REIHE INFORMATIK
34/95

The MoCA Workbench: Support for Creativity in Movie Content Analysis

Rainer Lienhart, Silvia Pfeiffer, and Wolfgang Effelsberg
Universität Mannheim
Praktische Informatik IV
L15,16
D-68131 Mannheim

The MoCA Workbench: Support for Creativity in Movie Content Analysis

Rainer Lienhart, Silvia Pfeiffer, and Wolfgang Effelsberg

Praktische Informatik IV

University of Mannheim, 68131 Mannheim, Germany

{lienhart, pfeiffer, effelsberg}@pi4.informatik.uni-mannheim.de

Abstract

Semantic access to the content of a video is highly desirable for multimedia content retrieval. Automatic extraction of semantics requires content analysis algorithms. Our MoCA (Movie Content Analysis) project provides an interactive workbench supporting the researcher in the development of new movie content analysis algorithms. The workbench offers data management facilities for large amounts of video/audio data and derived parameters. It also provides an easy-to-use interface for a free combination of basic operators into more sophisticated operators. We can combine results from video track and audio track analysis. The MoCA Workbench shields the researcher from technical details and provides advanced visualization capabilities, allowing attention to focus on the development of new algorithms. The paper presents the design and implementation of the MoCA Workbench and reports practical experience.

1 Introduction

Videos are a modern and attractive medium of information transfer between people, be it via videos in education, cinema movies or medical films. However, the information contained in a video is not easily accessible to computers; no computer can directly extract semantic information from a digitized video. In order to gain automatic access to the content of a movie, it is often indexed "manually", i.e. a separate description of the content is created which is in some way kept together with the movie. The sheer number of necessary annotations makes it impossible to index large amounts of video by hand. So algorithms for automatic or semi-automatic extraction of content information from videos (picture frames AND audio) are highly desirable (Figure 1).

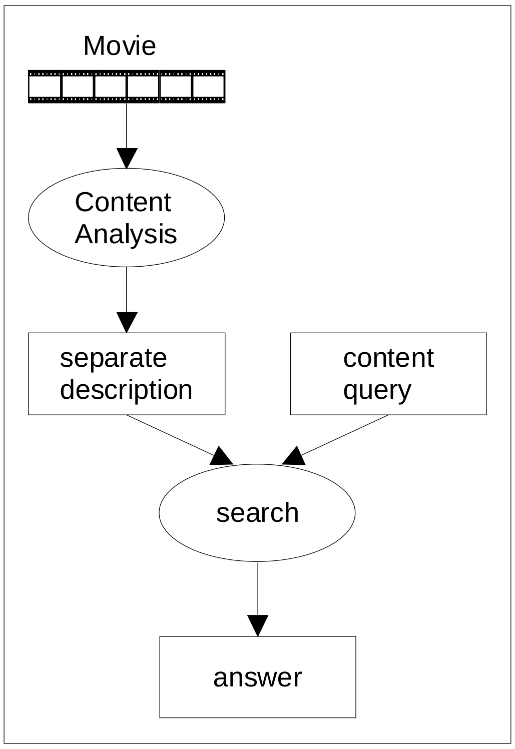


Figure 1: Computerized access to video contents.

The creation, implementation and evaluation of new algorithms for movie content analysis are the main goals of our MoCA (Movie Content Analysis) project. It is not yet fully understood how the human being derives content from something seen or heard; it is a continuing effort to model these processes and test them in algorithms. We have developed a content analysis workbench to support the invention, development, implementation and testing of such algorithms. It has proved handy in handling and combining the vast number of routines and intermediate data. By representing interim and final results graphically, and by allowing easy combination of procedures into new algorithms, our MoCA Workbench is a playground for creativity in developing new algorithms.

The rest of this paper is structured as follows. Section 2 presents key concepts of MoCA Workbench. Section 3 discusses the movie model, while Section 4 introduces the workbench's data types and video/audio operators. A description of the current status of the implementation follows in Section 5. Section 6 illustrates the use of our MoCA Workbench by example. Section 7 reviews related work, and Section 8 concludes with a summary and future research directions.

2 Key Concepts

The MoCA Workbench is a software environment for the development and fast prototyping of new audio and video content analysis algorithms. Work on movie content analysis algorithms has confronted us with a variety of technical problems, which the workbench has helped us to overcome. Its development has therefore been guided by our practical experience. Some key concepts are presented in the following.

2.1 Definition of a Large Set of Orthogonal Image, Video, and Audio Operators

There is no room for creativity if technical restrictions are too many and possibilities too few. Thus, a large set of basic algorithms is a prerequisite for the development of more elaborate image, video and audio algorithms.

For each basic analysis task, the set contains one or more algorithms. For example, if we want to detect cuts in a video, many different kinds of algorithms will do so [3][18][19], each having advantages and disadvantages. We do not want to restrict our working environment to the provision of one single algorithm for solving a certain problem and have developed MoCA such that the set of algorithms is, in principle, unrestricted. Each algorithm for analysis, processing or transformation of data is called an *operator*.

