Multimed Tools Appl (2007) 34:201–220
DOI 10.1007/s11042-006-0090-7

Early evaluation of future consumer AV content analysis applications with PC networks

Fons de Lange · Jan Nesvadba

Published online: 27 January 2007

Abstract The paper deals with software productivity improvement for consumer multimedia devices by means of PC and component technology and shows how this is done for complex real-time *content analysis* applications used in advanced new storage products of the future. *Content analysis* is a relatively new and immature technology. It is used for browsing and searching particular content items among thousands of others on "big" embedded storage devices like hard disks. As the storage capacity of hard disk and flash continues to grow rapidly, *content analysis* is bound to become a key enabling technology in future storage products. A major problem with content analysis features (and many other features as well) is that underlying algorithms are unstable, sometimes unavailable, or at least, very much in their infancy, and as such, subject to frequent changes. The paper describes an approach to facilitate *early evaluation* and integration of such immature features. This is done by packing each feature, *as-is*, into components and by providing PC network technology to interconnect them. In our prototyping framework, each component is an independent executable program that runs on some PC in the network, streaming AV data via TCP/IP and being controlled through UPnP networking. Experiences with large-scale prototyping activities we have carried out for the assessment of future *content analysis* systems, show that a PC based prototyping approach enables the integration of many different media processing features in a short time and that it allows for accurate analysis of the resource (CPU/memory) requirements of such components.

Keywords Multimedia content analysis · PC networking · Prototyping · UPnP · Service oriented architecture

1 Introduction

This paper presents the problem of "early" feature evaluation; i.e. how to assess the sense and simplicity of new features and feature combinations for yet nonexistent products and

F. de Lange (✉) · J. Nesvadba
Philips Research, Eindhoven, The Netherlands
e-mail: Fons.de.Lange@philips.com

J. Nesvadba
e-mail: Jan.Nesvadba@philips.com

how can this be done with little effort when features are still in their infancy? Although difficult, evaluation of still immature features is a must to enable the assessment of important aspects of a possible future product. In fact, when envisioning a new product-concept, very often still a lot of features and functions are nonexistent and need to be invented first, subsequently they must be implemented and combined to give a first impression of a product. Next, some evaluation of the feature combination must be possible to judge its value and feasibility. The outcome of this process is an updated vision of the imaginary product and will fuel the generation of new ideas and the development of other new features. Figure 1 visualizes this process of *early feature evaluation*, shown as four phases:

1. Imagine

   This is about envisioning and imagining a new product. An example of a *product vision* is a *Personal Video Recorder* that enables a user to find and watch any TV program that has been broadcast during the last few months for a set of preferred channels.
2. Invent

   Here, one has to think about the types of features and the enabling technologies required for the envisioned product. An example of an important enabling technology is *Content Analysis* [1] such as *Virtual Channel Creation* [7] that enables a user watch all his/her favorite TV programs.
3. Implement

   To increase the understanding and learn more about the possibilities, benefits and (technical) limitations of an imaginary product, critical parts must be prototyped. An

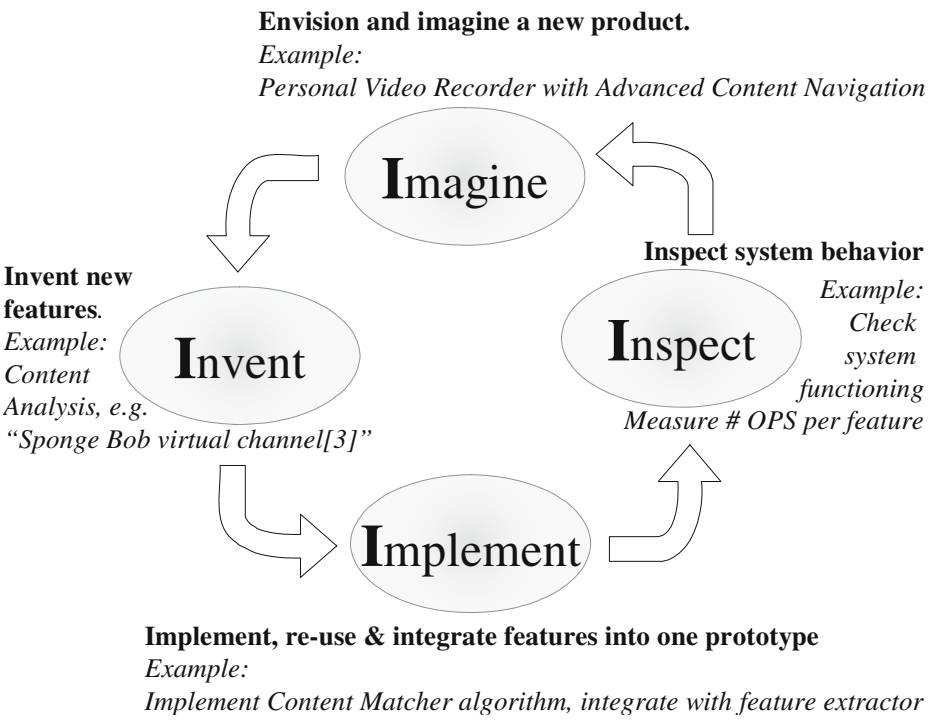


Fig. 1 Early evaluation of features to assess future product concepts based on audio/video analysis

effective way of doing this is to look for any technology that is relevant to the product concept and easy to integrate with other technologies. System functionality that is crucial to the product, but which is not available anywhere must first be invented and then created from scratch.

4. Inspect

Once a prototype is created that implements sufficient functionality of the envisioned product, one can analyze the system behavior, determine component interfaces/interactions and measure important characteristics of specific feature combinations, e.g. the memory, streaming bandwidth and performance requirements. Consider a specific feature for an imaginary *personal video recorder* such as a *football match detector*. By prototyping and analyzing its behavior one can determine if it is accurate enough, if it is feasible in combination with other features—with respect to performance and memory usage—and last but not least, if the feature is attractive and easy to use. This will further stimulate the imagination and ingenuity; see Fig. 1, leading to an improved product concept.

A complicating factor is that features, if implemented and available at all, are very much in their infancy and subject to frequent changes as applied by the feature designer. This problem is especially applicable to *content analysis* for storage systems, where *content analysis* supports the content retrieval and navigation process. Since the storage capacity of different types of storage devices, such as hard disk and flash, is rapidly growing, more and more audio, video and photo content can be stored on these devices. In the near future this will grow to several Terabytes of storage within a single Hard Disk device [8], capable of holding massive amounts of AV data, see Fig. 2. It becomes clear that *AV viewing*, *AV searching* and *AV browsing* functionality is therefore essential for a successful introduction of mass storage devices in CE products.

Content analysis is a key enabling technology in this respect. In the domain of AV content analysis, many new algorithms, such as *speech/music discrimination*, *scene change detection*, *commercial detection* [5, 9, 13], etc. are developed at an increasing rate and become available for third party use at sites inside and outside Philips, companies and universities [10]. These algorithms enable audio video content to be segmented such that content-viewing, browsing and searching can be greatly enhanced. All these *content analysis* algorithms are different with respect to how they are designed and implemented. Often they are developed with different programming tools, using different programming languages and having different communication models for interacting with their environment.

