of text locating, character recognition and word recognition. However, it was only the first task that drew actual contestants, and only five of them [Lucas&al03]. As the evaluation data set contained scene text, the competition results are neither very comparable to the text detection results reported in [Lienhart03], nor very

relevant *per se* in the context of this report. However, it could be useful to monitor the future development of this contest, and in particular the state-of-the-art systems it is intended to review and compare.

For text segmentation and recognition performance, Lienhart concludes that OCR accuracy is the only remotely comparable metric. Again, the presented results are those reported by each author respectively, using different evaluation sets. When comparing the typical character error rates for recent systems, reported between 12% and 30%, to the 5% presented in table 6.3, one should bear in mind that the recognition module developed within this project was tightly tuned to – and evaluated for – a specific type of text.

Ultimately, in a content-based retrieval or filtering context, it is in the search engine performance that the relevant effectiveness of video OCR is measured. Several approaches have been taken to boosting retrieval recall (often at the expense of precision), through the use of temporal redundancy – i.e. the same text appearing several times in the index, as the result of different OCR runs – and/or fault-tolerant searching. E.g. the Informedia system includes an "approximate string matching" technique, which allows a certain number of insertions, deletions and substitutions to transform a candidate from the OCR output into a query word, with this number depending on the length of the query words [Hauptmann&al03]. A simple variation on this approach was taken in the search results evaluation within this thesis project, described in section 6.5.

8.6 Conclusions

In summary, the problem of automatic content-based filtering of television news is as relevant today as in 1997. As has been shown e.g. in the recent Informedia TRECVID evaluations, the most successful solutions employ a multi-modal approach, making use of information available in teletext / closed captions transcripts, and via both audio analysis (including speech recognition) and video analysis – including, specifically, video OCR. This validates the overall theoretical design discussed in section 3. A major difference between now and then, though, is that systems built on such principles are commercially available today, most notably the Virage system, albeit with little or no support for Scandinavian languages. Had a project with the same objectives been started at Observer today, it would most likely have involved a thorough testing of such a system, together with an investigation into how to modify and complement it for successful use in the Scandinavian countries. A continuously relevant question is how to best access the rich semantic information contained in the "open captions" subtitles prevalent in Scandinavian broadcasts, for which a prototype video OCR system was developed within this project. Video OCR has been the subject of continued research and development in the years since 1997 and commercial applications have been released, e.g. SRI's ConTEXTract. However, a brief study of recent literature does not indicate that the approaches chosen within this project are now hopelessly obsolete, but rather that some of the findings are still relevant.

9 A&A (Abbreviations and Acronyms)

NB! Several of the acronyms in this list are trivial, especially to anyone with the most basic knowledge of computer technology. They are listed only to confirm that they are used with their commonplace meanings.

AC Alternating Current; in an image processing context, AC often refers to all but the first (i.e. all non-zero-frequency) coefficients of a DCT-compressed image (block)
ANN Artificial Neural Network
ASCII American Standard Code for Information Interchange
ATM Asynchronous Transfer Mode
AV Audio/Video (A/V)
BBC British Broadcasting Corporation
BNN Broadcast News Navigator; a content-based TV news retrieval system developed at Mitre Corporation, Bedford, Massachusetts
CC Closed Captions
CGI Common Gateway Interface
CMU Carnegie Mellon University; Pittsburgh, Pennsylvania
CNN Cable News Network
COTS Commercial Off-The-Shelf
CPU Central Processing Unit
DARPA (U.S.) Defense Advanced Research Program Agency
DC Direct Current; in an image processing context, DC often refers to the first (zero-frequency) coefficient of a DCT-compressed image (block)
DCT Discrete Cosine Transform
DEC Digital Equipment Corporation
FPS Frames Per Second (fps)
FFD Frame-to-Frame Difference; cf. [Hanjalic&al97]
GUI Graphical User Interface
HD Hard Disk
HMM Hidden Markov Models
HP Hewlett-Packard Company
HTML HyperText Markup Language
IBM International Business Machines Corporation
IDVL Informedia Digital Video Library; research system at CMU
IMPACT Interactive Motion Picture Authoring system for Creative Talent; multimedia system developed at Hitachi Central Research Laboratory, Tokyo
IR Information Retrieval
IRIT Institut de Recherche en Informatique de Toulouse; at Université Paul Sabatier, Toulouse, France
JPEG Joint Photographic Expert Group; refers to both a standard for digital image compression and its origin
KTH Kungliga Tekniska Högskolan; Royal Institute of Technology, Stockholm, Sweden
LAN Local Area Network
MAT Multimedia Applications in Telecooperation; research group at the German National Research Center for Information Technology in Sankt Augustin
MoCA MovieContent Analysis; research project at the University of Mannheim, Germany
MIT Massachusetts Institute of Technology
MPEG Moving Picture Experts Group; refers to both a standard for digital video compression and its origin
MS MicroSoft Corporation
NLP Natural Language Processing
NLU Natural Language Understanding
NPAC Northeast Parallell Architectures Center; at Syracuse University, New York