2.2 Combination of Operators

In addition to providing a set of pre-coded operators, the MoCA system gives the user the option to combine existing operators freely and easily to create new operators. There are several reasons behind this design decision. First, all basic operators are either on video or on audio, but we want to be able to combine information from *both channels* in order to derive new information, an example being the detection of a scene as a group of shots with a similar audio background. Second, we want to be able to combine several indicators from different basic and combined operators for detecting *higher semantics* in films, such as the classification of a movie into a certain genre [1]. Third, our experience shows that content understanding is only achievable if many different factors are taken into account; thus the composition of operators is a prerequisite for deriving meaningful content information.

2.3 Clear Presentation

Human beings are good at building new associations and deriving new ideas, but creativity depends to a large degree on the clear presentation of the objects of thought. If we apply this insight to a researcher working in the field of automatic movie content analysis, the demand for clarity in the presentation of derived values and content indicators is obvious. New relationships between data can be found more quickly. The researcher should also be able to control the original and derived data to be displayed, as well as the resolution/aggregation level.

To provide the user with a familiar screen layout, our approach adheres to the track concept: We think of a movie as initially two streams, represented by tracks: the picture track and the audio track, each shown on a time scale. Then, content analysis algorithms are executed on a selected time span. Their results generate new tracks; for example, one might show video cuts, another one recognized speakers. For each track, several display operators and display parameters can be selected. In addition, windows related to a track can be opened for a closer and more accurate look at the data. For instance, by opening a video window for the picture track, the video can be played back. If a video window is opened, e.g. for the Fourier Transform of the video, the corresponding Fast Fourier Transform images can be played back as a video.

2.4 Rapid Prototyping

New ideas should be able to be translated quickly into action. This hastens development and motivates the researcher to try out new ideas/operators. Rapid prototyping leads us to two requirements:

- calculations should be fast in order to keep the system interactive, and
- new operators should be able to be defined and edited at run-time, i.e. recompilation should be avoided.

These goals can be achieved by

- universal data types, because operators can then be combined more easily.

- storing already calculated results in a database to reduce processing time for new operators built upon them. For example, scene length statistics or audio frequency statistics can be used by several different higher-level operators.
- executing operators by an interpreter rather than by compiled modules. New operators can then be defined, and existing operators can be edited at runtime.

However, it is not always possible to provide fast, interactive response time for all operators. Object segmentation, for example, is computationally intensive. But in many cases we can compute parameter values at runtime.

2.5 Evaluation of the Output of New Operators

Visualizing the results of operators is often not enough. The question of how well an operator performs its task is also of major interest since no algorithm is perfect. For example, having developed a new scene break detection algorithm, one is interested in its quantitative performance, i.e. how many scene breaks are missed or falsely detected. The researcher may also want such errors to be marked on the time scale for easy locating and checking later. Our workbench incorporates qualitative and quantitative means for evaluating results of operators. The user generates reference streams for the automatically calculated results, which contain the correct information for comparison.

Having presented the key concepts of the MoCA Workbench, we now proceed to the technical details.

3 Digital Movie Model

The term *video* is not as clear as it seems to be. In daily life, video denotes any visual material stored on a video cassette. But in multimedia research, the term video often denotes motion pictures only. It is crucial for us to include audio into our analyses. Thus, we use the term "movie" and define it as follows: A digital *movie* is any digital video material, e.g. newscast, sportscast, feature film, cartoon or commercial containing both audio and video. Our work concentrates particularly on the combination of both sources of information.

As mentioned above, a movie in MoCA consists initially of two streams:

- a picture stream (stream of image frames) and
- an audio stream.

The *picture stream* is regarded as a stream of 3-dimensional data. The dimensions are the x-coordinate, the y-coordinate, and the color component (red, green, blue) of pixels consisting of byte element values ranging from 0 to 255 [11] [12]. The *audio stream* is defined as a 1-dimensional stream since there is always a vector of samples in relation to a video frame. We call that vector of samples *audio frame*. Both frame types, audio and video, are our basic data units in MoCA. We thus interpret a movie as a series of video and audio frames, dividing the raw movie into consecutive time intervals lasting 1/25 sec for PAL and 1/30 sec for NTSC. This time interval is the basic time unit for data types, operators, and the results in the databases.

The picture and audio streams together are considered the *raw movie*. From them we extract and derive higher-level information, stored then in additional streams. The results of all executed operators are stored in a database, called "calculated database" (Figure 2). There exists also a second database, called "reference database" (Figure 2). This database is used for the assessment of the operators. Its data can be compared automatically with the results in the calculated database, enabling quality control and improvement of the algorithms. The comparison displays the percentage and absolute number of correspondence, false hits and missed hits and/or the reference values in a track next to the evaluated track providing the user with fast feedback about the quality of his operator (see Figure 11 for an example). All these tracks together comprise the *extended movie* (Figure 2).

