# MPEG-4 Authoring Tool using Moving Object Segmentation and Tracking in Video Shots

Petros Daras, Ioannis Kompatsiaris, Ilias Grinias, Georgios Akrivas, Georgios Tziritas, Stefanos Kolias, and Michael G. Strintzis*

CORRESPONDING AUTHOR:
Prof. Michael G. Strintzis,
Informatics and Telematics Institute,
1st Km. Thermi-Panorama Road,
57001 Thermi-Thessaloniki, Greece
Tel. :+30310.996351, Fax : +30310.996342,
Email: strintzi@eng.auth.gr

**Abstract**

In this paper an authoring tool for the MPEG-4 multimedia standard integrated with image sequence analysis algorithms is described. Bringing much new functionality, MPEG-4 offers numerous capabilities and is expected to be the future standard for multimedia applications. However, the implementation of these capabilities requires a complex authoring process, employing many different competencies from image sequence analysis and encoding of Audio/visual/BIFS to the implementation of different delivery scenarios: local access on CD/DVD-ROM, Internet or broadcast. As multimedia system history teaches, however powerful the technologies underlying multimedia computing, the success of these systems depends on their ease of authoring. In this paper a novel authoring tool fully exploiting the object-based coding and 3D synthetic functionalities of the MPEG-4 standard is described. It is based upon an open and modular architecture able to progress with MPEG-4 versions and it is easily adaptable to newly emerging better and higher-level authoring and image sequence analysis features. The authoring tool is available for download from our web site:

http://uranus.ee.auth.gr/pened99/Demos/ Authoring_Tool/authoring_tool.html

**Keywords: MPEG-4, Authoring Tools, Image sequence analysis**

## I. Introduction

MPEG-4 is the next generation compression standard following MPEG-1 and MPEG-2. Whereas the former two MPEG standards dealt with coding of general audio and video streams, MPEG-4 specifies a standard mechanism for coding of audio-visual objects. MPEG-4 builds on the proven success of three fields [1], [2], [3]: digital television, interactive graphics applications (synthetic content) and interactive multimedia (World Wide Web, distribution of and access to content). Apart from natural objects, MPEG-4 also allows coding of two-dimensional and three-dimensional, synthetic and hybrid,

This work was supported by the PENED99 project of the Greek Secretariat of Research and Technology and from the P2People EC IST project. P. Daras, I. Kompatsiaris and M. G. Strintzis are with the Informatics and Telematics Institute, Thessaloniki, Greece. I. Grinias and G. Tziritas are with the Computer Science Dept, University of Crete, Greece. G. Akrivas and S. Kolias are with the National Technical University of Greece, Athens, Greece.

audio and visual objects. Coding of objects enables content-based interactivity and scalability [4]. It also improves coding and reusability of content (Figure 1).

Far from the past "simplicity" of MPEG-2 one-video-plus-2-audio-streams, MPEG-4 allows the content creator to compose scenes combining, spatially and temporally, large numbers of objects of many different types: rectangular video, arbitrarily shaped video, still image, speech synthesis, voice, music, text, 2D graphics, 3D, and more. However, the implementation of these capabilities requires a complex authoring process, employing many different competencies from image sequence analysis and encoding of Audio/visual/BIFS to the implementation of different delivery scenarios: local access on CD/DVD-ROM, Internet or broadcast. As multimedia system history teaches, however powerful the technologies underlying multimedia computing, the success of these systems ultimately depends on their ease of authoring.

In [5] the most well-known MPEG-4 authoring tool (MPEG-Pro) was presented. This includes a graphical user interface, BIFS update and a timeline but it can only handle 2D scenes and it is not integrated with any image sequence analysis algorithms. In [6] an MPEG-4 compliant authoring tool was presented, which, however, is capable only for the composition of 2D scenes. In other articles [7], [8], [9], [10], MPEG-4 related algorithms are presented for the segmentation and generation of Video Objects which, however, do not provide a complete MPEG-4 authoring suite. Commercial multimedia authoring tools such as IBM Hotmedia and Veon [11], [12] are based on their proprietary formats rather than widely acceptable standards. Other commercial solutions based on MPEG-4 like application suites with authoring, server and client capabilities from iVAST and Envivio, are still under development [13], [14]. In [15], [16], an authoring tool with 3D functionalities was presented but it didn't include any support for image sequence analysis procedures.

Although the MPEG-4 standard and powerful MPEG-4 compliant authoring tools will provide the needed functionalities in order to compose, manipulate and transmit the "object-based" information, the production of these objects is out of the scope of the standards and is left to the content developer. Thus, the success of any object-based authoring, coding and presentation approach depends largely on the segmentation of the scene based on its image contents. Usually, segmentation of image sequences is a two step process; first scene detection is performed, followed by moving object segmentation and tracking.

Scene detection can be considered as the first stage of a non-sequential (hierarchical) video repre-

sentation [17]. This is due to the fact that a scene corresponds to a continuous action captured by a single camera. Therefore, application of a scene detection algorithm will partition the video into "meaningful" video segments. Scene detection is useful for coding purposes, since different coding approaches can be used according to the shot content. For this reason, scene detection algorithms have attracted a great research interest recently, especially in the framework of the MPEG-4 and MPEG-7 standards and several algorithms have been reported in the literature dealing with the detection of cut, fading or dissolve changes either in the compressed or uncompressed domain. A shot is the part of the video that is captured by the camera between a record and a stop operation [18], or by video editing operations. The boundaries between shots are called shot changes, and the action of extracting the shot changes is called shot detection. A shot change can be abrupt or gradual. Examples of gradual changes are mixing, fade in and fade out. During mixing, both shots are shown for a short time (a few seconds). For fade in and fade out, the first and the second shots, respectively, are the blank shot.

After shot detection, motion segmentation is a key step in image sequence analysis and its results are extensively used for determining motion features of scene objects, as well as for coding purposes to reduce storage requirements [19]. In the past, various approaches have been proposed for motion or spatio-temporal segmentation. A recent survey of these techniques can be found in [20]. In these approaches, a 2-D motion or optical flow field is taken as input and a segmentation map is produced, where each region undergoes a movement described by a small number of parameters. There are top-down techniques which rely on the outlier rejection starting from the dominant motion, usually that of the background. Other techniques are bottom-up starting from an initial segmentation and merging regions until the final partition emerges [21] [22]. Direct methods are reported too [23] [24] [25]. All these techniques could be considered automatic, since only some tuning parameters are fixed by the user. Grinias and Tziritas [26] proposed a semi-automatic segmentation technique which is suitable for video object extraction for post-production purposes and object-scalable coding such as that introduced in the MPEG-4 standard.

In this paper an authoring tool for the MPEG-4 multimedia standard integrated with image sequence analysis algorithms is described. The tool handles the authoring process from the end-user interface specification phase to the cross-platform MP4 file. It fully exploits the object-based coding and 3D synthetic functionalities of the MPEG-4 standard. More specifically, the user can insert basic 3D objects (e.g. boxes, spheres, cones, cylinders) and text and modify their attributes. Generic 3D

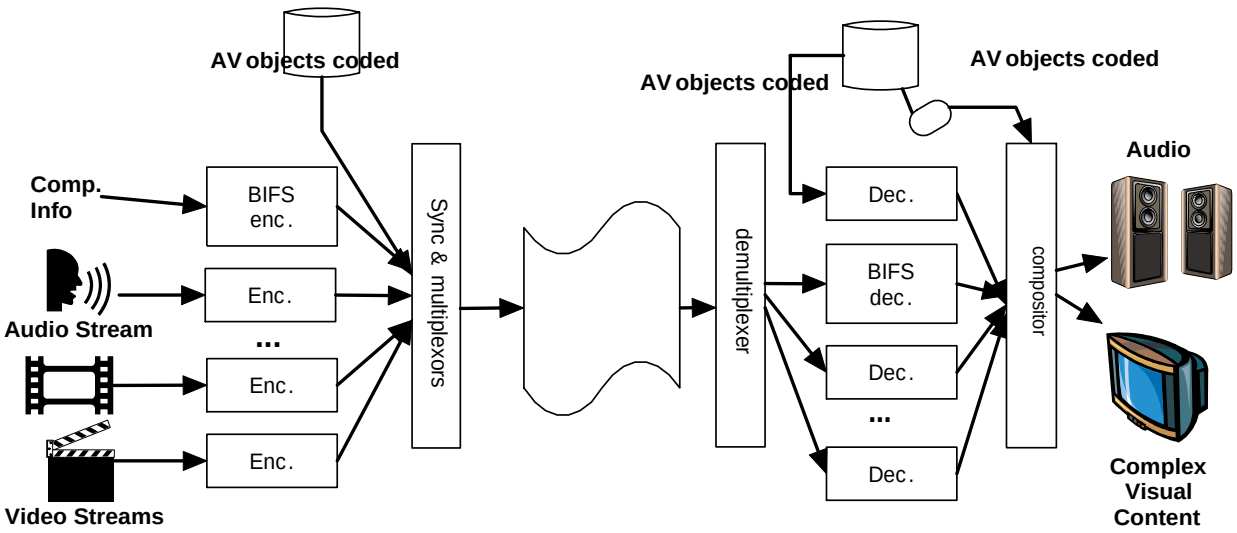



Fig. 1. Overview of MPEG-4 Systems.