NTT Nippon Telegraph and Telephone Corporation, Japan
OCR Optical Character Recognition
ORL Olivetti and Oracle Research Laboratory
PAL Phase Alternated (by) Line; television standard used in Europe
PCM Pulse-Code Modulation; standard digital audio format
QBIC Query By Image Content; image representation and retrieval system developed at IBM
RAID Redundant Array of Independent Disks; originally meant Redundant Array of Inexpensive Disks
RAM Random Access Memory
RF Radio Frequency
RGB Red Green Blue; image/video color scheme (rgb)
SGI Silicon Graphics Inc.
SGML Standard Generalized Markup Language
SR Speech Recognition
SVT Sveriges Television; Swedish National Television Corporation
TCSH Tenex C Shell (tcsh)
TNS Telemedia Networks and Systems; research group at the MIT Laboratory for Computer Science
UDP User Datagram Protocol
VHS Video Home System
VMR Video Mail Retrieval; research project at Cambridge University and ORL
VVE Virage Video Engine; digital video system developed at Virage Inc., San Mateo, California
WWW World Wide Web

10 References

10.1 Printed publications

[AhangerLittle96] G. Ahanger and T. D. C. Little: "A Survey of Technologies for Parsing and Indexing Digital Video", *Journal of Visual Communication and Image Representation*, 7(1), March 1996, pp 28-43

[ArikiSaito96] Y. Ariki and Y. Saito: "Extraction of TV News Articles Based on Scene Cut Detection Using DCT Clustering", *Proc. ICIP'96*, 1996, pp III:847-850.

[ArikiSugiyama96] Y. Ariki and Y. Sugiyama: "Study of Self-Organization of Speech & Video data – Extraction, classification and retrieval of news video articles", *Proc. Symposium on Self-Organizing Information Base Systems for Creative Research and Development*, Science and Technology Agency, Tokyo, March 1996.

[ArikiSugiyama97] Y. Ariki and Y. Sugiyama: "A TV News Retrieval System with Interactive Query Function", *Proc. 2nd IFCIS Intern. Conf. on Cooperative Information Systems (CoopIS'97)*, Kiawah Island, SC, June 1997, pp 184-192.

[Arman&al93] F. Arman, A. Hsu and M.-Y. Chiu: "Image Processing on Compressed Data for Large Video Databases", *Proc. ACM Multimedia 93*, Anaheim, CA, August 1993, pp 267-272

[BealeJackson90] R. Beale and T. Jackson: *Neural Computing – An Introduction*, Institute of Physics Publishing, Bristol, UK, 1990

[BelkinCroft92] N. J. Belkin and W. B. Croft: "Information filtering and information retrieval: Two sides of the same coin?", *Communications of the ACM*, 35(12), December 1992, pp 29-38

[BoreczkyRowe96] J. S. Boreczky and L. A. Rowe: "Comparison of Video Shot Boundary Detection Techniques", *Proc. SPIE Conference on Storage and Retrieval for Image and Video Databases*, Vol. 2670, San Jose, February 1996, pp 170-179

[Brown&al95] M. G. Brown, J. T. Foote, G. J. F. Jones, K. Spärck Jones and S. J. Young: "Automatic Content-Based Retrieval of Broadcast News", *Proc. ACM Multimedia 95*, San Francisco, November 1995.