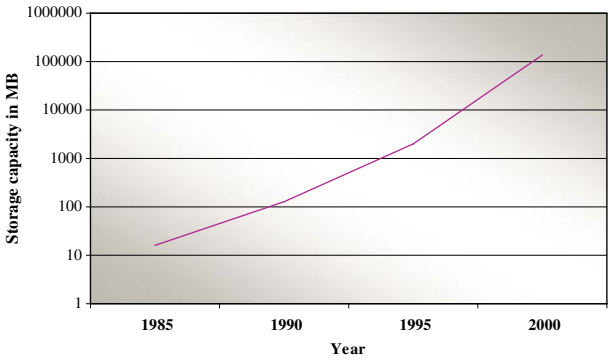


Fig. 2 Trend for Hard Disk storage capacity

The solution that the paper describes to enable product-concept assessment for such *content analysis* enabled systems is a PC based design methodology and Service Oriented Architecture [15] for easy integration of features. This is done using component technology for packing each feature, *as-is*, into components and by providing standard technology and tools/platform to flexibly interconnect them. In our prototyping framework, each component is an independent executable program that communicates with other components via TCP/IP and UPnP networking [17]. Basic TCP/IP is used for streaming data over the network, while UPnP is used to enable applications to automatically find components, set-up connections between them and to control them. All this irrespective of their location in the network. Section 3 describes this approach in more detail.

The paper is structured as follows. Section 2 gives a short introduction to the field of AV *content analysis*. Section 3 describes the PC based approach to early evaluation of (*content analysis*) features and algorithms. Section 4 illustrates the effectiveness of the approach. In particular it describes the architecture of a large demonstrator system, integrating many AV content analysis algorithms and an AV database. Finally, Section 5 presents the major conclusions of the PC based feature integration/evaluation approach.

2 Introduction to AV content analysis

This section provides background information on multimedia content analysis.

2.1 Rationale

Extrapolation of the storage capacity of a single hard disk drive, shown in Fig. 2, yields 100 TB in 10 years from now, which comes close to storing all songs in the world.[1] Table 1 gives an indication of what 1 TB can offer to a CE device.

It is obvious that it becomes an impossible job to find a particular video clip among thousands of others on some storage device by just watching video segments. The only way to find content is to use metadata to summarize and describe the content, which can then be used by a user to more efficiently search for content.

Several sources of metadata for broadcast video exist. These are Teletext for analog TV in Europe, Digital Video Broadcast *Service Information* (http://www.dvb.org) and the Internet. The reliability of all these sources of metadata depends on the associated profitability for the provider of these services. Moreover, different providers use different metadata formats and metadata semantics. All this makes it difficult to reliably and consistently annotate the recorded AV material to enable easy navigation and retrieval of content.

Native content analysis, i.e. content analysis done within the consumer device, eliminates these problems. It enables *duty-free* gathering of metadata for any received and recorded AV material in a consistent way. It can be used to generate metadata in the absence of a metadata service, to enrich the provided metadata to offer extended searching capabilities, and last but not least to enable new advanced content navigation features such as *commercial skip* [5] and *virtual channel* creation [7].

[1] According to the CD database on the internet, http://www.cddb.org, there are currently 20 million songs on CD.

Table 1 What can be recorded on a 1 TB storage device

Number	Content type	Item-size
200	DVD	5 GB
1000	VCD	1 GB
2000	CD	500 MB
200,000	MP3	5 MB
1,000,000	Photo	1 MB

2.2 Content analysis process

Audiovisual content analysis can be applied to a variety of content types such as broadcast commercial content, downloaded content or private home video content.

In consumer electronics (CE) devices the content management, retrieval and navigation has to be intuitive, easy to use and *lean-backward* oriented. As a consequence the interaction with the user has to be performed on a semantic meaningful level, which requires high-level semantic metadata about the audiovisual content.

Semantic metadata can be extracted through the combination of various low- and mid-level descriptors of multiple modalities such as audio, video and text. The audio and audio–visual descriptors are extracted by analyzing the audio, respectively audio–visual signal, either in the baseband (pixels for video or PCM[2] for audio) or the compressed domain e.g. MPEG-2.

Figure 3 visualizes the overall process of semantic metadata extraction, consisting of three phases: *low-level-, mid-level- and high-level feature extraction*.

At each stage in the process, metadata can be stored and retrieved from memory, to be used by the next step in the process and by navigation applications.

Each step is described in more detail in the next subsections.

2.2.1 Low-level feature extraction

Low-level features in the audio–visual domain can be descriptors per frame, such as motion, luminance, color, texture, hard transitions, compression factor, audio energy, audio frequency spectrum coefficients and audio modulation.

The limited processing power requirements of audio analysis allow for format independent processing in e.g. the PCM domain on current commodity processing platforms in consumer electronic (CE) devices. On contrary, the video content analysis algorithms are quite processing power demanding. State-of-the-art solutions are either based on parameters that are generated during video compression, on commodity CE platforms, or based on pixel analysis on advanced multimedia ICs.

The Philips Chrysalis (PNX7100) integrated circuit (http://www.semiconductors.philips.com/products/nexperia/home/products/dvd_recording/pnx7100/) is an example of a MPEG-2 CODEC that generates a large number of audiovisual parameters, described below, that can be used for content analysis.

[2] Pulse Code Modulation.

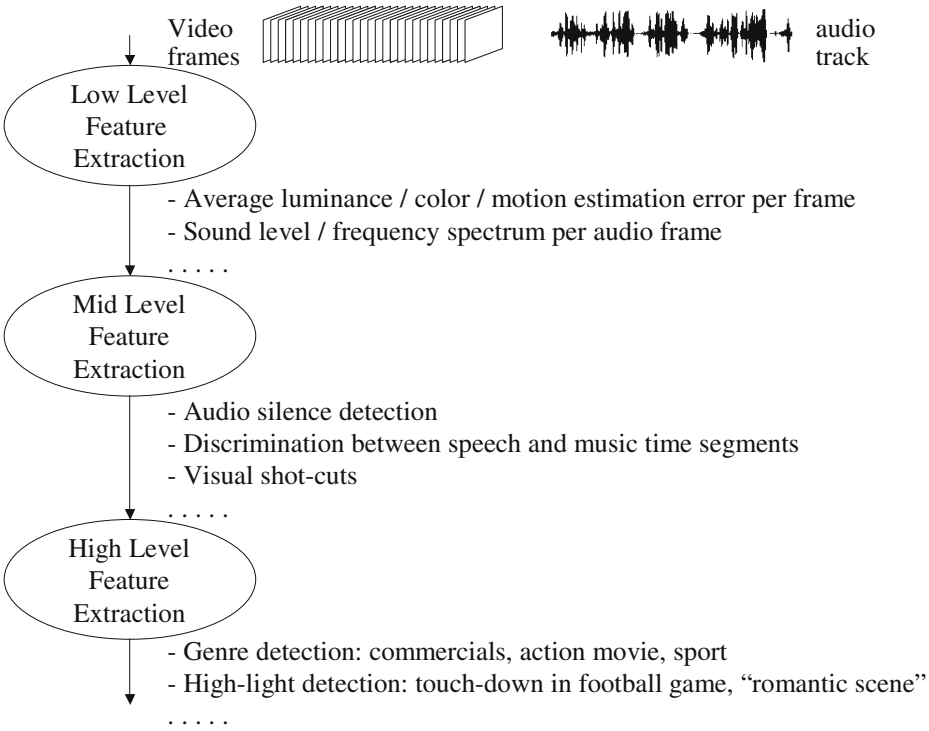


Fig. 3 Overall AV content analysis process

2.2.2 Mid-level feature extraction

Mid-level audiovisual features are usually based on a combination of various low-level features. Audio related mid-level features are e.g. silences, beat, rhythm, tempo and audio/music genre probabilities [9]. Equivalent in the video domain are features such as detection of black frames, a 4:3-to-16:9 aspect ratio differentiation and video transition detection. The availability of compression-related low-level parameter inside a dedicated hardware block of the PNX7100 enable a cost-efficient real-time generation of various mid-level parameters, including shot boundary transitions (shot cuts), beat, rhythm, tempo and music genre classification.