models can be created or inserted and modified using the *IndexedFaceSet* node. Furthermore, the behavior of the objects can be controlled by various sensors (time, touch, cylinder, sphere, plane) and interpolators (color, position, orientation). Arbitrarily shaped static images and video can be texture mapped on the 3D objects. These objects are generated by using image sequence analysis integrated with the developed authoring tool. For the shot detection phase, the algorithm presented in [18] is used. It is based on a method for the extraction of the dc coefficients from MPEG-1 encoded video. After the shots have been detected in an image sequence, they are segmented and the extracted objects are tracked through time using a moving object segmentation and tracking algorithm. The algorithm is based on the motion segmentation technique proposed in [26]. The scheme incorporates an active user who delineates approximately the initial locations in a selected frame and specifies the depth ordering of the objects to be tracked. The segmentation tasks rely on a Seeded Region Growing (SRG) algorithm, initially proposed in [27] and modified to suit our purposes. First, colour-based static segmentation is obtained for a selected frame through the application of a region growing algorithm. Then, the extracted partition map is sequentially tracked from frame to frame using motion compensation and location prediction, as described in [26].

The user can modify the temporal behavior of the scene by adding, deleting and/or replacing nodes over time using the *Update* commands. Synthetic faces can also be added using the *Face* node and their associated Facial Animation Parameters (FAP) files. It is shown that our choice of an open and modular architecture of the MPEG-4 Authoring System endows it with the ability to easily integrate

new modules.

MPEG-4 provides a large and rich set of tools for the coding of audio-visual objects [28]. In order to allow effective implementations of the standard, subsets of the MPEG-4 Systems, Visual, and Audio tool sets have been identified, that can be used for specific applications. These subsets, called "Profiles", limit the tool set a decoder has to implement. For each of these Profiles, one or more Levels have been set, restricting the computational complexity. Profiles exist for various types of media content (audio, visual and graphics) and for scene descriptions. The authoring tool presented here is compliant with the following types of profiles: *The Simple Facial Animation* Visual Profile, *The Scalable Texture* Visual Profile, *The Hybrid* Visual Profile, *The Natural Audio* Profile, *The Complete Graphics* Profile, *The Complete Scene Graph* Profile, and the *The Object Descriptor* Profile which includes the Object Descriptor (OD) tool.

The paper is organized as follows. In Sections II and III the image sequence analysis algorithms used in the authoring process are presented. In Section IV MPEG-4 BIFS are presented and the classes of nodes in a MPEG-4 scene are defined. In Section V an overview of the authoring tool architecture and the graphical user interface is given. In Section VI, experiments demonstrate 3D scenes composed with the authoring tool. Finally, conclusions are drawn in Section VII.

## II. Shot detection

The shot detection algorithm, used in the authoring process is an adaptation of the method presented originally by Yeo and Liu [18]. The basic tenet is that the dc coefficients of the blocks from a MPEG-1 encoded video contain enough information for the purpose of shot detection. In addition, as shown in [18] the use of this spatially reduced image ("dc image") due to its smoothing effect, can reduce the effects of motion and increase the overall efficiency of the method. Computing the dc coefficient for P- and B- frames would be computationally complex, because it requires motion compensation. The dc coefficient is therefore approximated as a weighted average of the dc coefficients of the four neighboring blocks of the previous frame, according to the motion vector. The weights of the averaging operation are proportional to the surface of overlap between the current block and the respective block of the previous frame. By using this approximation and comparing each two subsequent images, using an appropriate metric as described in the sequel, a sequence of differences between subsequent frames is produced. Abrupt scene changes manifest themselves as sharp peaks at the sequence of differences. The algorithm must detect these peaks among the signal noise.

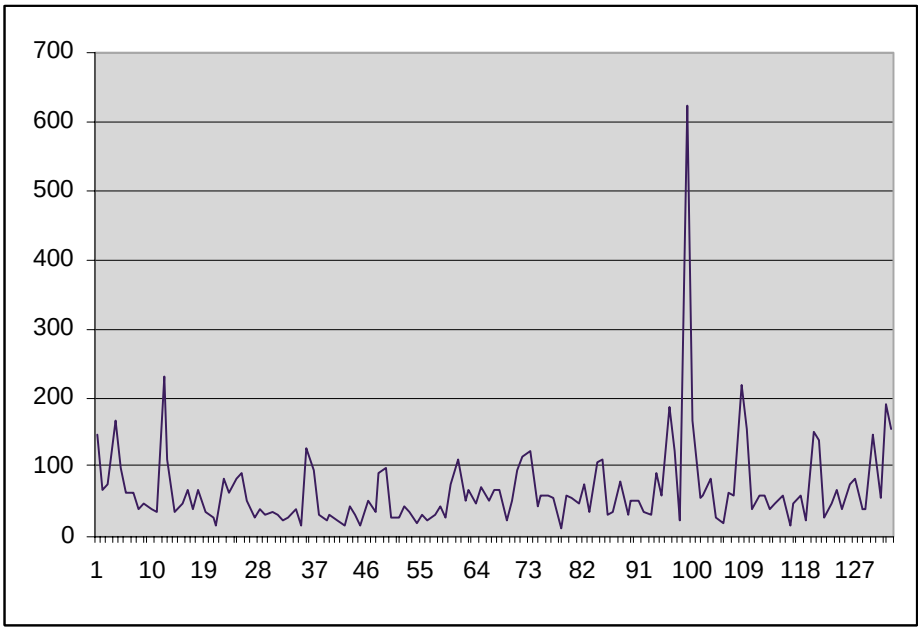



Fig. 2. Absolute difference of consecutive dc images.

In the proposed procedure, the video is not available in MPEG format, therefore the aforementioned method is applied to YUV raw video after a low – pass filtering, which effectively reduces each frame to a dc image.

Two metrics were proposed for comparing frames, that of the absolute difference, and that of the difference of the respective histograms. The first method, which was chosen by the authors of this paper for its computational efficiency, uses directly the absolute difference of the dc images [29]:

$$diff(X,Y) = \frac{1}{M \times N} \sum_{i,j} |x_{i,j} - y_{i,j}| \ , \tag{1}$$

where $M, N$ are the dimensions of the frame and $x_{i,j}$, $y_{i,j}$ represent two subsequent frames. As Yeo and Liu [18] note, this is not efficient in the case of full frames, because of the sensitivity of this metric to motion, but the smoothing effect of the dc coefficient estimation can compensate that, to a large extent. The second metric compares the histograms of the dc images. This method is insensitive to motion, [18] and most often, the number of bins $b$ used to form the histograms is in the range 4-6.

Once the difference sequence is computed (Fig. 2), a set of two rules is applied to detect the peaks. First, the peak must have the maximum value in an interval with a width of $m$ frames, centered at the peak. Secondly, the peak must be $n$ times greater than the second largest value of the second interval. This rule enforces the sharpness of the peak.

When just the two aforementioned rules were used, the system seemed to erroneously detect low-

valued peaks, which originated from errors related to P- and B- frames. These short peaks can be seen in Fig. 2. Therefore, we introduced a third rule, that of an absolute threshold, which excludes these short peaks. The threshold equals $d \times M \times N$, where $M, N$ are the dimensions of the frame and $d$ is a real parameter. In the case of histograms, the threshold is also proportional to $2^b$.

In our experiments, good results, in terms of shot recall and precision, were obtained with $m = 3-5$, $n = 1.5 - 2.0$ and $d \approx 0.0015$. A more thorough discussion on the topic of the choice of parameters can be found in [29].