[Chellappa&al95] R. Chellappa, C. L. Wilson and S. Sirohey: "Human and Machine Recognition of Faces: A Survey", *Proc. IEEE*, 83(5), May 1995, pp 705-740

[Christel&al96] M. Christel, S. Stevens, T. Kanade, M. Mauldin, R. Reddy and H. Wactlar: "Techniques for the creation and exploration of digital video libraries", in B. Furht (ed.): *Multimedia Tools and Applications*[23], Kluwer Academic Publishers, Norwell, MA, 1996, pp 283-327

[Cook&al97] G. D. Cook, D. J. Kershaw, J. D. M. Christie, S. W. Seymour and S. R. Waterhouse: "Transcription of Broadcast Television and Radio News: The 1996 Abbot System", *Proc. IEEE Intern. Conf. on Acoustics, Speech and Signal Processing (ICASSP-97)*, Munich, April 1997, pp 723-726

[CorridoniDelBimbo95] J. M. Corridoni and A. Del Bimbo: "Automatic Video Segmentation through Editing Analysis", *Proc. 8th Intern. Conf. on Image Analysis and Processing (ICIAP'95)*, San Remo, Italy, September 1995

[CorridoniDelBimbo96] J. M. Corridoni and A. Del Bimbo: "Structured Digital Video Indexing", *Proc. 13th Intern. Conf. on Pattern Recognition (ICPR'96)*, Vienna, Austria, 1996

[Doermann&al95] D. Doermann, E. Rivlin and I. Weiss: "Logo Recognition", *Technical Report*, CS-TR-3145, Center for Automation Research, University of Maryland, College Park, MD, 1995

[Flickner&al95] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele and P. Yanker: "Query by Image and Video Content: The QBIC System", *IEEE Computer*, September 1995, pp 23-32

[Furht&al95] B. Furht, S. W. Smoliar and H.J. Zhang: *Video and Image Processing in Multimedia Systems*, Kluwer Academic Publishers, Norwell, MA, 1995

[GonzalezWoods92] R. C. Gonzalez and R. E. Woods: *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992

[23] *Multimedia Tools and Applications* is the name of both a book and a quarterly journal, both edited by Borko Furht and published by Kluwer.

[Hampapur&al96] A. Hampapur, R. Jain and T. E. Weymouth: "Production Model Based Digital Video Segmentation", in B. Furht (ed.): *Multimedia Tools and Applications*, Kluwer Academic Publishers, Norwell, MA, 1996, pp 111-153

[Hampapur&al97] A. Hampapur, A. Gupta, B. Horowitz, C.-F. Shu, C. Fuller, J. Bach, M. Gorkani and R. Jain: "Virage Video Engine", *Proc. SPIE Conf. on Storage and Retrieval for Image and Video Databases V*, San Jose, CA, February 1997, Vol. 3022, pp 188-198

[Hanjalic&al97] A. Hanjalic, M. Ceccarelli, R. L. Lagendijk and J. Biemond: "Automation of systems enabling search on stored video data", *Proc. SPIE Conf. on Storage and Retrieval for Image and Video Databases V*, San Jose, CA, February 1997, Vol. 3022, pp 427-438

[Hauptmann&al95] A. G. Hauptmann, M. J. Witbrock and M. G. Christel: "News-on-Demand: An Application of Informedia Technology", *D-Lib Magazine*, September 1995

[HauptmannWactlar97] A. G. Hauptmann and H. D. Wactlar: "Indexing and Search of Multimodal Information", *Proc. IEEE Intern. Conf. on Acoustics, Speech and Signal Processing (ICASSP-97)*, Munich, April 1997

[HauptmannWitbrock97] A. G. Hauptmann and M. J. Witbrock: "Informedia: News-on-Demand Multimedia Information Acquisition and Retrieval", in M. Maybury (ed.): *Intelligent Multimedia Information Retrieval*, AAAI Press / MIT Press, Cambridge, MA, 1997

[Hjelsvold&al95] R. Hjelsvold, S. Langørgen, R. Midstraum and O. Sandstå: "Integrated Video Archive Tools", *Proc. ACM Multimedia 95*, San Francisco, November 1995.