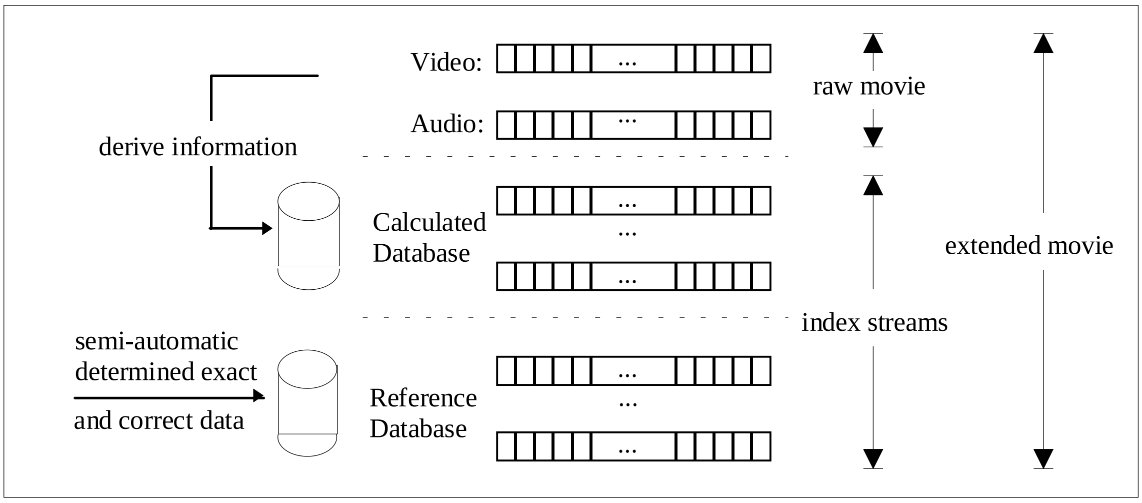


Figure 2: The digital movie model.

The human user interface of the MoCA Workbench is laid out according to our movie model: Initially, we display a time scale, where a time unit is a video frame. Scaling to multiples is possible. Via this time scale, a contiguous group of frames can be selected on which certain operators are executed. If the operator results are chosen for display, additional tracks are opened below the time scale (Figure 3).

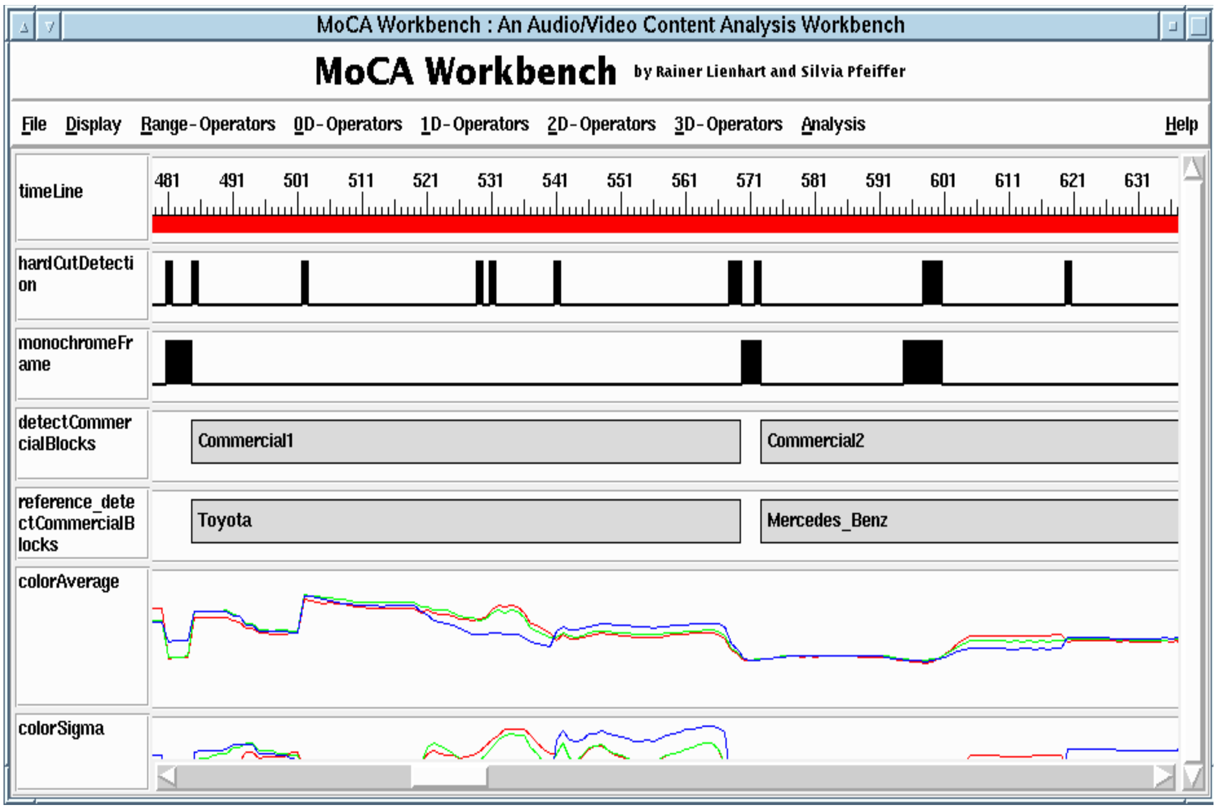


Figure 3: Human user interface of the MoCA Workbench.