Another issue is the relative importance of chrominance in peak detection. In particular, the formula $d = (1 - c)d_L + cd_C$ was applied. $c = 0.4 - 0.7$ gives good results, but acceptable results (about 30% inferior) are obtained with other values of this parameter as well.

## III. Moving object segmentation and tracking

### *A. Overall structure of video segmentation algorithms*

After shot detection, a common requirement in image sequence analysis is the extraction of a small number of moving objects from the background. The presence of a human operator, called here the *user* of the authoring tool, can greatly facilitate the segmentation work, for obtaining a semantically interpretable result. The proposed algorithm incorporates an active user for segmenting the first frame, and for subsequently dealing with occlusions during the moving object tracking.

For each object, including the background, the user draws a closed contour entirely contained within the corresponding object. Then, a region growing algorithm, expands the initial objects to their actual boundaries. Unlike [26], where the segmentation of the first frame is mainly based on the motion information, the region growing is based on the color of the objects and is done in a way that overcomes their color inhomogeneity. Having obtained the segmentation of the first frame, the tracking of any moving object is done automatically, as it is described in [26]. Only the layered representation of the scene is needed by the user in order to correctly handle overlaps. We assume that each moving region undergoes a simple translational planar motion, represented by a two-dimensional velocity vector, and we re-estimate an update for this vector from frame to frame using a region matching (RM) technique, which is an extension of block matching to regions of any shape and provides the required computational robustness. This motion estimation is performed after shrinking the objects, in order to ensure that object contours lie within the objects. The "shrunken" objects are projected

onto their predicted position in the next frame using motion compensation and the region growing algorithm is applied from that position.

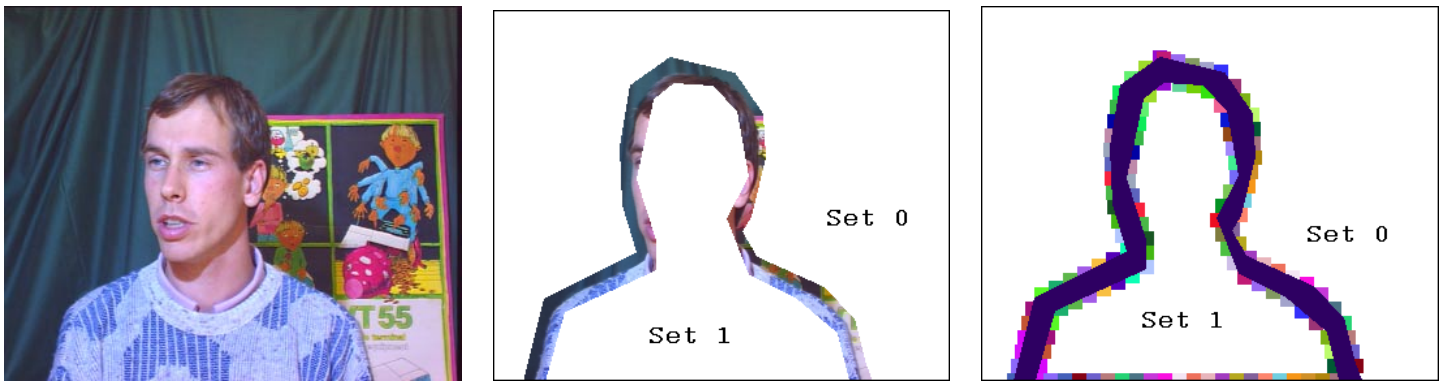
In the following subsection the Seeded Region Growing algorithm is presented. In Subsection III-C the initial segmentation is described, as well as the modifications applied to SRG, in order to cope with the color inhomogeneity of objects. Subsection III-D presents in summary, how the SRG algorithm is used for the temporal tracking of the initial segmentation.

### *B. The seeded region growing algorithm*

Segmentation is carried out by a seeded region growing algorithm which was initially proposed for static image segmentation using a homogeneity measure on the intensity function [27]. It is a sequential labelling technique, in which each step of the algorithm labels exactly one pixel, that with the lowest dissimilarity. Letting $n$ be the number of objects (classes), an initial set of connected components $A_1^0, A_2^0, \ldots, A_n^0$ is required. At each step $m$ of the algorithm, let $B^{m-1}$ be the set of all yet unlabelled points which have at least one immediate neighbor already labelled, *i.e.*, belonging to one of the partially completed connected components $\{A_1^{m-1}, A_2^{m-1}, \ldots, A_n^{m-1}\}$. In this work, 8-connection neighborhoods are considered. For each pixel $p \in B^{m-1}$, let us denote by $i(p) \in \{1, 2, \ldots, n\}$ the index of the set $A_i^{m-1}$ that $p$ adjoins and by $\delta(p, A_{i(p)}^{m-1})$ the dissimilarity measure between $p$ and $A_{i(p)}^{m-1}$, which depends on the segmentation features used. If the characterization of the sets is not updated during the sequential labelling process, the dissimilarity will be $\delta(p, A_{i(p)}^0)$. If $p$ adjoins two or more of the sets $A_i^{m-1}$, we define $i(p)$ to be the index of the set that minimizes the criterion $\delta(p, A_j^{m-1})$ over all neighboring sets $A_j^{m-1}$. In addition, we can distinguish a set $F$ of boundary pixels and add $p$ to $F$ when $p$ borders more than one set. In our implementation, boundary pixels $p$ are flagged as belonging to $F$ and at the same time, they are associated with the set that minimizes the dissimilarity criterion over all sets on whose boundary they lie. The set of boundary points $F$ is useful for boundary operations, as we shall see in Section III-D. Then we choose among the points in $B^{m-1}$ one satisfying the relation

$$z = \underset{p \in B^{m-1}}{\arg\min} \{\delta(p, A_{i(p)}^{m-1})\} \tag{2}$$

and append $z$ to $A_{i(z)}^{m-1}$, resulting in $A_{i(z)}^m$. This completes one step of the algorithm and finally, when the border set becomes empty after a number of steps equal to the number of initially unlabelled pixels, a segmentation map $(R_1, R_2, \ldots, R_n)$ is obtained with $A_i^m \subseteq R_i$ (for all $i, m$) and $R_i \cap R_j = \emptyset$ $(i \neq j)$, where $\cup_{i=1}^n R_i = \Omega$ is the whole image.




(a) (b) (c)

Fig. 3. User provided input of initial sets (b) and automatically extracted representative points (c) for *erik*'s frame 0(a).

For the implementation of the SRG algorithm, a list that keeps its members (pixels) ordered according to the criterion value $\delta(\cdot,\cdot)$ is used, traditionally referred to as Sequentially Sorted List (SSL).

### C. *Object initialization and static segmentation*

The initial regions required by the region growing algorithm must be provided by the user. A tool has been built for drawing a rectangle or a polygon inside any object. Then points which are included within these boundaries define the initial sets of object points. This concept is illustrated in Figure 3 (b), where the input of initial sets for the frame 0 of the sequence *erik* is shown. The user provides an approximate pattern for each object in the image that is to be extracted and tracked.

The color segmentation of the first frame is carried out by a variation of SRG. Since, the initial sets may be characterized by color inhomogeneity, on the boundary of all sets we place representative points, for which we compute the locally average color vector in the Lab system. In Figure 3 (c), the small square areas correspond to the regions of points that participate to the computation of the average color vector, for each such representative point. The dissimilarity of the candidate for labelling and region growing point $z$ of Eq. (2) from the labelled regions that adjoins is determined using this feature and the Euclidean distance, which may be possibly combined with the meter of the color gradient of $z$. After the labelling of $z$ the corresponding feature is updated. Therefore, we search for sequential spatial segmentation based on color homogeneity, knowing that the objects may be globally inhomogeneous, but presenting local color similarities, sufficient for their discrimination.

When the static color segmentation is completed, every pixel $p$ is assigned a label $i(p) \in \{1, 2, \ldots, n\}$, while boundary information is maintained in set $F$. Thus, the set map $i$ is the first segmentation map $i_0$, which is going to be tracked using the method that has been presented in [26] in detail and is

described shortly in the following subsection.

### *D. Tracking*

We now briefly describe how the result of the initial segmentation (set map $i_0$) is tracked over a number of consecutive frames. We assume the result has been tracked up to frame $k-1$ (set map $i_{k-1}$) and we now wish to obtain the set map $i_k$ corresponding to frame $k$ (partition of frame $k$). The initial sets for the segmentation of frame $k$ are provided by the set map $i_{k-1}$. The description of the tracking algorithm follows, while the motivations of the algorithm have already been presented in Section III-A.