[Jonas&al95] K. Jonas, H. Jungblut, J. Kaeber, M. Kaul, I. Müller, H. Santo, J. Schäfer and R. Wegner: "The Information Footprint - A Satellite-based Information on Demand Teleservice", *Proc. 6th Joint European Networking Conf. (JENC6),* Tel Aviv, Israel, May 1995. Also published in*: Computer Networks and ISDN Systems*, 28, 1996, pp 563-573

[Karperyd98] M. Karperyd: "Mediebevakning med IT-stöd: Hantering av sökprofiler för fritextsökning", *Master's Thesis*, 98-X-064, Royal Institute of Technology, Stockholm, 1998 (in Swedish)

[Kim96] H.-K. Kim: "Efficient Automatic Text Location Method and Content-Based Indexing and Structuring of Video Database", *Journal of Visual Communication and Image Representation*, 7(4), December 1996, pp 336-344

[Kurakake&al97] S. Kurakake, H. Kuwano and K. Odaka: "Recognition and visual feature matching of text region in video for conceptual indexing", *Proc. SPIE Conf. on Storage and Retrieval for Image and Video Databases V*, San Jose, CA, February 1997, Vol. 3022, pp 368-379

[Kurmanowicz96] M. Kurmanowicz: "Video Indexing via Closed Captioning", *Report*, Index. No. 80, Northeast Parallell Architectures Center, Syracuse University, 1996

[LeGall91] D. Le Gall: "MPEG: A Video Compression Standard for Multimedia Applications", *Communications of the ACM*, 34(4), April 1991

[LiebholdHoffert91] M. Liebhold and E. M. Hoffert: "Toward an Open Environment for Digital Video", *Communications of the ACM*, 34(4), April 1991, pp 103-112

[Lienhart96] R. Lienhart: "Automatic Text Recognition for Video Indexing", *Technical Report*, TR-96-006, Department of Computer Science, University of Mannheim, Germany, April 1996. Also appearing as "Indexing and Retrieval of Digital Video Sequences based on Automatic Text Recognition", *Proc. ACM Multimedia 96*, Boston, November 1996, pp 11-20

[LienhartStuber95] R. Lienhart and F. Stuber: "Automatic text recognition in digital videos", *Technical Report*, TR-95-036, Department of Computer Science, University of Mannheim, Germany, December 1995. Also appearing in *Proc. SPIE Conf. on Image and Video Processing IV*, Vol. 2666-20, 1996

[Lindblad&al95] C. J. Lindblad, D. J. Wetherall, W. F. Stasior, J. F. Adam, H. H. Houh, M. Ismert, D. R. Bacher, B. M. Phillips and D. L. Tennenhouse: "ViewStation Applications: Implications for Network Traffic", *IEEE Journal on Selected Areas in Communications*, 13(5), June 1995, pp 768-778

[LindbladTennenhouse96] C. J. Lindblad and D. L. Tennenhouse: "The VuSystem: A Programming System for Compute-Intensive Multimedia", *IEEE Journal on Selected Areas in Communications*, 14(7), September 1996

[Luther92] A. C. Luther: *Digital Video in the PC Environment*, 2nd ed., McGraw-Hill, 1991

[MengChang96] J. Meng and S.-F. Chang: "CVEPS – A Compressed Video Editing and Parsing System", *Proc. ACM Multimedia 96*, Boston, November 1996

[MerlinoMorey97] A. Merlino and D. Morey: "Personalized Broadcast News from BNN", *The Edge Newsletter*, 1(2), Mitre Corp., Bedford, MA, July 1997, pp 3-4.