4 Data Model and Database

4.1 Data types

The data model is a crucial element of MoCA. As mentioned above, the data types should be as universal as possible. Restriction to a certain kind of data would limit the combination of operators due to type incompatibilities. Consequently, the MoCA Workbench distinguishes only five general data types:

- range data,
- scalar values,
- vector values,
- matrix values, and
- three-dimensional data.

They are called *RANGE*, *0D*, *1D*, *2D*, and *3D* data (Figure 4). The *RANGE* data is a list containing a string and two numbers (elements). This data type is often used to express properties of frame sequences. Figure 5 gives an example. The numbers specify the range by the start and stop frame number of the parameter or property described by the string.

Data type	Description	generally refers to	for displaying refers to
RANGE	list of {string no. no.}	frame range	frame range
0D	scalar	frame or frame range	frame
1D	vector	frame or frame range	frame
2D	matrix	frame or frame range	frame
3D	cube	frame or frame range	frame

Figure 4: Data types in MoCA.

The *0D*, *1D*, *2D*, and *3D* data are arrays with 0, 1, 2, and 3 dimensions, respectively. A single element belongs to one of the classical data types bit, byte, long, float or double. However, for operators there is no difference e.g., between 2D data consisting of byte values and 2D data consisting of double values. Each operator implicitly converts the elements to the suitable format if necessary. Thus, operators only differentiate data with regard to dimension but without regard to its elements' data type (double, float etc.). Also, the operators decide themselves if the data is to be interpreted as belonging either to a single frame or to a frame range.

For display and database access only, data is classified with regard to its appearance in the time span of one frame of a movie (Figure 4). This time span, as already explained, is the atomic unit in MoCA. Therefore, a scalar associates a single value with each frame or frame range. An example is the average of the luminance ("brightness") of all pixels in a frame. 1D data associate a vector with each frame or frame range, such as a set of audio samples, its frequency distribution or the average luminance separately calculated for each color component (RGB). A grayscale picture is an example of a 2D value, where there is a brightness value for each pixel of a frame. Similarly, a color picture in RGB representation or a multi-spectral image is regarded as 3D data, because there are several values to each pixel. Range values describe information such as the start and stop points of scene transitions, i.e. with the Range data type, we can associate a single value with a group of frames (Figure 5).

fadeFromBlack	1345	1367
fadeToBlack	1629	1654
commercials	4030	6032

Figure 5: Example of RANGE data.

For convenience, MoCA also has a "sixth" data type called *xD*. Many operators such as the Fast Fourier Transformation operator can operate on xD data with $x \in \{0,1,2,3\}$. Thus, the *xD* data type is not really a new data type, but just indicates that the data can be of any dimension, i.e. the data type can be 0D, 1D, 2D or 3D, supporting semantic overloading to an even higher degree.

4.2 Operators

The above definition of data types allows an *operator* to be defined as a function executed on data of one or several types and producing data of a single data type (Figure 6):

$$f(X_1, ..., X_n) \longrightarrow Y \quad \text{with} \quad \begin{array}{l} X_1, ..., X_n \subseteq A, \\ Y \subseteq A, \\ A \in \{\text{Range, 0D, 1D, 2D, 3D, xD}\}, \text{ and} \\ n \in N \end{array}$$

Figure 6: Definition of data operators.

Operators may be *stateless* (i.e. they cannot reuse results of a previous execution) or *stateful* (i.e. they store their previous result(s)). A typical stateless operator is the edge detection calculation for a frame. It does not depend on the preceding or succeeding frames or on any other values except the input image. As a rule, all operators should be designed as stateless. However, using only stateless operators sometimes leads to difficult and/or inefficient implementations. For instance, a stateless optical flow computation operator would be quite inefficient if calculated frame-by-frame on contiguous frames. For each frame calculation a number of preceding frames (e.g. 10) must be processed. Repeating this calculation every time would be quite time consuming. Using existing state information, the optical flow algorithm can be rewritten as a stateful operator so that only the current frame is new input, and the output is computed by using the results already available for the preceding frames. This algorithm is faster by an order of ten. By letting the operators work on frame ranges and return a result for that range stateful operators can be avoided. They are only used due to rapid prototyping and flexibility.

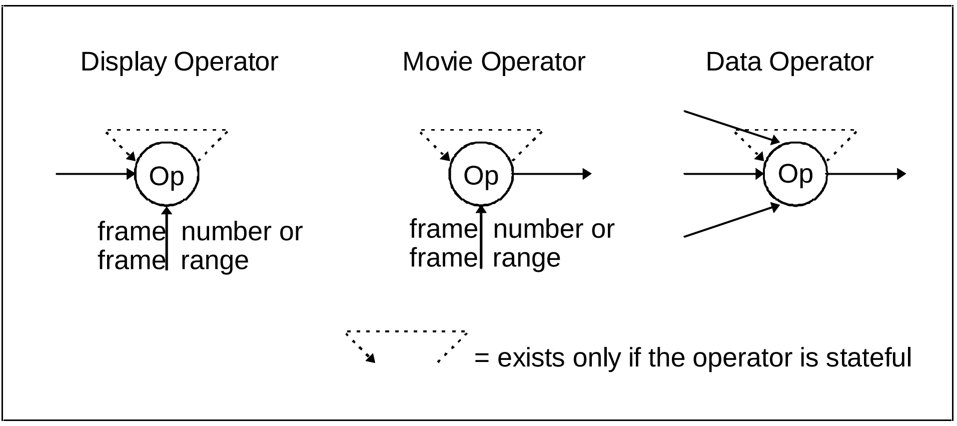

Display Operator
Movie Operator
Data Operator
Op
Op
Op
frame number or frame range
frame number or frame range
= exists only if the operator is stateful

Figure 7: Operators in MoCA.