To be more specific, Fig. 4 sketches the schematic of a general MPEG-2 compressor as can be found in the PNX7100.

Video compression parameters such as the luminance average value (luminance DC value) and color average of individual frames can be derived in this compressor after the Discrete Cosine Transformation (DCT), see point *a* in Fig. 4. The motion compensated inter-frame correlation factor after the motion estimator, see point *b* in Fig. 4, provides an expressive input for the detection of shot boundary transitions in the video signal, also called *visual shot cuts* or *shot boundaries* as explained in [11], see Fig. 5. The smart reuse of available MPEG compression parameters, such as the inter-frame correlation factor, reduces the required processing for shot boundary detection to less then 1/2 MIPS (million instructions per second).

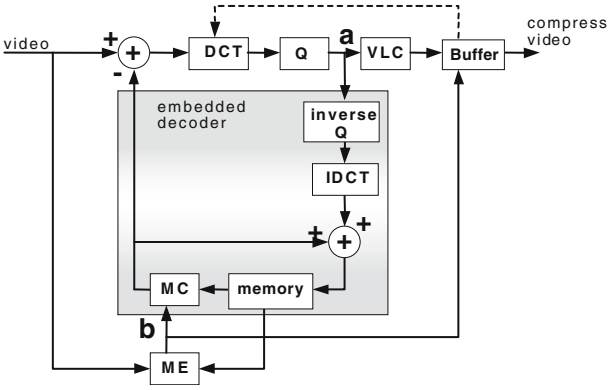


Fig. 4 Low/mid level feature extraction by MPEG compression

Furthermore, audiovisual-related mid-level features can be based on the combination of audio and video low-level features as well as AV mid-level features. As an example, consider scene change detection based on video shot-cut detection and audio silence detection. Here, meaningful audiovisual scene changes are detected by temporal correlation of audio silences and shot boundary transitions [13], see Fig. 6.

2.2.3 High-level feature extraction

Features categorized as high-level features mostly have a semantic meaning to users. Audio-related features of this category include the style of the music, e.g. classical or rock music, spoken keywords of speech and mood of speakers. Video-related features are genre, such as movie, documentary or news bulletin. Other examples are more processing intensive face-detection and face-recognition [12] algorithms. As an example a Viola-Jones-based face detector as described in [12] can use up almost 80% of a state-of-the-art general purpose processor (3 GHz Pentium IV).

Yet another example of an audiovisual high-level feature, corresponding to a video segment, is a TV commercial. Several techniques and implementations for automatic TV

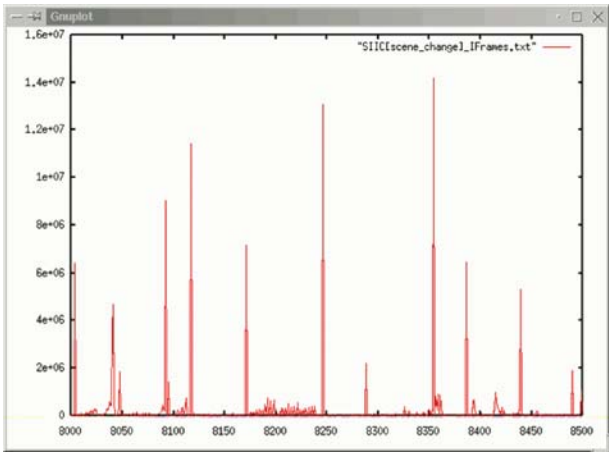


Fig. 5 Visual shot-cut indication for video sequence, measured at point *b* in Fig. 4

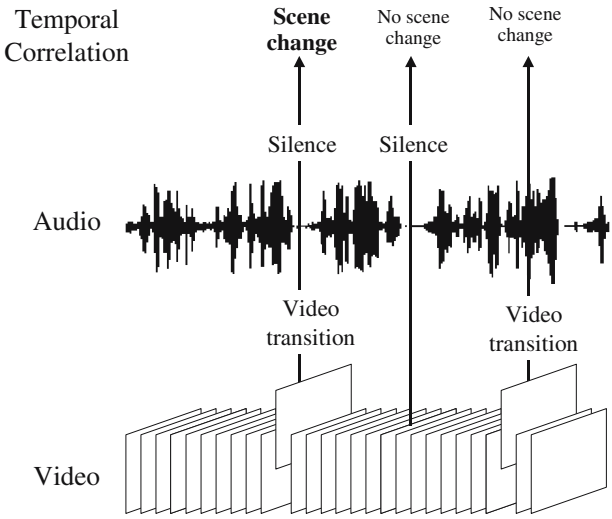


Fig. 6 Detection of semantic meaningful audiovisual scene changes

commercial detection are known from literature. As an example, consider [5], where a combination of a few mid-level features such as mono-chrominance frames, letterbox instances (i.e. 4:3/16:9 aspect ratio), hard video transitions, i.e. shot cuts and silences is used to implement a TV commercial detector with high accuracy (with about 90% recall and almost 100% precision). Due to the reuse of hardware provided low- and mid-level parameter, such a TV commercial indicator can be implemented with neglectable processing costs, i.e. in the range of several MIPS.

2.3 Applications based on content analysis

Applications based on content analysis are meant to improve the searching of specific AV content and/or enable advanced navigation within single content items. An example of the first is an application that clusters all episodes of a particular television series. An example of the second is an application that automatically creates chapters for single content items. This application has been prototyped on a set of networked PCs, which is described in more detail in Section 4. Figure 7 shows the graphics user interface of this application, called *MediaBrowser* [6].

The application shows key-frames, where each key-frame represents the start of a new, semantically correlated, chapter. The user is able to view and select all key-frames with this user interface to resume playback at a particular chapter, skip chapters or change the chapter-viewing order. The menu is created in real-time by *scene change detection* based on low-level features as illustrated in Fig. 8.

This can be done in real-time on consumer electronic devices, such as a DVD/HDD recorder, to automatically chapter recorded audiovisual content.

In combination with genre detectors, such as a *commercial indicator*, it is possible to realize applications that are capable to segment recorded audiovisual content into correlating audiovisual content sequences with meaningful genre descriptions that enable the user to browse in a non-linear way both inside and between recorded content items.