[Merlino&al97] A. Merlino, D. Morey and M. Maybury: "Broadcast News Navigation using Story Segmentation", *Proc. ACM Multimedia 97*, Seattle, WA, November 1997

[ML97] *Using Matlab*, Version 5, The MathWorks Inc., Natick, MA, 1997

[MLIPT97] *Matlab Image Processing Toolbox – User's guide*, Version 2, The MathWorks Inc., Natick, MA, 1997

[MotegiAriki96] Y. Motegi and Y. Ariki: "Indexing to TV News Articles Based on Character Recognition", *Technical Report*, PRU95-240, Institute of Electronics, Information and Communication Engineers (IEICE), Japan, March 1996, pp 33-40. (Abstract in English, paper in Japanese)

[NakamuraKanade97] Y. Nakamura and T. Kanade: "Semantic Analysis for Video Contents Extraction – Spotting by Association in News Video", *Proc. ACM Multimedia 97*, Seattle, WA, November 1997

[Ohyo&al94] J. Ohyo, A. Shio and S. Akamatsu: "Recognizing Characters in Scene Images", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(2), February 1994, pp 215-220

[OtsujiTonomura93] K. Otsuji and Y. Tonomura: "Projection Detecting Filter for Video Cut Detection", *Proc. ACM Multimedia 93*, Anaheim, CA, August 1993, pp 251-257

[Pallett&al97] D. S. Pallett, J. G. Fiscus and M. A. Przybocki: "1996 Preliminary Broadcast News Benchmark Tests", *Proc. 1997 DARPA Speech Recognition Workshop*, Chantilly, VA, February 1997

[PatelSethi97a] N. V. Patel and I. K. Sethi: "Video Classification Using Speaker Identification", *Proc. SPIE Conf. on Storage and Retrieval for Image and Video Databases V*, San Jose, CA, February 1997, Vol. 3022, pp 218-225

[PatelSethi97b] N. V. Patel and I. K. Sethi: "Video Shot Detection and Characterization for Video Databases", *Pattern Recognition*, 30(4), April 1997, pp 583-592

[Pentland&al96] A. Pentland, R. W. Picard and S. Sclaroff: "Photobook: Content-Based Manipulation of Image Databases", *Intern. Journal of Computer Vision*, 18(3), 1996, pp 233-254

[Pfeiffer&al96a] S. Pfeiffer, R. Lienhart, S. Fischer and W. Effelsberg: "Abstracting Digital Movies Automatically", *Journal of Visual Communication and Image Representation*, 7(4), December 1996, pp 345-353

[Pfeiffer&al96b] S. Pfeiffer, S. Fischer and W. Effelsberg: "Automatic Audio Content Analysis", *Technical Report*, TR-96-008, Department of Computer Science, University of Mannheim, Germany, April 1996, also appearing in *Proceedings*, ACM Multimedia 96, Boston, November 1996

[Rowley&al95] H. A. Rowley, S. Baluja and T. Kanade: "Human Face Detection in Visual Scenes", *Technical Report*, CMU-CS-95-158R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, November 1995

[RådeWestergren90] L. Råde and B. Westergren: *BETA – Mathematics Handbook*, 2nd ed., Studentlitteratur, Lund, Sweden, 1990

[Schneider94] M. Schneider: "What is Teletext?", *technical whitepaper*, Philips Semiconductors, June 1994

[Shannon48] C. E. Shannon: "A Mathematical Theory of Communication", *The Bell System Technical Journal*, Vol. 27, July, October 1948, pp 379-423, 623-656

[SmithKanade97] M. A. Smith and T. Kanade: "Video Skimming and Characterization through the Combination of Image and Language Understanding Techniques", *Technical Report*, CMU-CS-97-111, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, February 1997

[Stern97] R. M. Stern: "Specification of the 1996 Hub 4 Broadcast News Evaluation", *Proc. 1997 DARPA Speech Recognition Workshop*, Chantilly, VA, February 1997