In MoCA, we distinguish three types of operators according to the data flow:

- data operators,
- movie operators, and
- display operators.

Data operators transform input data into output data (Figure 6, Figure 7). Dependent on the input values and - possibly - on the operator's state (caused by previous executions), output values are calculated. It is the paradigm of data operators that they take existing information, apply some (conditional) processing to it and output a result. An example is the "detect silence" operator: audio data is taken as input (in audio frame sizes), then calculated, and a scalar value for the specified frame is returned. Data operators are the most flexible operators in MoCA and can be used to construct other operators. Examples include color histogram, color average and standard deviation, Fast Fourier Transform, edge detection, optical flow calculation, directional filters, unary and two-place pixel arithmetic, etc. Figure 8 shows the grayscale histogram as an example of a basic data operator. There is no need to specify the frame number, since the operator receives a frame as its input.

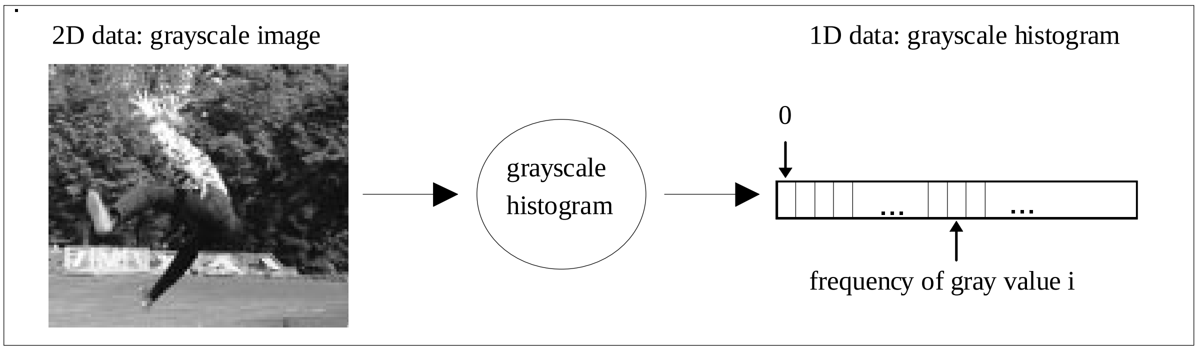


Figure 8: The grayscale histogram: A basic data operator

Movie operators are more specialized and are used to calculate information about movies. They are characterized by the omission of all input data except the frame number or the frame range for which they should calculate the information (Figure 7, Figure 9).

$$\begin{array}{llll} f(X) & \longrightarrow Y & \text{with} & X = \{\text{startFrameNo}, ..., \text{stopFrameNo}\}, \\ & & & Y \subseteq A, \text{ and} \\ & & & A \in \{\text{Range}, 0D, 1D, 2D, 3D, xD\} \end{array}$$

or

$$\begin{array}{llll} f(X_1, X_2) & \longrightarrow Y & \text{with} & X_1, X_2 = \{\text{startFrameNo}, ..., \text{stopFrameNo}\}, \\ & & & x_1 \leq x_2, x_1 \in X_1, x_2 \in X_2, \\ & & & Y \subseteq A, \text{ and} \\ & & & A \in \{\text{Range}, 0D, 1D, 2D, 3D, xD\} \end{array}$$

Figure 9: Definition of movie operators.

Since the movie operators know which information they should calculate for which frame number or frame range, they automatically request the required picture frames, audio frames and results from other movie operators. For instance, the difference between color histograms of subsequent frames can be used to develop a cut detection operator. If the absolute value of difference is above a certain threshold, a cut is detected [19]. Since the input data necessary for such a cut detection operator are clear, the movie operator needs only the frame number as input. It automatically calls the color histogram operator (another movie operator) for the required frames, calculates the difference, thresholds it and returns the result.

Movie operators operate on movies or movie pieces while data operators transform data. Therefore, movie operators are used to archive useful information extraction algorithms.

Display operators take input data and display them on the screen. They are used to visualize video data and results from movie operators. Thus, for data of every data type there must be a display possibility. As there are usually different ways to do this, there exist plenty of visualization possibilities for each of the five data types, especially the higher-dimensional ones. Data will usually be displayed in a track below the time scale, so that the temporal relationship between the video and audio frames and the data can be seen directly. For some data this is not applicable; these will be displayed in a separate window, e.g. a video abstract from a feature film will play in a separate video window.