For the purpose of tracking, a layered representation of the sets, rather than the planar one implied by SRG, is introduced in order to be able to cope with real world sequences which contain multiple motions, occlusions or a moving background. Thus, we assume that sets are ordered according to their distance from the camera:

$$\forall i, j \in \{1, 2, \ldots, n\}, \quad R_i \text{ moves behind } R_j \text{ if and only if, } \quad i < j \tag{3}$$

In this way, set $R_1$ refers to the background, set $R_2$ moves in front of set $R_1$ and behind the other sets, *etc.* The user is asked to provide this set ordering in the stage of objects initialization.

Having this set ordering available, for each set $R \in \{R_2, R_3, \ldots, R_n\}$ of set map $i_{k-1}$, the following operations are applied in order of proximity, beginning with the most distant:

- The border of $R$ is dilated for obtaining the set of seeds $A$ of $R$, which are required as input by SRG.
- The velocity vector of $R$ is re-estimated assuming that remains almost constant over time. The estimation is done using RM (with sub-pixel accuracy) on the points of $A$.
- The "shrunken" subset $A$ of region $R$ is translated from image $k-1$ to image $k$ according to the estimated displacement.

The last step, before applying the motion-based SRG, is the estimation of the background's velocity vector. Then, SRG is applied to points that remain unlabelled after the above operations, as it is described in [26].

Furthermore, two boundary regularization operations are proposed in [26] to stabilize object boundaries over time. The first one smooths the boundary of the objects, while the second computes an average shape using the information of a number of previously extracted segmentation maps.

### *E. System description*

The proposed algorithm was designed for semi-automatic segmentation requiring an initial user input (the user must draw a rough boundary of the desired object), therefore it is suited for an authoring tool where user interaction is expected. The spatio-temporal algorithm is a separate module developed in Java integrated with the authoring tool, which was developed in C++ for Windows (Borland Builder C++ 5) and OpenGL interfaced with the "core" module and the tools of the IM1 (MPEG-4 implementation group) software platform. The IM1 3D player is a software implementation of a MPEG-4 Systems player [30]. The player is built on top of the Core framework, which includes also tools to encode and multiplex test scenes. It aims to be compliant with the Complete 3D profile [1]. This shows the flexibility of the architecture of the presented authoring tool to efficiently combine different modules and integrate the results in the same MPEG-4 compatible scene. As can be seen for the experimental results, the SRG algorithm was shown to be very efficient. In case the tracking fails, the user can select a more appropriate boundary for the desired object else the tracking process may be restarted from the frame where the tracking failed.

## IV. BIFS scene description features

The image sequence analysis algorithms described above are going to be integrated with a MPEG-4 Authoring Tool providing a mapping of BIFS nodes and syntax to user friendly windows and controls. The BIFS description language [31] has been designed as an extension of the VRML 2.0 [32] file format for describing interactive 3D objects and worlds. VRML is designed to be used on the Internet, intranets, and local client systems. VRML is also intended to be a universal interchange format for integrated 3D graphics and multimedia. The BIFS version 2 is a superset of VRML and can be used as an effective tool for compressing VRML scenes. BIFS is a compact binary format representing a pre-defined set of scene objects and behaviors along with their spatio-temporal relationships. In particular, BIFS contains the following four types of information:

- The attributes of media objects, which define their audio-visual properties.
- The structure of the scene graph which contains these objects.
- The pre-defined spatio-temporal changes of these objects, independent of user input.
- The spatio-temporal changes triggered by user interaction.

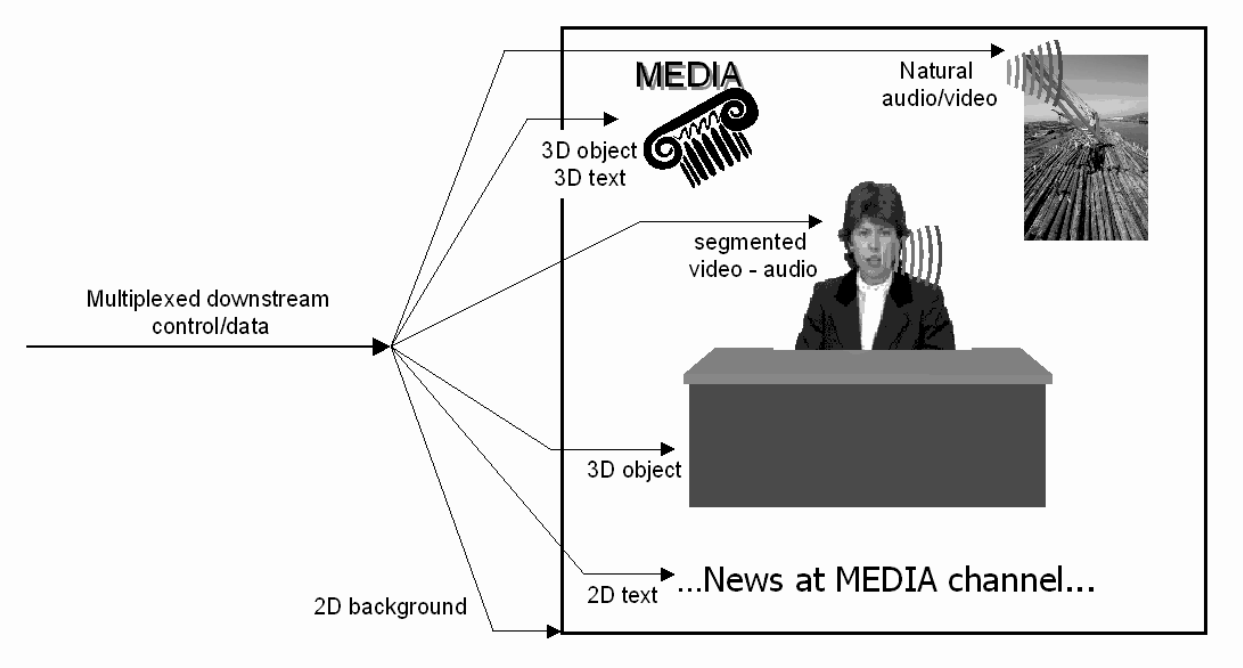



Fig. 4. Example of an MPEG-4 scene.

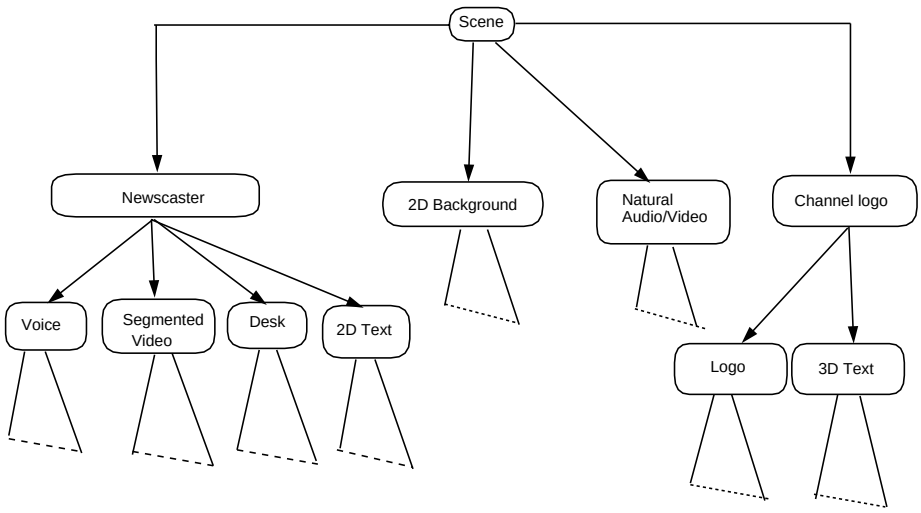



Fig. 5. Corresponding scene tree.

The scene description follows a hierarchical structure that can be represented as a tree (Figures 4, 5). Each node of the tree is an audiovisual object. Complex objects are constructed by using appropriate scene description nodes. The tree structure is not necessarily static. The relationships can evolve in time and nodes may be deleted, added or be modified. Individual scene description nodes expose a set of parameters through which several aspects of their behavior can be controlled. Examples include the pitch of a sound, the color of a synthetic visual object, or the speed at which a video sequence is to be played. There is a clear distinction between the audiovisual object itself, the attributes that enable

the control of its position and behavior, and any elementary streams that contain coded information representing attributes of the object.