[Strawn94] J. Strawn: "Digital Audio Representation and Processing", in J. F. Koegel Buford (ed.): *Multimedia Systems*, ACM Press / Addison-Wesley, New York, 1994

[Ström96] N. Ström: "Continuous speech recognition in the WAXHOLM dialogue system", *Quarterly Progress and Status Report*, No. 4/1996, Dept. of Speech, Music and Hearing, Royal Institute of Technology (KTH), Stockholm, pp 67-95

[Tenzler&al96] T. Tenzler, K. Jonas and J. Kaeber: "News on Demand", *Proc. IEEE Global Telecommunications Conf. / IEEE Global Internet '96*, London, November 1996

[Trier&al96] Ø. D. Trier, A. K. Jain and T. Taxt: "Feature Extraction Methods for Character Recognition – A Survey", *Pattern Recognition*, 29(4), April 1996, pp 641-661

[TurkPentland91] M. Turk and A. Pentland: "Eigenfaces for recognition", *Journal of Cognitive Neuroscience*, 3(1), 1991, pp 71-86

[Ueda&al91] H. Ueda, T. Miyatake and S. Yoshizawa: "IMPACT: An Interactive Natural-Motion-Picture Dedicated Multimedia Authoring System", *Proc. ACM CHI'91*, 1991

[vanRijsbergen79] C. J. van Rijsbergen, *Information Retrieval*, 2nd ed., Butterworths, London, 1979

[Wactlar&al96] H. D. Wactlar, T. Kanade, M. A. Smith and S. M. Stevens: "Intelligent Access to Digital Video: Informedia Project", *IEEE Computer*, May 1996.

[WechslerSchäuble95] M. Wechsler and P. Schäuble: "Speech Retrieval Based on Automatic Indexing", *Proc. Final Workshop on Multimedia Information Retrieval (MIRO'95)*, Glasgow, Scotland, September 1995, Published by Springer Verlag

[Wei&al97] Q. Wei, H.J. Zhang and Y. Zhong: "A robust approach to Video Segmentation Using Compressed Data", *Proc. SPIE Conf. on Storage and Retrieval for Image and Video Databases V*, San Jose, CA, February 1997, Vol. 3022, pp 448-456

[Weng&al97] F. Weng, H. Bratt, L. Neumeyer and A. Stolcke: "A Study of Multilingual Speech Recognition", *Proc. EUROSPEECH97*, Greece, September 1997.

[Wold&al96] E. Wold, T. Blum, D. Keislar and J. Wheaton: "Content-Based Classification, Search and Retrieval of Audio", *IEEE Multimedia*, 3(3), 1996, pp 27-36

[YeoLiu95] B.-L. Yeo and B. Liu: "A Unified Approach to Temporal Segmentation of Motion JPEG and MPEG Compressed Video", *Proc. IEEE Intern. Conf. on Multimedia Computing and Systems (ICMCS'95)*, May 1995

[YeoLiu96] B.-L. Yeo and B. Liu: "Visual Content Highlighting via Automatic Extraction of Embedded Captions on MPEG Compressed Video", *Proc. SPIE Conf. on Digital Video Compression: Algorithms and Technologies*, San Jose, CA, February 1996, Vol. 2668, pp 38-47

[Young&al97] S. J. Young, M. G. Brown, J. T. Foote, G. J. F. Jones and K. Spärck Jones: "Acoustic Indexing for Multimedia Retrieval and Browsing", *Proc. IEEE Intern. Conf. on Acoustics, Speech and Signal Processing (ICASSP-97)*, Munich, April 1997, Vol. 1, pp 199-202

[YowCipolla97] K. C. Yow and R. Cipolla: "Feature-Based Human Face Detection", *Journ. of Image and Vision Computing*, 15(9), September 1997, pp 713-735

[Zhang&al97] H.J. Zhang, J. Wu, D. Zhong and S. W. Smoliar: "An Integrated System for Content-Based Video Retrieval and Browsing", *Pattern Recognition*, 30(4), April 1997, pp 643-658

[Zhong&al95] Y. Zhong, K. Karu and A. K. Jain: "Locating Text in Complex Color Images", *Pattern Recognition*, 28(10), 1995, pp 1523-1535