5 Implementation

The MoCA Workbench was implemented on a SUN SPARCstation 5 under Solaris 2.4 using C++ and Tcl/Tk 7.4/4.0 [9]. The implementation comprises about 4000 lines of Tcl/Tk code and 4600 lines of C++ code. For the MoCA Workbench a new Tcl interpreter was created, extended by our image and video processing algorithms [1], the Vista library 1.3 [11] [12], the DEC AudioFile V3.0 [4] and a JPEG image loader. The movies were recorded from German television, digitized by a parallax video card, and stored as M-JPEG [10]. Currently, we are developing new fast 2D and 3D visualization widgets for our Tcl/Tk interpreter. Moreover, the base of available video/audio operators is increasing continuously. The operators are first implemented in Tcl as described in Section 4.2. Once an operator is stable, i.e. due to good results needs no further modification, it is re-implemented in C++ and added to the MoCA interpreter in order to speed up calculation. The MoCA Workbench has been used successfully in the movie genre recognition project [1].

6 "One Day with MoCA"

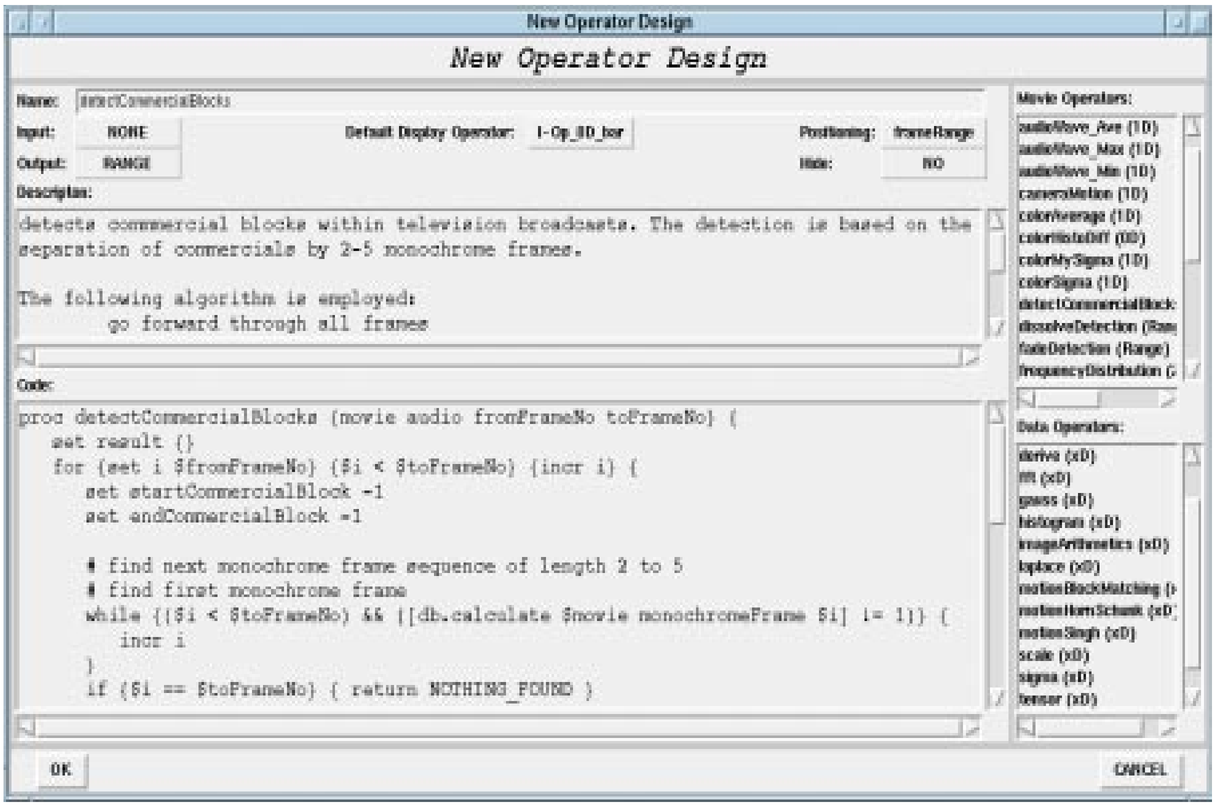
Example

In this section we cover a complete example, step-by-step. We show how a new content analysis algorithm is coded, executed, and evaluated, as well as how its derived data are displayed. The example demonstrates how the key concepts were realized in MoCA and how MoCA facilitates the development of new movie content analysis algorithms in practice.