The proposed MPEG-4 authoring tool implements the BIFS nodes graph structure allowing authors to take full advantage of MPEG-4 nodes functionalities in a friendly graphical user interface.

### *A. Scene structure*

Every MPEG-4 scene is constructed as a direct acyclic graph of nodes. The following types of nodes may be defined:

- *Grouping nodes* construct the scene structure.
- *Children nodes* are offsprings of grouping nodes representing the multimedia objects in the scene.
- *Bindable children nodes* are the specific type of children nodes for which only one instance of the node type can be active at a time in the scene (a typical example of this is the Viewpoint for a 3D scene; a 3D scene may contain multiple viewpoints or " cameras", but only one can be active at a time).
- *Interpolator nodes* constitute another subtype of children nodes which represent interpolation data to perform key frame animation. These nodes generate a sequence of values as a function of time or other input parameters.
- *Sensor nodes* sense the user and environment changes for authoring interactive scenes.

### *B. Nodes and fields*

BIFS and VRML scenes are both composed of collections of nodes arranged in hierarchical trees. Each node represents, groups or transforms an object in the scene and consists of a list of fields that define the particular behavior of the node. For example, a Sphere node has a radius field that specifies the size of the sphere. MPEG-4 has roughly 100 nodes with 20 basic field types representing the basic field data types: boolean, integer, floating point, two- and three-dimensional vectors, time, normal vectors, rotations, colors, URLs, strings, images, and other more arcane data types such as scripts.

### *C. ROUTEs and dynamical behavior*

The event model of BIFS uses the VRML concept of ROUTEs to propagate events between scene elements. ROUTEs are connections that assign the value of one field to another field. As is the case with nodes, ROUTEs can be assigned a "name" in order to be able to identify specific ROUTEs for

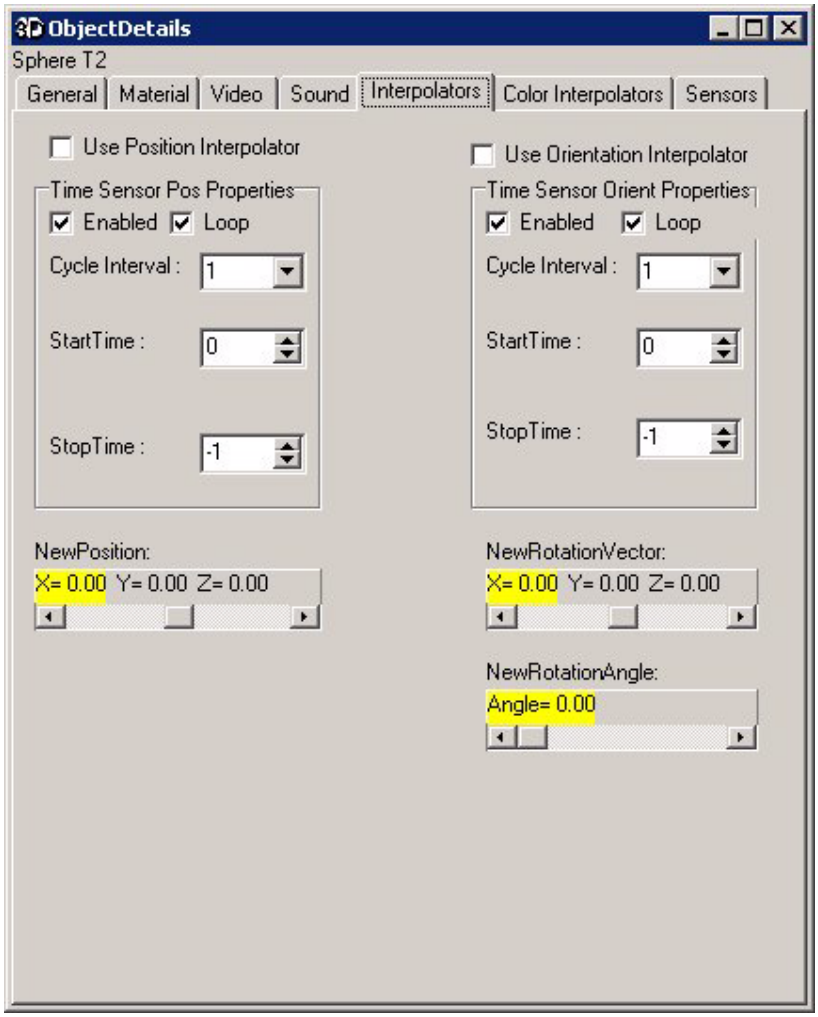



Fig. 6. The interpolators panel.

modification or deletion. ROUTEs combined with interpolators can cause animation in a scene. For example, the value of an interpolator is ROUTEd to the rotation field in a Transform node, causing the nodes in the Transform node's children field to be rotated as the values in the corresponding field in the interpolator node change with time. This event model has been implemented in a graphical way, allowing users to add interactivity and animation to the scene (Figure 6).

*D. Streaming scene description updates: BIFS-Command*

The mechanism with which BIFS information is provided to the receiver over time comprises the BIFS-Command protocol (also known as BIFS-Update), and the elementary stream that carries it, thus called BIFS-Command stream. The BIFS-Command protocol conveys commands for the replacement of a scene, addition or deletion of nodes, modification of fields, etc. For example, a "ReplaceScene" command becomes the entry (or random access) point for a BIFS stream, in exactly the same way

as an Intra frame serves as a random access point for video. A BIFS-Command stream can be read from the Web as any other scene, potentially containing only one "ReplaceScene" command, but it can also be broadcast as a "push" stream, or even exchanged in a communications or collaborative application. BIFS commands come in four main functionalities: scene replacement, node/field/route insertion, node/value/route deletion, and node/field/value/ route replacement. The BIFS-Command protocol has been implemented so as to allow the user to temporarily modify the scene using the authoring tool graphical user interface.

### *E. Facial Animation*

The Facial and Body Animation nodes can be used to render an animated face. The shape, texture and expressions of the face are controlled by the Facial Definition Parameters (FDPs) and/or the Facial Animation Parameters (FAPs). Upon construction, the face object contains a generic face with a neutral expression. This face can be rendered. It can also immediately receive the animation parameters from the bitstream, which will produce animation of the face: expressions, speech etc. Meanwhile, definition parameters can be sent to change the appearance of the face from something generic to a particular face with its own shape and (optionally) texture. If so desired, a complete face model can be downloaded via the FDP set. The described application implements the Face node, using the generic MPEG-4 3D face model, allowing the user to insert a synthetic 3D animated face.

## V. MPEG-4 Authoring Tool

### *A. System Architecture*

The process of creating MPEG-4 content can be characterized as a development cycle with four stages: Open, Format, Play and Save (Figure 7). In this somewhat simplified model, the content creators can:

• edit/format their own scenes inserting synthetic 3D objects, such as spheres, cones, cylinders, text, boxes and background (Figure 8). Also, they may group objects, modify the attributes (3D position, color, texture, etc) of the edited objects or delete objects from the content created. The user can perform the image sequence analysis procedures described in Sections II and III in order to create arbitrarily shaped video objects and insert them into the scene. Also, insert sound and natural video streams, add interactivity to the scene, using sensors and interpolators and control dynamically the scene using an implementation of the BIFS-Command protocol. Generic 3D models can be created

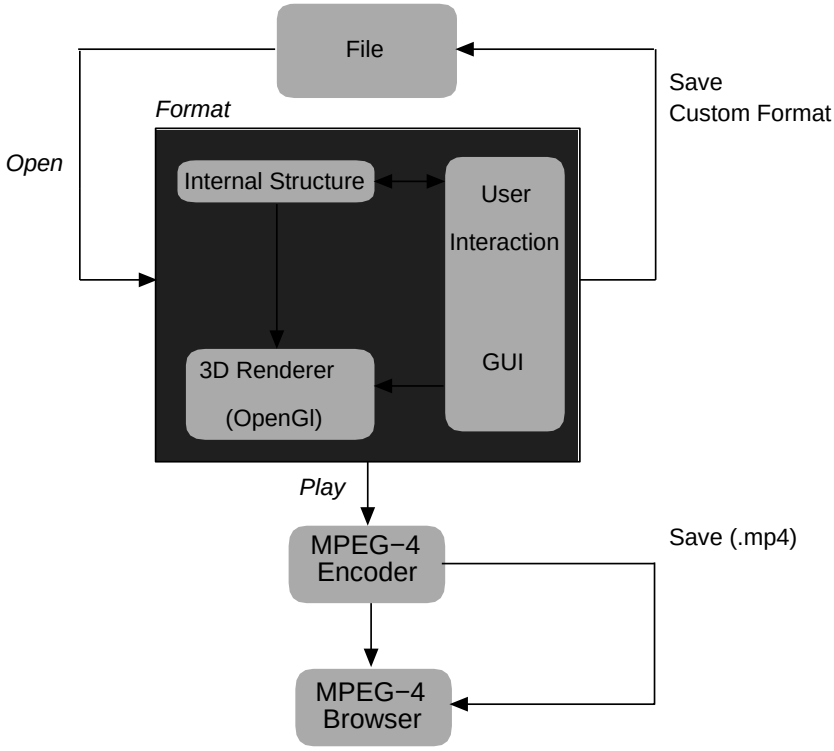



Fig. 7. System Architecture.