10.2 Online publications and web sites

[Berkeley97] "Computer Vision Research", University of California at Berkeley, `http://elib.cs.berkeley.edu/vision.html`

[BNN97] "Broadcast News Navigator", Mitre Corp., `http://www.mitre.org/resources/centers/advanced_info/g04f/bnn.html`

[CapFAQ97] The Caption FAQ, maintained by Gary Robson, `http://www.caption.com/capfaq/`

[CMUSpeech97] `http://www.speech.cs.cmu.edu/speech/`

[DR97] Danmarks Radio (Danish Broadcasting Corp.), `http://www.dr.dk/`

[Dubner97] Dubner International Inc., `http://www.dubner.com/`

[Entropic97] `http://www.entropic.com/`

[Glimpse97] `http://glimpse.cs.arizona.edu/` and `http://glimpse.cs.arizona.edu/ghttp/`

[Hauppauge97] Hauppauge Digital Inc., `http://www.hauppauge.com/`

[IBMDL97] "IBM Digital Library", http://www.software.ibm.com/is/dig-lib/factst.htm

[Informedia97] http://www.informedia.cs.cmu.edu/html/

[MATNoD95] "MAT News on Demand", http://mats.gmd.de/ralf/english/nod.html

[MoCATR97] "MoCA Text Segmentation and Text Recognition in Digital Videos", University of Mannheim, Germany, http://www.informatik.uni-mannheim.de/informatik/pi4/projects/MoCA/MoCA_TextRecognition.html

[MW97] The MathWorks Inc., http://www.mathworks.com/

[NPACVoD97] "NPAC Video on Demand", http://trurl.npac.syr.edu/rlvod/

[NRK97] Norsk Rikskringkasting (Norwegian Broadcasting Corp.), http://www.nrk.no/

[Nuance97] http://www.nuance.com/

[OnlineTV97] http://www.tvtoday.de/onlinetv/

[Opt97] http://www.opt.com/

[Pelican97] http://www.pelican.com.au/

[Philips97] http://www.semiconductors.philips.com/

[QBIC97] "QBIC Home Page", http://wwwqbic.almaden.ibm.com/

[SoftSound98] http://www.softsound.com/

[SVT97] Sveriges Television (Swedish Television Corp.), http://www.svt.se/

[Televitesse97] http://www.televitesse.com/

[VVCB97] "Virage Video Cataloger & Browser" http://www.virage.com/vide_cb.htm

10.3 References for section 8

[ALERT03] "Alert System for Selective Dissemination of Multimedia Information – Final Report", 2003, downloaded from http://www.hltcentral.org/projects/detail.php?acronym=ALERT

[Antani&al02] S. Antani, R. Kasturi and R. Jain: "A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video", *Pattern Recognition*, 35, 2002, pp 945-965

[Autonomy05] http://www.autonomy.com/

[Crandall&al03] D. Crandall, S. Antani and R. Kasturi: "Extraction of special effects caption text events from digital video", *International Journal on Document Analysis and Recognition*, 5, 2003, pp 138-157

[CTT05] The Centre for Speech Technology at the Royal Institute of Technology, Stockholm, http://www.speech.kth.se/ctt/

[HauptmannChristel04] A. Hauptmann and M. Christel: "Successful Approaches in the TREC Video Retrieval Evaluations", *Proc. ACM Multimedia 2004*, New York, October 2004, pp 668-675

[Hauptmann&al03] A. Hauptmann, D. Ng, R. Baron, M-Y. Chen, M. Christel, H. Duygulu, C. Huang, W-H. Lin, H. Wactlar, N. Moraveji, N. Papernick, C.G.M. Snoek, G. Tzanetakis, J. Yang, R. Yan and R. Jin: "Informedia at TRECVID 2003: Analyzing and Searching Broadcast News Video", *Proc. TRECVID 2003*, Gaithersburg, MD, November 2003, (http://www-nlpir.nist.gov/projects/trecvid/)