Our example demonstrates the development of an operator to automatically detect blocks of commercials in TV broadcasts. In German TV, the program is periodically interrupted by a commercial block, i.e. a sequence of several commercials. Each commercial within a commercial block is separated by 2 to 5 (dark) monochrome frames. A commercial lasts between 5 and 160 seconds. The idea is to recognize commercial blocks based on these characteristics [1], leading to the following primitive commercial detection algorithm:

```
go forward through all frames {
    set startCommercialBlock = -1
    set endCommercialBlock = -1
    If (find next monochrome frame sequence of length 2 to 5) {
        set numberOfCommercials = 0
        set startCommercialBlock = current frame number
        while (find next monochrome frame sequence of length 2 to 5
            not further than 160 seconds ahead) {
            /* commercial block identified */
            set endCommercialBlock = current frame number
            set numberOfCommercials += 1
        }
        output "Commercial block from startCommercialBlock to endCommercialBlock"
    }
}
```

This primitive commercial detection algorithm can be coded easily and quickly with MoCA by invoking the "Create New Operator" Menu item and typing in the code shown in Figure 10. After a click on the OK button, the new "detectCommercialBlocks" algorithm is defined and appended to the list of RANGE movie operators. It can be executed immediately by selecting it on the "RANGE movie operator" Menu.



New Operator Design

Name: detectCommercialBlocks

Input: NONE

Output: RANGE

Default Display Operator: I-Op_0D_bar

Positioning: frameRange

Hide: NO

Description:

```
detects commercial blocks within television broadcasts. The detection is based on the
separation of commercials by 2-5 monochrome frames.

The following algorithm is employed:
        go forward through all frames
```

Code:

```
proc detectCommercialBlocks (movie audio fromFrameNo toFrameNo) {
   set result {}
   for {set i $fromFrameNo} {$i < $toFrameNo} {incr i} {
      set startCommercialBlock -1
      set endCommercialBlock -1

      # find next monochrome frame sequence of length 2 to 5
      # find first monochrome frame
      while {($i < $toFrameNo) && ([db.calculate $movie monochromeFrame $i] != 1)} {
         incr i
      }
      if {$i == $toFrameNo} { return NOTHING_FOUND }
```

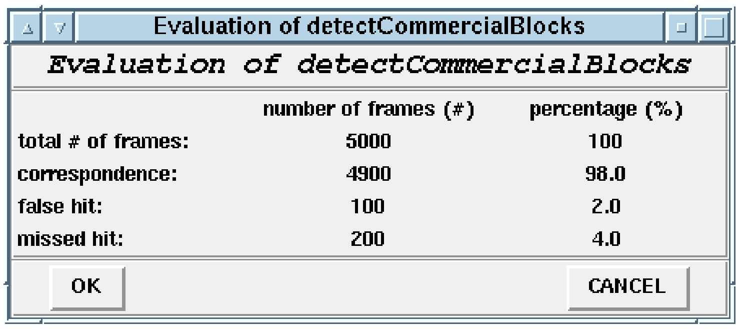
Figure 10: Rapid prototyping of the algorithm "detectCommercialBlocks".

Visualizing the result of the new operator

Results are visualized using the selected display operator for RANGE data. One possibility is shown in Figure 3.

Evaluation of the new operator

To evaluate the newly developed algorithm, the correct values (i.e. where the commercial blocks really are) have to be determined manually. These reference values are stored in the reference database. It is a laborious and tedious task to determine the correct range values and feed them into MoCA manually, despite the computer support provided. But once entered, the data are archived and can be used to evaluate different algorithms for the same content extraction. Thus, different algorithms for the same job can be compared easily. Figure 11 gives an example of the final result.



Evaluation of detectCommercialBlocks

	number of frames (#)	percentage (%)
total # of frames:	5000	100
correspondence:	4900	98.0
false hit:	100	2.0
missed hit:	200	4.0

Figure 11: Evaluation result of the primitive commercial block detection algorithm.

7 Related Work

The MoCA Workbench is something new and does not fall into existing categories. However, related work was performed earlier. On the one hand, there exist several automatic content analysis systems [5] [6] [19] [20] [21] [15]. These all concentrate on the calculation and employment of the research group's latest research results and serve a special purpose such as automatic newscast or soccer parsing, video retrieval and so on. Unlike MoCA none of them concentrates on the support of the researcher during the development of new algorithms.

On the other hand, numerous application development systems for computer vision and image processing employ
well-known algorithms to facilitate development of efficient, fast, reusable and/or portable applications [2] [14] [17]. A similar situation is found in the audio environment: there is a big collection of audio processing software packages, most of which originated in the speech processing environment [13] [22]. These packages usually have some kind of interface to view audio data in the time and frequency domain while offering little support for visualizing extracted audio content.

MoCA is not only intended as a means of plugging existing modules together to solve a special task, but also as a workbench for discovering new associations in the huge amount of data. Most other systems concentrate on image processing and have either limited video/audio combination operations [14] or none at all [12].

8 Conclusion and Outlook

Cognitive science research addresses what is going on in the human mind. H. Christopher Longuet-Higgins writes in his book on "Mental Processes: Studies in Cognitive Science" [7]: "...in order to understand our mental and perceptual processes in detail, we may find it most profitable to try and make working models of them. By working models I do not mean just electronic devices, but carefully designed programs which express what we think may happen when we look, listen, speak, or act." He draws an analogy between "...the programs we write and the mental processes they are meant to simulate." Our MoCA workbench supports the user in testing his "working models" of mental and perceptual processes concerning audio and video.