or inserted and modified using the *IndexedFaceSet* node. The user can insert a synthetic animated face using the implemented *Face* node. During these procedures the attributes of the objects and the commands as defined in the MPEG-4 standard and more specifically in BIFS, are stored in an internal program structure, which is continuously updated depending on the actions of the user. At the same time, the creator can see in real-time a 3D preview of the scene, on an integrated window using OpenGL tools (Figure 9). Further, the creator can:

- present the created content by interpreting the commands issued by the edition phase and allowing the possibility of checking whether the current description is correct.
- open an existing file.
- save the file either in custom format or after encoding/multiplexing and packaging in a MP4 file [28], which is expected to be the standard MPEG-4 file format. The MP4 file format is designed to contain the media information of an MPEG-4 presentation in a flexible, extensible format which facilitates interchange, management, editing and presentation of the media.

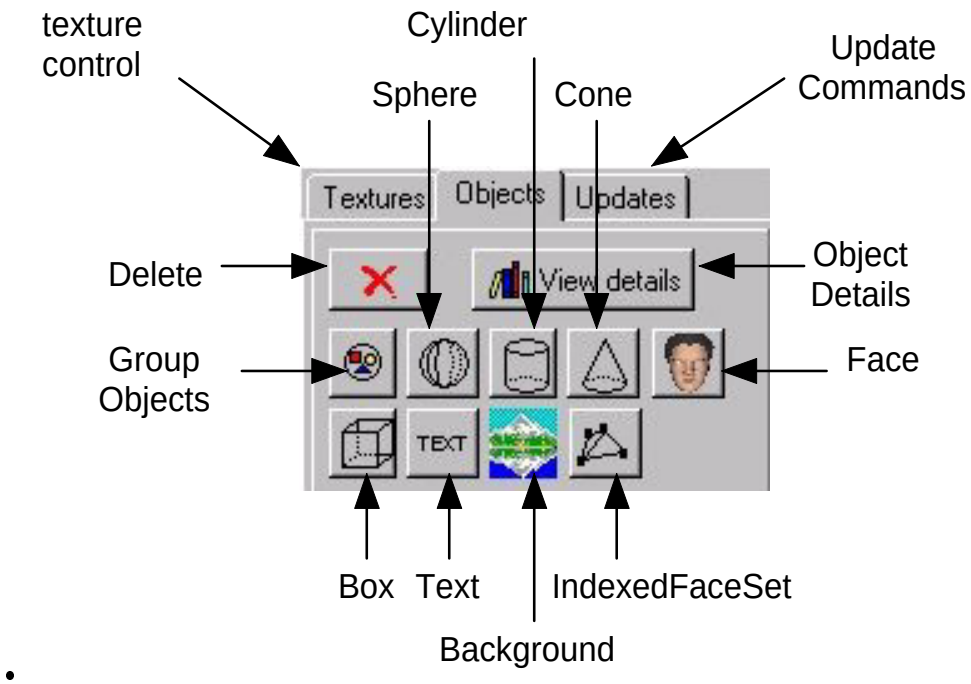



Fig. 8. Authoring tool application toolbar.

*B. User Interface*

To improve the authoring process, powerful graphical tools must be provided to the author [33]. The temporal dependence and variability of multimedia applications, hinders the author from obtaining a real perception of what he is editing. The creation of an environment with multiple, synchronized views and the use of OpenGL was implemented to overcome this difficulty. The interface is composed of three main views, as shown in Figure 9.

*Edit/Preview:* By integrating the presentation and editing phases in the same view the author is enabled to see a partial result of the created object on an OpenGL window. If any given object is inserted in the scene, it can be immediately seen on the presentation window (OpenGL window) located exactly in the given 3D position. The integration of the two views is very useful for the initial scene composition.

*Scene Tree:* This attribute provides a structural view of the scene as a tree (a BIFS scene is a graph, but for ease of presentation, the graph is reduced to a tree for display). Since the edit view cannot be used to display the behavior of the objects, the scene tree is used to provide more detailed information concerning them. The drag-n-drop and copy-paste modes, can also be used in this view.

*Object Details:* This window, shown in Figure 10, offers object properties that the author can use to assign values other than those given by default to the synthetic 3D objects. The user can

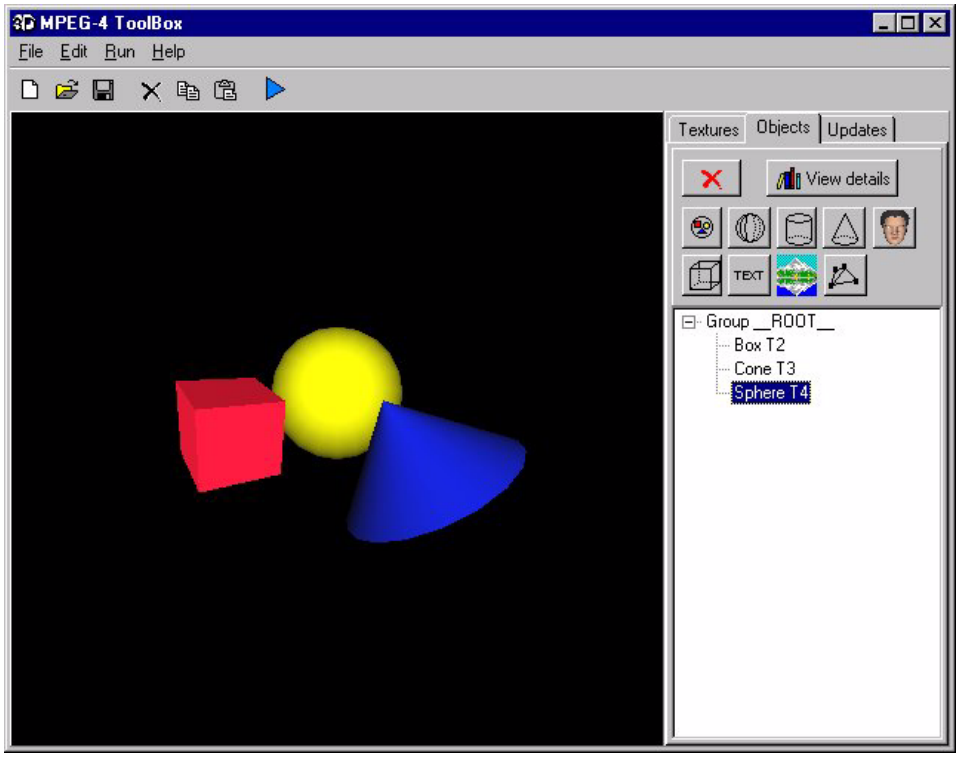



Fig. 9. Main Window, indicating the different components of the user interface.

perform the image sequence analysis procedures described in Sections II and III in order to create arbitrarily shaped video objects and insert them into the scene. This arbitrarily shaped video can be used as texture on every object. Other supported properties are: 3D position, 3D rotation, 3D scale, color (diffuse, specular, emission), shine, texture, video stream, audio stream (the audio and video streams are transmitted as two separated elementary streams according to the object descriptor mechanism), cylinder and cone radius and height, textstyle (plain, bold, italic, bolditalic) and fonts (serif, sans, typewriter), sky and ground background, texture for background, interpolators (color, position, orientation) and sensors (sphere, cylinder, plane, touch, time) for adding interactivity and animation to the scene. Furthermore, the author can insert, create and manipulate generic 3D models using the *IndexedFaceSet* node. Simple VRML files can also be inserted in a straightforward manner. Synthetically animated 3D faces can be inserted by the *Face* node. The author must provide a FAP file [34] and the corresponding Encoder Parameter File (EPF), which is designed to give the FAP encoder all information related to the corresponding FAP file, like I and P frames, masks, frame rate, quantization scaling factor and so on. Then, a **bifa** file (**bi**nary **f**ormat for **a**nimation) is automatically created, so as to be used in the Scene Description and Object Descriptor files.