[HTK05] The HTK web site at Cambridge University, http://htk.eng.cam.ac.uk/

[Gauvain&al02] J.L. Gauvain, L. Lamel, and G. Adda: "The LIMSI Broadcast News Transcription System", *Speech Communication*, 37(1-2), 2002, pp 89-108

[Identix05] http://www.identix.com/

[Idioma05] http://www.idiomasolutions.com/

[ISS05] Internet Security Systems corporate web site, http://www.iss.net/

[Iurgel&al02] U. Iurgel, S. Werner, A. Kosmala, F. Wallhoff and G. Rigoll: "Audio-Visual Analysis of Multimedia Documents for Automatic Topic Identification", *Proc. SPPRA 2002*, Crete, June 2002

[IST05] The European Commission IST Web, `http://www.cordis.lu/ist/`

[Kalman&al01] M. Kalman, I. Keslassy, D. Wang and B. Girod: "Classification of compound images based on transform coefficient likelihood", *Proc. Intern. Conf. on Image Processing*, Thessaloniki, Greece, October 2001, pp 750-753

[Kraaij&al04] W. Kraaij, A.F. Smeaton, P. Over and J. Arlandis: "TRECVID 2004 – An Overview", *Proc. TRECVID 2004*, (`http://www-nlpir.nist.gov/projects/trecvid/`)

[Lienhart01] R. Lienhart: "Reliable Transition Detection in Videos: A Survey and Practitioner's Guide", *Int. Journal of Image and Graphics*, 1(3), 2001, pp 469-486

[Lienhart03] R. Lienhart: "Video OCR: A Survey and Practitioner's Guide", in *Video Mining*, Kluwer Academic Publishers, 2003, pp 155-184

[Lucas&al03] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong and R. Young: "ICDAR 2003 Robust Reading Competitions", *Proc. 7th Intern. Conf. on Document Analysis and Recognition (ICDAR)*, vol. 2, 2003, pp 682-687

[Maybury&al04] M. Maybury, W. Greiff, S. Boykin, J. Ponte, C. McHenry and L. Ferro: "Personalcasting: Tailored Broadcast News", *User Modeling and User-Adapted Interaction*, 14(1), February 2004, pp 119-144

[Neto&al03] J. Neto, H. Meinedo, R. Amaral and I. Trancosco: "A system for selective dissemination of information resulting from the ALERT project", *Proc. ISCA workshop on Multilingual Spoken Document Retrieval*, Macau/Hong Kong, 2003

[Nuance05] `http://www.nuance.com/`

[Observer05] `http://www.observergroup.com/`

[PENG05] `http://www.peng-project.org/`

[Perlmutter&al96] K. O. Perlmutter, N. Chaddha, J. B. Buckheit, R. M. Gray and R. A. Olshen: "Text Segmentation in Mixed-Mode Images Using Classification Trees and Transform Tree-Structured Vector Quantization", *Proc. IEEE Intern. Conf. on Acoustics, Speech and Signal Processing (ICASSP-96)*, vol. 4, 1996, pp 2231-2234

[REVEAL05] `http://www.reveal-this.org/ or http://sifnos.ilsp.gr/RevealThis/`

[SAIL05] `http://www.sail-technology.com/`

[SnoekWorring05] C.G.M. Snoek and M. Worring: "Multimodal Video Indexing: A Review of the State-of-the-art", *Multimedia Tools and Applications*, 25(1) , January 2005, pp 5-35

[SoftSound05] `http://www.softsound.com/`

[Sphinx05] The CMU Sphinx resource page, `http://cmusphinx.sourceforge.net/sphinx4/`

[SRI05] `http://www.sri.com/`

[TVEyes05] `http://www.tveyes.com/`

[Virage05] `http://www.virage.com/`

[XML05] The World Wide Web Consortium's XML Working Groups home page, `http://www.w3.org/xml/`

[Zhong&al00] Y. Zhong, H.J. Zhang and A. K. Jain: "Automatic Caption Localization in Compressed Video", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(4) , April 2000, pp 385-392