In managing data (movie data, reference data and derived data) and operators (algorithms management), the MoCA workbench releaves the user from the technical aspects, thus freeing his mental power for creativity in developing new operators. It also provides homogeneous access to the different operators. Moreover, it affords active research support by offering many different visualization possibilities for data, through the availability of fast prototyping due to the use of an interpreter, and by providing orthogonal and algorithmic combination of operators and interim results.

We have introduced the MoCA Workbench into our MoCA Project Research Group. Preliminary experiences are thus far very promising. Still, a lot of work remains to be done. Future plans for the workbench include:

- more parametric display operators which adapt immediately to changing display options like scaling, coloring, view angle, etc.,
- automatic execution optimization for operator groups which are chosen for calculation, and
- dealing with several movies at once in order to build operators for a group of movies.

References

[1] Stefan Fischer, Rainer Lienhart, and Wolfgang Effelsberg; Automatic Recognition of Film Genres. *Proc. ACM Multimedia 95*, San Francisco, CA, Nov. 1995, pp. 295-304.

[2] Myron Flicker, Mark Lavin, and Sujata Das; An Object-oriented Language for Image and Vision Execution (OLIVE); IEEE ???, 1990.

[3] Arun Hampapur, Ramesh Jain, and Terry Weymouth. Production model based digital video segmentation. *Journal of Multimedia Tools and Applications*, Vol 1, No. 1, pp. 1-38, March 1995.

[4] Thomas. M. Levergood, Andrew. C. Payne, James. Gettys, G. Winfried. Treese, and Lawrence C. Stewart. AudioFile: A Network-Transparent System for Distributed Audio Applications. *Proceedings of USENIX Technical Conference*, Summer 1993, pp. 219-236.

[5] Christopher J. Lindblad, David J. Wetherall, and William Stasior. ViewStation Applicatons: Implications for Network Traffic. *IEEE Journal on Selected Areas in Communications*, Vol. 13, 1995.

[6] Christopher J. Lindblad, David J. Wetherall, and David L. Tennenhouse. The VuSystem: A Programming System for Visual Processing of Digital Video. *Proc. ACM Multimedia 94*, San Francisco, CA, Oct. 1994, pp. 307-314.

[7] H. Christopher Longuet-Higgins "In Our Image?" from H.C. Longuet-Higgins "Mental Processes: Studies in Cognitive Science" The MIT Press 1987

[8] James Matthews, Peter Gloor, and Fillia Makedon. VideoScheme: A Programmable Video Editing System for Automation and Media Recognition. *Proc. ACM Multimedia 94*, Annaheim, CA, 1993, pp. 419-426.

[9] J. Ousterhout; Tcl and the Tk Toolkit; Addison-Wesley Publishing Company, Inc; 1994.

[10] William B. Pennebraker and Joan L. Mitchell. JPEG Still Image Data Compression Standard. Van Nos Rheinhold, New York, 1993.

[11] Arthur R. Pope, Daniel Ko, David G. Lowe. Introduction to Vista ProgrammingTools. Department of Computer Science, University of British Columbia, Vancouver.

[12] Arthur R. Pope and David G. Lowe. Vista: A Software Environment for Computer Vision Research. Department of Computer Science, University of British Columbia, Vancouver.

[13] C. Read, E. Buder, and R. Kent. Speech Analysis Systems: An Evaluation. Journal of Speech and Hearing Research, April 1992, pp. 314-332.

[14] Rasure and Kubica. The Khoros Application Development Environment. *Experimental Environments for Computer Vision and Image Processing*, editor H. I. Christensen and J. L. Crowley, World Scientific 1994.

[15] Takashi Satou and Masao Sakauchi. Video Acquisition on Live Hypermedia. *IEEE ICMCS*, Mai 1995, pp. 175-181.

[16] Michael Shantzis. A Model for Efficient and Flexible Image Computing. *SIGGRAPH ‘94 Proceedings*, pp. 147-154

[17] Jonathan Swartz and Brian C. Smith. A Resolution Independent Video Language. *Proc. ACM Multimedia 95*, San Francisco, CA, Nov. 1995, pp.179-188.

[18] Ramin Zabih, Justin Miller, and Kevin Mai. A Feature-Based Algorithm for Detecting and Classifying Scene Breaks. *Proc. ACM Multimedia 95*, San Francisco, CA, Nov. 1995, pp. 189-200.

[19] HongJiang Zhang, Atreyi Kankanhalli, and Stephen W. Smoliar. Automatic Partitioning of Full-Motion Video. *Multimedia Systems*, Vol. 1, No. 1, pp. 10-28, 1993.

[20] HongJiang Zhang, Yihong Gong, Stephen W. Smoliar, Shuang Yeo Tan. Automatic Parsing of News Video, *Proc. IEEE Conf. on Multimedia Computing and Systems*, 1994.

[21] HongJiang Zhang and Stephen W. Smoliar. Developing Power Tools for Video Indexing and Retrieval, *Proc. SPIE Conf. on Storage and Retrieval for Image and Video Databases*, San Jose, CA, 1994.

[22] Speech Tools User Manual. Center for Spoken Language Understanding. Oregon Graduate Institute of Science and Technology, August 1993.