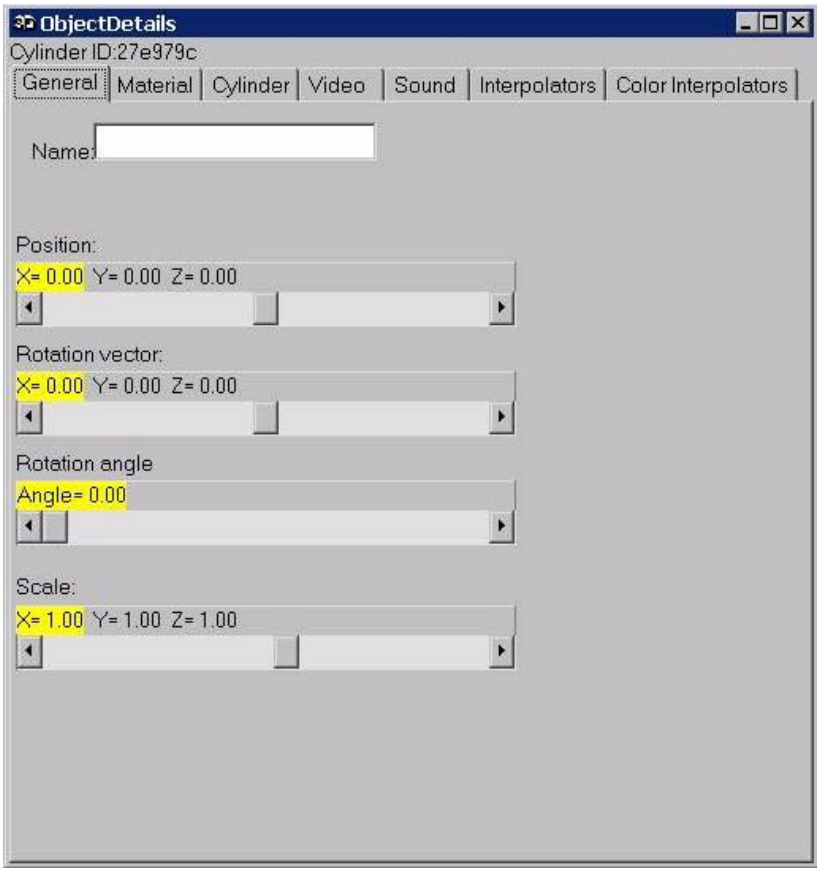


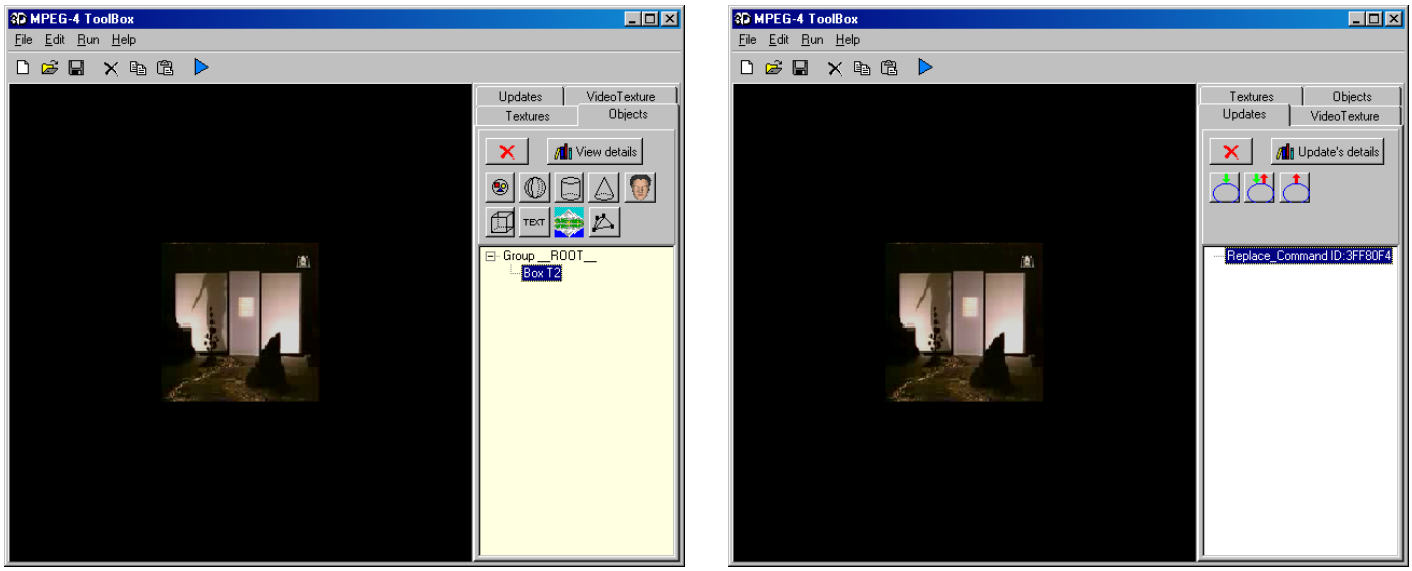

Fig. 10. Object details Window, indicating the properties of the objects.

## VI. EXPERIMENTAL RESULTS

In this section two examples are presented, describing the steps that lead to the creation of two MPEG-4 scenes.

The first example demonstrates the use of the BIFS-Commands (Update), which is used to give to the user a real perception about what he/she is editing in a temporal editing environment. In this scene, a textured box is first created and after a period of time is replaced by a textured sphere. The exact steps are the following: On the main window, a Box with a video texture is created (Figure 11 (a)). On the Updates tab (Figure 11 (b)) the Replace command is selected ("Replace" button). On the Update Command Details panel (Figure 12 (a)) in tab "UpdateData" a sphere with another video texture is selected. On the same panel, in tab "General", (Figure 12 (b)) the box is specified ("Set Target" button) and the time ("Time of Action" button), of action needed (e.g. 500 ms). Finally, by pressing the button "Play" the result is shown by the 3D MPEG-4 Player (Figures 13 (a), (b)).

The second example leads to the creation of an MPEG-4 scene containing arbitrarily shaped video