Fig. 7 Automatically generated menu for chapter-based content browsing

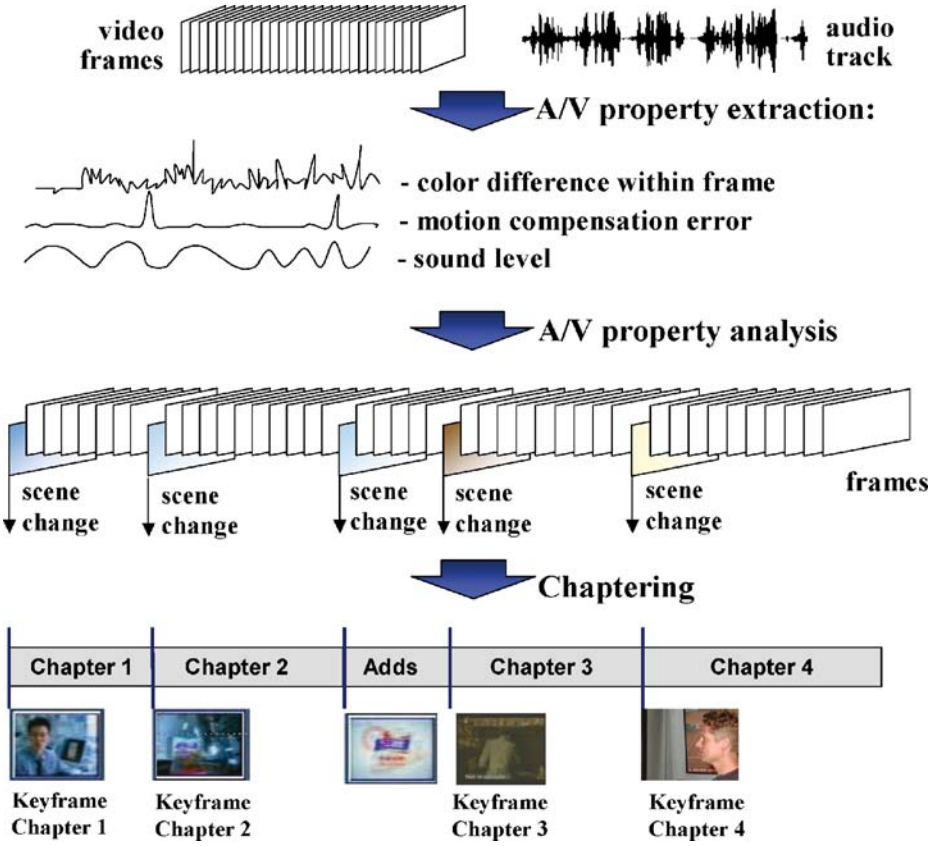


Fig. 8 Automatic chaptering of content

3 PC based feature-assessment approach

This section describes the PC based approach and service oriented architecture that enable early evaluation of *content analysis* features and applications.

3.1 Basics

When considering new product concepts based on *multimedia content analysis*, one should not concentrate on minimizing the hardware, CPU/memory requirements for each *content analysis feature*, but instead one should assess their usefulness in combination with other features. As a consequence, the assessment of product concepts in the early stages of design requires a powerful prototyping system with ample CPU and memory resources. Moreover, implementation and integration of new experimental algorithms should be easy and fast. Finally, the mapping of a selected set of algorithms to an embedded system must be straightforward and with minimal effort.

The prototyping system we use for the assessment of CE products with multimedia content-analysis features satisfies these requirements through powerful PCs, off-the-shelf PCI cards, standard PC development tools and networking technology. Content-analysis algorithms are modeled as black box components with standardized interfaces that are invariant across all platforms, which facilitates the mapping process. Only the algorithms need to be tuned for a specific hardware platform, while optimized implementations of interconnection technology are available for each platform, offering the same interfaces across all platforms.

3.2 Components and interfaces

As an example of this approach, consider a simple streaming application as depicted in Fig. 9a and its implementation in software (Fig. 9b). It shows three software components, i.e. *Encoder*, *AVDB* and *Decoder*, which are controlled via interfaces (*Control API*) by component *Control* and stream AV data to each other via channels offering streaming interfaces (*Stream API*). Moreover, Fig. 9b also shows that channels are implemented using shared memory, a common practice in embedded systems.

3.3 Early evaluation of features

For fast, early evaluation of new advanced experimental features, it is preferred to pack all features into components without any modification (as far as possible). This means that such features are typically hardly optimized, because the first priority now is to assess their functionality. Evidently at this point it is a waste of effort to optimize any features, before appreciating what they may have to offer. This means that in many cases a single PC is not enough to assess the combined functionality of a number of features. Depending on the number and type of features, even the most powerful multiprocessor PC may be incapable of doing so.

A solution is then to use multiple PCs, run CPU hungry components on different computers and have them communicate control and streaming data over the network. As an example, Fig. 10 shows a simple network of three PCs, where each runs one or two components of Fig. 9b in different process address spaces. Evidently, this is not possible without introducing additional entities that handle the networking of control and streaming data for each component. This is described in the next subsection.

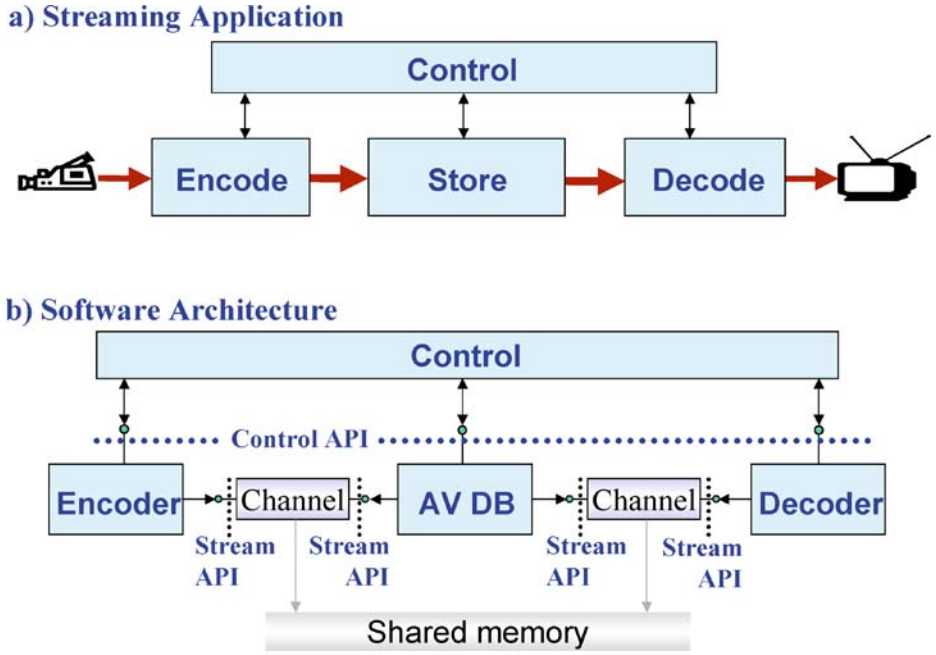


Fig. 9 Streaming application and corresponding SW architecture

3.4 Networked components

To enable control applications to find streaming components in the network, create streaming connections between them and allow components to communicate control and streaming data via the network, each component must be extended with networking functionality. Figure 11 shows this for the example of Fig. 10.

As shown in Fig. 11, each component is extended with a stub, which interfaces with the network. At the control side, a new entity is introduced, the *connection manager*. It creates a *proxy* for each component it discovers in the network. This is done with UPnP networking [17]. To this purpose, the connection manager functions as an UPnP control point, while each *component stub* is an *UPnP device*, announcing its presence in the network. As a result, the connection manager can create *component proxies*, and hand the associated interface pointers to the application.

Each *component proxy* implements exactly the same interface as the component itself. Moreover, once created by the *connection* manager, each *component proxy* only communicates with its associated *component stub* via the network using UPnP networking. The *component stub* translates the network commands into straight *control API* calls for the bare, non-networked component it is attached to. As a consequence, when optimization has taken place, and all components can run on the same computer within the same process address space, then all network proxies and stubs can be removed and the application can directly connect to the components.

Figures 9b and 11 also show the important interfaces for control and streaming. It is key that both the streaming and the control interfaces are well defined and widely used in embedded system architectures of consumer products, as to facilitate the mapping of PC prototypes to real product architectures. In our prototyping framework, all components and

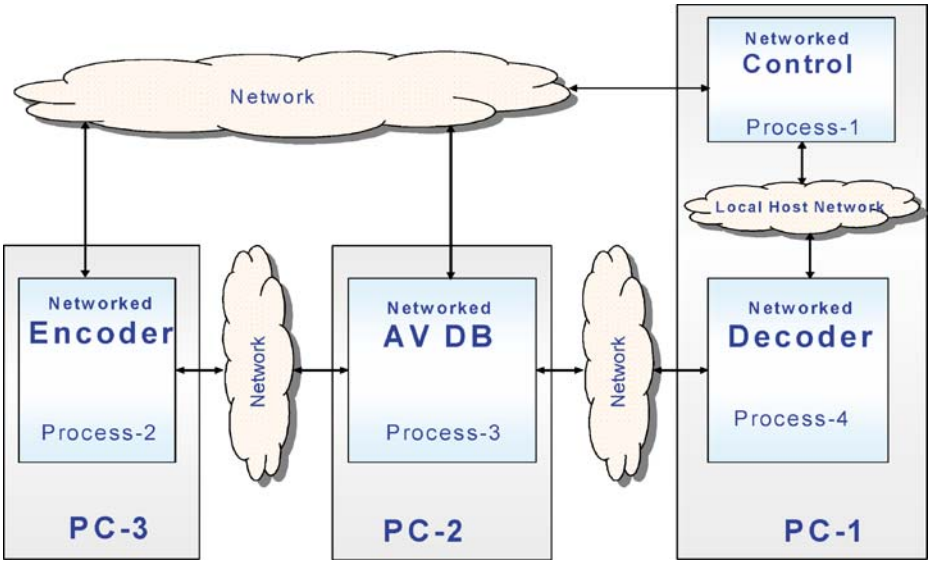


Fig. 10 Streaming system of Fig. 9, with networked enabled components, mapped to a three-PC network

interconnection technology adhere to the UHAPI [16] interface standard for controlling stream processing functionality and C-HEAP [14]/YAPI [3] as the interface for passing streaming data between signal processing components. All these interfaces must be preserved when implementing systems on a PC network, see Fig. 11.

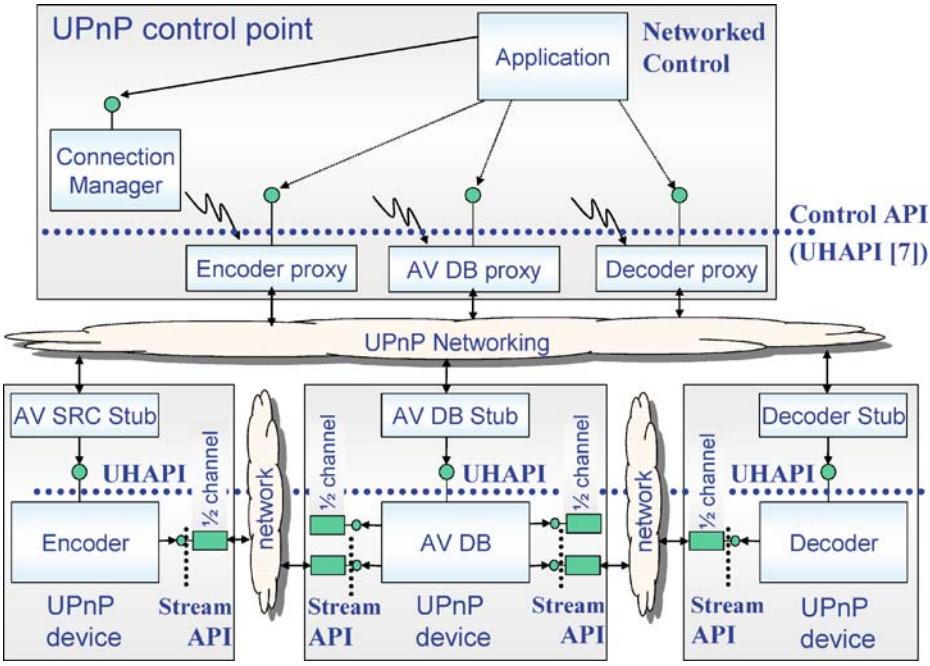


Fig. 11 Proxies, stubs and half channels to enable networking for applications and stream components

The *channel* component in Fig. 9b merely implements a streaming interface to shared memory, which is used in an embedded system to buffer the data between stream processing elements. For the networked case (Figs. 10, 11) the channel is split in two *half channels* to preserve the streaming interface. The buffering of streaming data can be done indirectly via some *memory server* in the network or directly by standard network buffers.

This approach enables adequate assessment and testing of new features and combinations of features before doing the actual product development. As a result, requirement specifications will be more accurate, feature interactions are better understood beforehand and resource requirements of features can be obtained by measurement (and can be minimized if necessary).

3.5 Advanced connection management

Based on experiences so far, the PC based prototyping approach is currently considerably being improved. Among others things, ease of use and faster integration of components is realized by more advanced *connection management*.

Previously, a particular combination of features was realized by specifying the connections between components in plain C or C++. For each feature combination, a different piece of code was required. Even worse, for live and seamless switching between different feature combinations, a separate piece of code was required to stop components, flush stream data, destroy connections and subsequently create new connections and resume components. For each transition between feature combinations, such a piece of code was required, which grows with the square of the number of feature combinations.

Currently, a new connection management approach is being introduced that enables easy specification of feature combinations, by means of a graphics editor and XML descriptions. Switching between these feature combinations has been automated. This is done by a small program that analyzes the network of components for the current and next feature combination. It then only stops components and connections that are no longer used in the new configuration, or temporarily pauses components for which a connection must be changed.

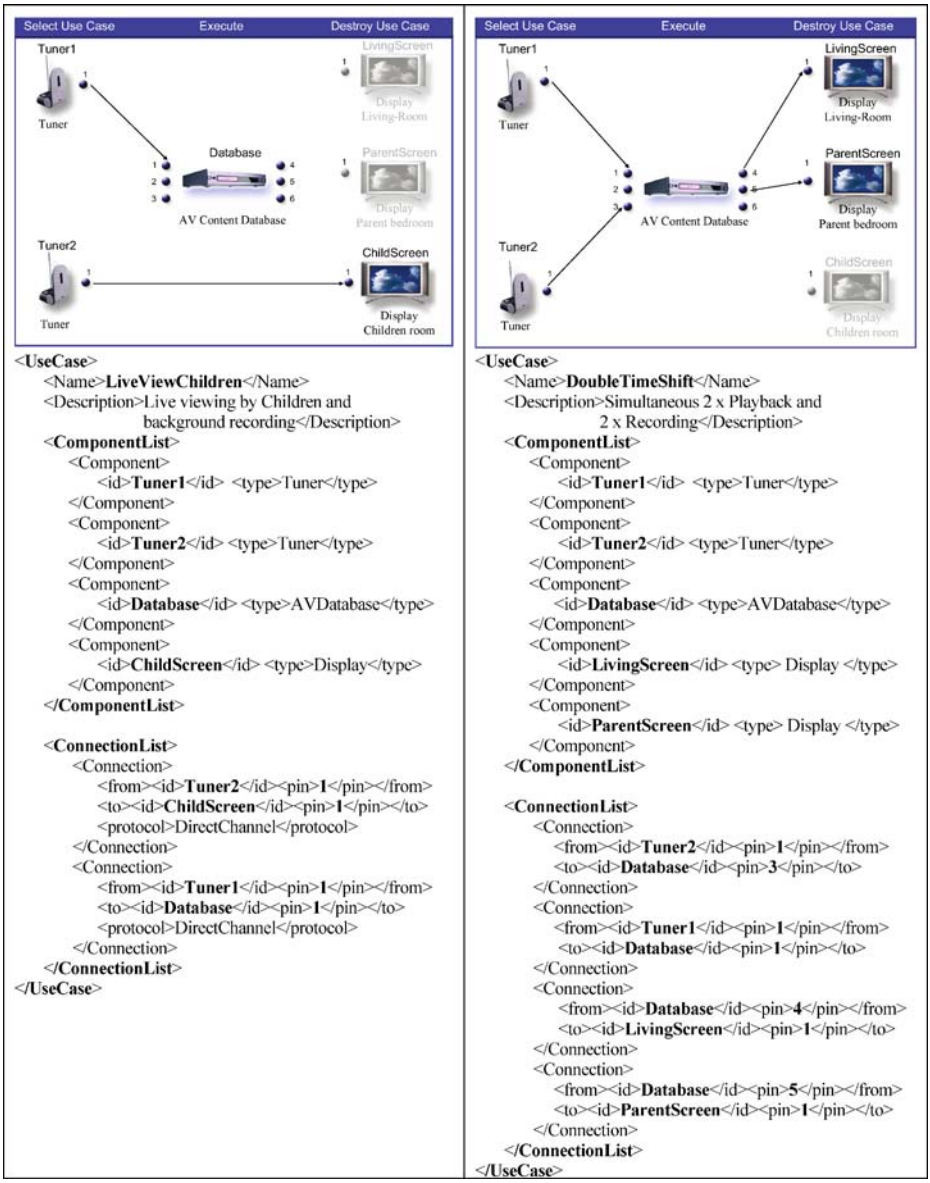
As an example, Fig. 12, shows two screenshots of a graphics editor/component network visualizer, illustrating two different feature combinations (using three or four components out of six components) between which seamless switching can be done automatically. Note that the components depicted can be simple to complex software and or hardware entities, either *embedded in a consumer electronics device* or *distributed in an in-home network*.

The configuration descriptions simply list for each use-case the components involved and the connections between them. Additional information is given on the type of the components and the communication protocol for connections. This information can be used for automatic selection of available and suitable components for complex use-cases.

Other work is being carried out to improve the software architecture and implementation of the platform, introducing robustness, load balancing, auto-restart for crashed components and more efficient network communication.

4 Demonstrator

This section describes a *content analysis demonstrator* system as we have built using the PC technology discussed in the previous section. It gives an overview of the basic features




```
<UseCase>
    <Name>LiveViewChildren</Name>
    <Description>Live viewing by Children and
                 background recording</Description>
    <ComponentList>
        <Component>
            <id>Tuner1</id> <type>Tuner</type>
        </Component>
        <Component>
            <id>Tuner2</id> <type>Tuner</type>
        </Component>
        <Component>
            <id>Database</id> <type>AVDatabase</type>
        </Component>
        <Component>
            <id>ChildScreen</id> <type>Display</type>
        </Component>
    </ComponentList>

    <ConnectionList>
        <Connection>
            <from><id>Tuner2</id><pin>1</pin></from>
            <to><id>ChildScreen</id><pin>1</pin></to>
            <protocol>DirectChannel</protocol>
        </Connection>
        <Connection>
            <from><id>Tuner1</id><pin>1</pin></from>
            <to><id>Database</id><pin>1</pin></to>
            <protocol>DirectChannel</protocol>
        </Connection>
    </ConnectionList>
</UseCase>
```

```
<UseCase>
    <Name>DoubleTimeShift</Name>
    <Description>Simultaneous 2 x Playback and
                 2 x Recording</Description>
    <ComponentList>
        <Component>
            <id>Tuner1</id> <type>Tuner</type>
        </Component>
        <Component>
            <id>Tuner2</id> <type>Tuner</type>
        </Component>
        <Component>
            <id>Database</id> <type>AVDatabase</type>
        </Component>
        <Component>
            <id>LivingScreen</id> <type> Display </type>
        </Component>
        <Component>
            <id>ParentScreen</id> <type> Display </type>
        </Component>
    </ComponentList>

    <ConnectionList>
        <Connection>
            <from><id>Tuner2</id><pin>1</pin></from>
            <to><id>Database</id><pin>3</pin></to>
        </Connection>
        <Connection>
            <from><id>Tuner1</id><pin>1</pin></from>
            <to><id>Database</id><pin>1</pin></to>
        </Connection>
        <Connection>
            <from><id>Database</id><pin>4</pin></from>
            <to><id>LivingScreen</id><pin>1</pin></to>
        </Connection>
        <Connection>
            <from><id>Database</id><pin>5</pin></from>
            <to><id>ParentScreen</id><pin>1</pin></to>
        </Connection>
    </ConnectionList>
</UseCase>
```

Fig. 12 Screen shots of network editor/visualizer for two different feature combinations and corresponding configuration descriptions in XML

and components, it presents the overall architecture and describes a number of interesting *contentanalysis* algorithms that have been integrated into this system.

4.1 Overview

The *content analysis demonstrator* is a system integrating more than 40 different content analysis algorithms, including all those described in Section 2, and run on six PCs that

Fig. 13 Part of system set-up: six LCD screens showing multi-media analysis results

concurrently stream data to one another via the network. These algorithms analyze audio/video signals in real-time; their output is displayed on different computer displays, see Figs. 13 and 14 and stored into a SQL database.

At the same time, the audio/video signal is stored in a real-time file system. Offline, further content analysis is done, e.g. to detect and analyze key-frames, and results are communicated with a graphics user interface.

Among other things, the GUI shows key-frames as thumbnail pictures on the screen, which a user can select to start a replay of the stored audio/video content from the position that corresponds to the thumbnail picture, see Fig. 7.

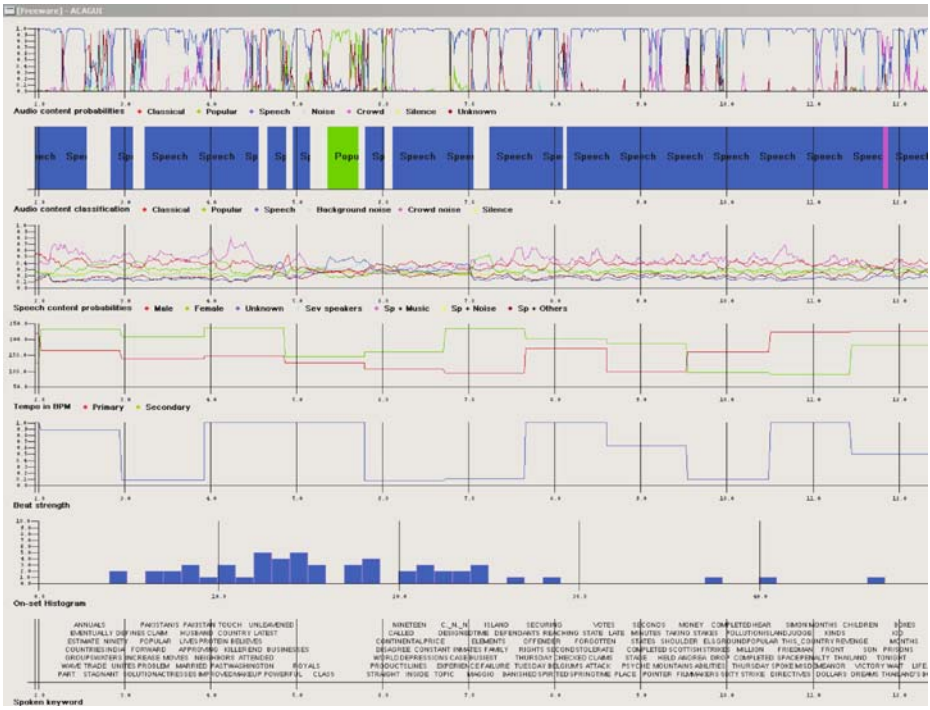


Fig. 14 Screenshot of LCD display showing extracted features in real-time

4.2 Architecture

The overall system architecture is depicted in Fig. 15. A digital signal is captured, decoded and re-encoded with a MPEG2 Codec, which extracts some low-level parameters out/of the video signal, e.g. average luminance and color per frame. The same thing is done for audio but no special hardware is used for this except for a simple audio capture card for PC.

The actual content analysis part, *Real Time Content Analysis* in Fig. 15, is quite complex: the complete set of real-time content analysis algorithms has been clustered into 11 streaming tasks and distributed over 6 PCs. They communicate with each other and with graph visualization tasks (*GUIxxx*) via stream channels. Figure 16 shows this part of the architecture in more detail. The most interesting parts are described in more detail below.

First, the AV stream is separated into an audio and a video stream. Audio is captured through standard PC audio capture cards on PCs that run the *audio content analysis components*. These are the *Automatic Speech Recognition* (*ASR*) and the *Audio Analysis Content Analysis* (*ACA*) [9]. The ASR extracts spoken words from the audio stream and stores them as ASCII into the meta-database via the *Metadata Writer* component. In parallel the *ACA* module classifies the audio content into classes such as speech, music, noise, silence and cheering [9]. The video part of the stream is the input for various Video/ Multimedia Content Analysis (*VCA*, *MCA*) modules such as the Film Mode discriminator (*FilmDetector*); it separates streams into *interlaced video* and *moving pictures* segments [2]. Another example of *VCA* is Face Detection (*FaceDetectDCT, FaceDetectAC*). These modules identify faces and their location on the screen [12]. Consecutively, the Face Recognizer (*FaceRecognition*) tries to find the matching ID, i.e. name, of the face instance by means of a biometrics database. Additionally, the Signature Recognition/Matching (*Signature Recognition*) matches extracted video signatures with a signature database to identify repeating content. To this purpose it uses low-level features that are extracted by the hardware MPEG encoder (*PNX*). An Overlay Text Detection (*OverlayTxtAC*) module identifies and localizes text instance, e.g. subtitles, in the video content. For more information on these algorithms see also Section 2.

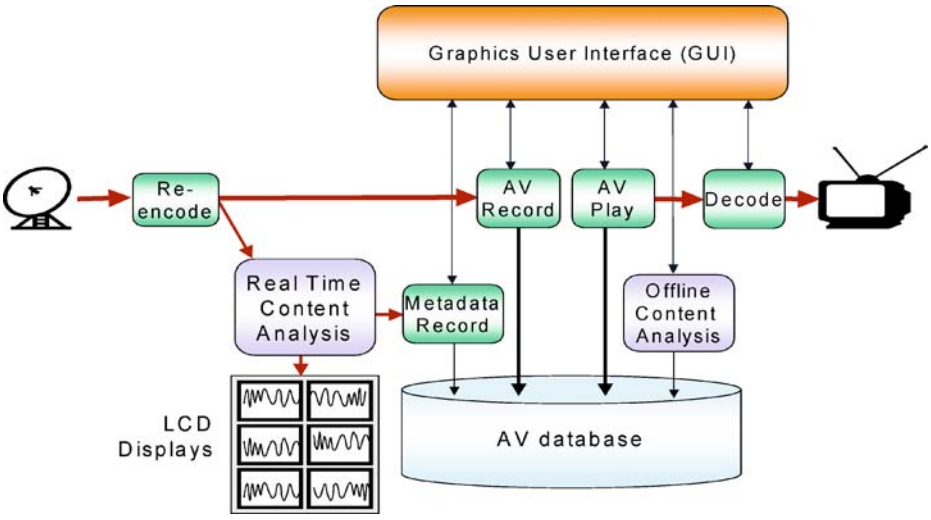


Fig. 15 Top level view of content analysis system

The central component, indicated by *PNX*, performs both low-level feature extraction and MPEG2 encoding (http://www.semiconductors.philips.com/products/nexperia/home/products/dvd_recording/pnx7100/). Moreover, it generates a time-code on video frame basis, which is fed to all content analysis components. They use the time-code to provide a time stamp for all generated features (*FrameRef_t*). This way all subsequent processing is able to display and store the extracted features in a synchronized way. The *MetaDataWriter* assures that all extracted metadata is synchronized before storing it into the database. An efficient synchronization mechanism was implemented to enable this [4].

Each component transmits the features it has extracted over a network channel to another component. Since each component generates unique metadata features, each pair of communicating components must *know* the type of the data being transmitted. To this purpose over 12 metadata data types were defined, see Fig. 16, enabling each component to correctly interpret any metadata received.

Although the stream processing components/tasks in the system stem from many different sources/development groups within Philips, they were easily integrated into one content analysis system by encapsulating them into a thin communication shell.

4.3 Measurements

A performance estimate was done for a number of selected algorithms that are candidate for realizing new features, e.g. *commercial detection*, in consumer storage products of the near future. These measurements were done by instrumenting the content analysis algorithms with *operation counters*, which count the number and type of operations, e.g. +, −, *, *if*, etc,

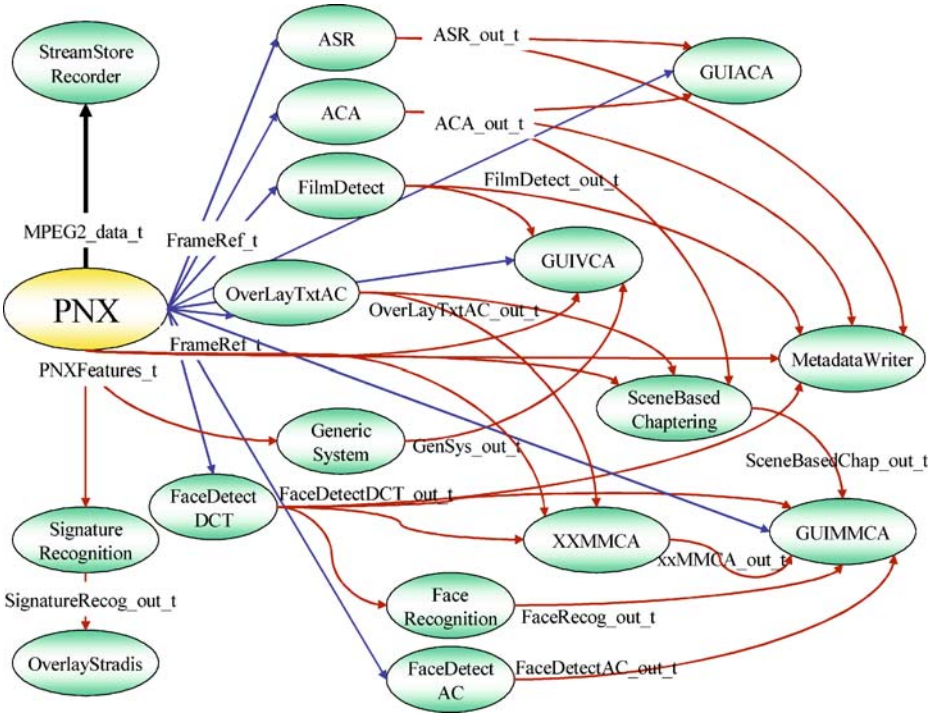


Fig. 16 AV content analysis tasks concurrently running on different PCs, communicating via the network

carried out in each algorithmic processing loop of the algorithms for each audio/video frame.

It turned out that performance requirements are quite low, i.e. <20 MOPS for the video processing algorithms. The reason is that most complexity is in the feature extraction, which is already done in special hardware blocks in the PNX7100 Video CODEC (http://www.semiconductors.philips.com/products/nexperia/home/products/dvd_recording/pnx7100/). Table 2 gives more detail on performance requirements for *scene based chaptering* and *commercial block detection* [5].

Note that other types of algorithms, such as face detection and face recognition require much more performance, i.e. 80% of all the CPU cycles on a 3 GHz Pentium IV processor, see [12].

4.4 Load balancing

Assigning components/algorithms to a specific PC is a manual process that is based on functional partitioning and performance requirements. For example, all the audio analysis algorithms are executed on one PC, except for speech recognition, since this is more performance demanding. The same is done for video; simple algorithms are clustered and run on a single PC, while others require the full processing power of a powerful PC to run in real time, e.g. face detection and recognition.

With new and changing algorithms, performance requirements are likely to change in the future. To this purpose, more flexible and automated load-balancing is desirable. This is a current topic of research.

5 Conclusions

To enable fast evaluation of *contentanalysis* systems, to come to sensible solutions that are easy to use, PC based prototyping is a must. This is extremely important to be able to understand the feasibility of future CE storage products that heavily depend on advanced content analysis features.

Table 2 Performance measurements for commercial block detection

Feature	Algorithms	Performance requirements
Scene based chaptering		2,600 OPS, 320 FLOPS
	Scene change detection	1,000 OPS, 200 FLOPS
	Key frame selection	1,600 OPS. 120 FLOPS
Commercial detection		5,060 OPS
	Audio-cut-silence detection	5,000 OPS
	Video monochrome frame detection	10 OPS
	Audio–video detection result combination and decision making	50 OPS

However, to prevent PC based system solutions being created that cannot be implemented on more resource constrained systems having a different software/hardware architecture, a number of boundary conditions must be met. Among other things, this can be achieved by adhering to standardized interfaces for control and streaming, such as UHAPI [16] and C-HEAP [14]/YAPI [3], by using interconnection technology that is designed for the platform at hand, and by optimizing the feature-implementation for each underlying HW/SW platform.

On the other hand, the approach presented can also be used to leverage the application of embedded system control/streaming interfaces, such as UHAPI and YAPI in the in-home network domain. This makes it easier to distribute or merge stream functionality either over or into an arbitrary number of networked in-home devices.

Experiences with the large-scale prototyping activities we have carried out at Philips Research, see e.g. [1], for the assessment of future *contentanalysis* systems, show that a PC based prototyping approach enables the integration of many different media processing features in a short time and that it allows for accurate analysis of the resource (CPU/memory) requirements of such components.

References

1. CASSANDRA project. http://www.research.philips.com/technologies/storage/cassandra
2. de Haan G (2000) Video processing for multimedia systems. In: ISBN: 90-9014015-8, Eindhoven
3. de Kock EA et al, YAPI: application modeling for signal processing systems. In: Proceedings of the 37th conference on Design automation
4. de Lange A, Snijder F, Method of and device for synchronizing multiple input streams. Patent application, PHAT050002 US
5. Dimitrova N, Jeannin S, Nesvadba J, McGee T, Agnihotri L, Mekenkamp G (2002) Real time commercial detection using MPEG features. In: 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2002, Annecy France July 1–5 2002
6. Hollemans G (2003) Meaningful navigation. http://www.research.philips.com/technologies/misc/homelab/downloads/homelab_365.pdf
7. Ma Q, Kondo H, Sumiya K et al (2001) Virtual TV channel filtering, merging and presenting internet broadcasting channels. SIGNotes Information Processing Society of Japan, Jun
8. Maxtor, Big Drives, Maxtor Technologies White Paper. http://www.maxtor.com/_files/maxtor/en_us/documentation/white_papers/big_drives_white_papers.pdf
9. McKinney M, Breebaart J (2003) Features for audio and music classification. In: 4th International Symposium on Music Information and Retrieval, Oct, Baltimore, Maryland
10. MultimediaN, Multimedia analysis, database technology, and human computer interaction. http://homepages.cwi.nl/~mk/multimedianonline/
11. Nesvadba J, Ernst F, Perhavc J, Benois-Pineau J, Primaux L (2005) Comparison of shot boundary detectors. In: Int. Conf. for Multimedia and Expo, Amsterdam, The Netherlands, June 6–8, 2005
12. Nesvadba J, Miguel Fonseca P, Kleihorst R, Broers H, Fan J (2004) Face related features in consumer electronic (CE) environments. In: IEEE SMC, Den Haag, The Netherlands
13. Nesvadba J, Louis N, Benois-Pineau J, Desainte-Catherine M, Klein Middelink M (2004) Low-level cross-media statistical approach for semantic partitioning of audio-visual content in a home multimedia environment. In: Proc. IEEE IWSSIP'04 (Int. Workshop on Systems, Signals and Image Processing), pp 235–238, Poznan, Poland, September 13–15, 2004
14. Nieuwland A, Kang J, Gangwal OP et al (2002) A heterogeneous multi-processor architecture and scalable and flexible protocol for the design of embedded signal processing systems. Des Autom Embed Syst 7(3):233–270 (Kluwer)
15. Sayed Hashimi, Service oriented architecture explained. http://www.ondotnet.com/pub/a/dotnet/2003/08/18/soa_explained.html
16. UHAPI, New application programming interface for the consumer electronics industry. http://www.uhapi.org
17. UPnP, The universal plug and play forum. http://www.upnp.org

Fons de Lange is an embedded system architect at Philips since 1991. He has worked and published on a variety of HW/SW architectures, IC design tools, applications and implementations, including multi window TV, high throughput digital video processors, software architecture for digital TV set-top boxes, prototyping of embedded system architectures with PC networks, and holds multiple US patent applications on these topics as well as numerous publications. Currently, Fons works as a senior research scientist in the Healthcare Systems Architecture group at Philips Research. His current research interests are in information technology for medical systems with a focus on generic software architectures for multi-modality medical imaging. Dr. Fons de Lange has a MS degree in Electrical Engineering, particularly on Computer Aided IC Design, as well as a PhD in computer science on design methods for parallel processors, both from Delft University of Technology.

Jan Nesvadba received his diploma in Electrical Engineering (telecommunications) from the Technical University of Vienna, Austria. He did his masters thesis in the field of electro-biology on di-electrophoresis of biological cells. He joined Philips Research 1998 and worked on a digital return channel for HFC-networks. He works as a researcher in the Storage Systems and Applications group at Philips Research. His current research interests include retrieval algorithms for video/audio content (baseband and compressed domain), the adaptation of existing IC's (encoders, codecs) for the generation of content descriptors and the optimal use of these descriptors in consumer storage devices and user interfaces.