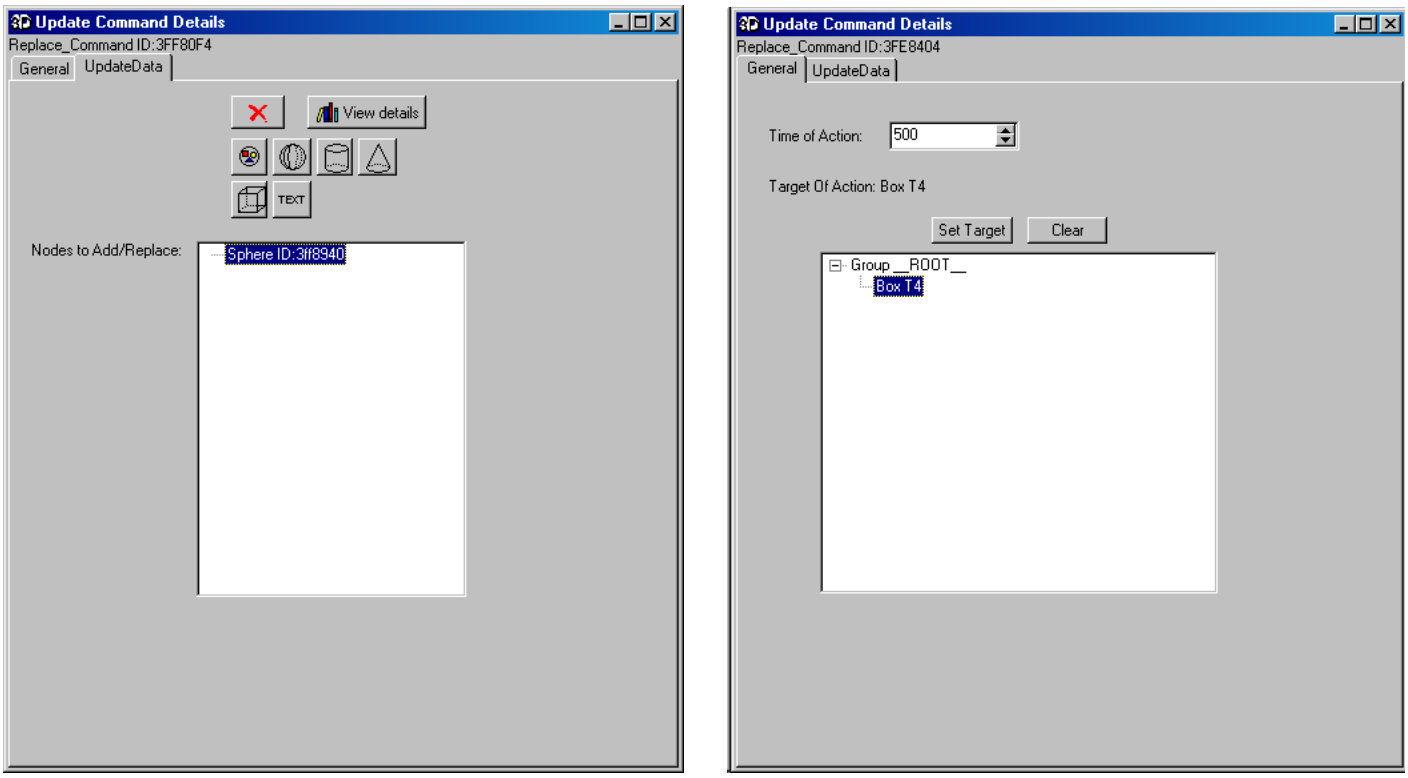



(a) (b)

Fig. 11. Using Update Commands in the Authoring Tool

objects using the shot detection and object segmentation procedures. The scene represents a virtual studio (Figure 21). The scene contains several groups of synthetic objects including boxes with textures, and text objects (Figure 20). The "logo" group which is located on the upper left corner of the studio is composed of a rotating box and a text object that describes the name of the channel. The background contains four boxes (left-right side, floor and back side) with image textures. The desk is created using two boxes. On the upper right corner of the scene a box with natural video texture is presented. On this video-box relative videos are loaded according to the news. The newscaster (image sequence "Akiyo") is an arbitrarily shaped video object produced using the algorithms described in Sections II and III.

In order to test the shot detection algorithm, a test sequence was created composed of the two image sequences "Akiyo" and "Eric". Using the user interface of the authoring tool (Fig. 14) the user can select a video for processing. The supporting formats are YUV color and grayscale at $176 \times 144$ pixels (QCIF) and $352 \times 288$ pixels (CIF). As soon as the user selectes the video, the algorithm presented in Section II is performed. The result is the temporal segmentation of the image sequence into shots. After the shot detection procedure, the semi-automatic moving object segmentation procedure begins (Section III). The user draws a rough boundary around the moving foreground object (Fig. 15) of each shot and the algorithm automatically performs the region growing and tracking procedures (Fig. 16). The result is a set of segmentation masks for each shot of the image sequence (Fig. 14). The user can easily select the objects from each shot that are to be included in the scene. Every selected mask (for



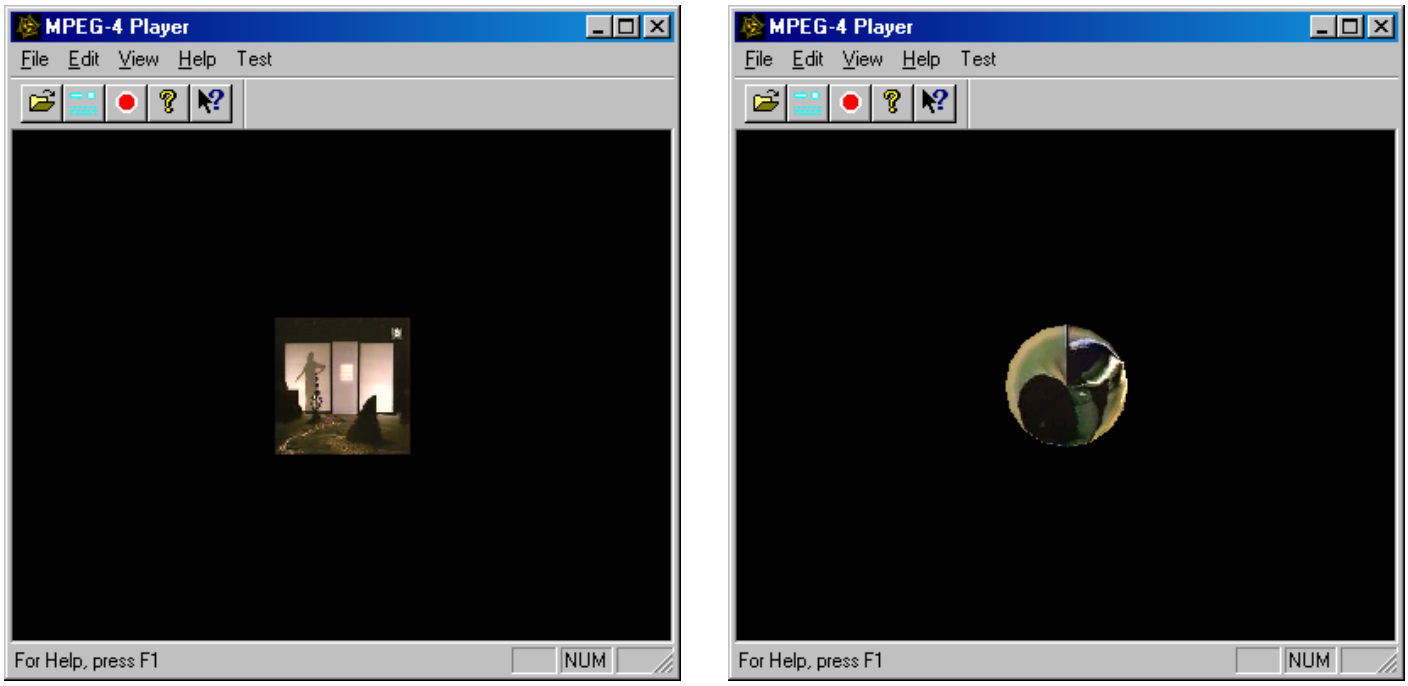

(a) (b)

Fig. 12. Specifying the appropriate properties

every shot) is given as input to the MPEG-4 Video Reference Software [35] which is used for encoding and decoding video sequences. After a transcoding procedure, the final result is an H.263 video which is compliant with the current MPEG-4 IM1 player implementation. This video can be used as texture on every object as shown in Figures 17, 18. The block diagram of the complete procedure is presented in Fig. 19.

The same scene can be easily modified so as to contain a synthetic newscaster (Figure 23). The body of the newscaster is an IndexedFaceSet imported from a VRML 3D model. The 3D face was inserted by using the corresponding button. After the selection of a FAP (Face Animation Parameters) file and an audio stream (a saxophone appears on the upper left corner), the face animation is configured according to the selected FAP file. The video stream (H.263) and the audio stream (G.723) are transmitted as two separate elementary streams according to the object descriptor mechanism. All animation (except the face animation) is implemented using interpolator nodes. Some major parts of the produced scene description file (.txt) are the following:

```
    DEF ID_014    AnimationStream      #fap animation stream
    {
             url 50
    }
  Transform {
```




(a) (b)

Fig. 13. The result of the Update Commands as shown in the Authoring Tool.

```
        translation  0.000 1.529 1.690
        rotation  0.000 0.000 0.000 0.000
        scale  0.013 0.013 0.013
    Children Face                    #face node
    {
        fap DEF ID_104 FAP{}
        renderedFace []
    }
  }
. . .
DEF T120661744  Transform {
    translation  0.000 0.000 0.000
    rotation  1.786 1.014 0.000 0.911
    children Shape {
        appearance Appearance {
        texture ImageTexture {
             url 10
     }
        textureTransform TextureTransform {
    }
}
        geometry Box {        #box with image texture
            size 0.796 0.796 0.694
        }
    }
}

DEF OrientTS120658180      TimeSensor {
    stopTime -1
    startTime 0
    loop TRUE                    # time sensor for interpolation
                                 #    purposes
    cycleInterval 15
}
DEF ORI120658180    OrientationInterpolator {
    key      [0, 1]
```

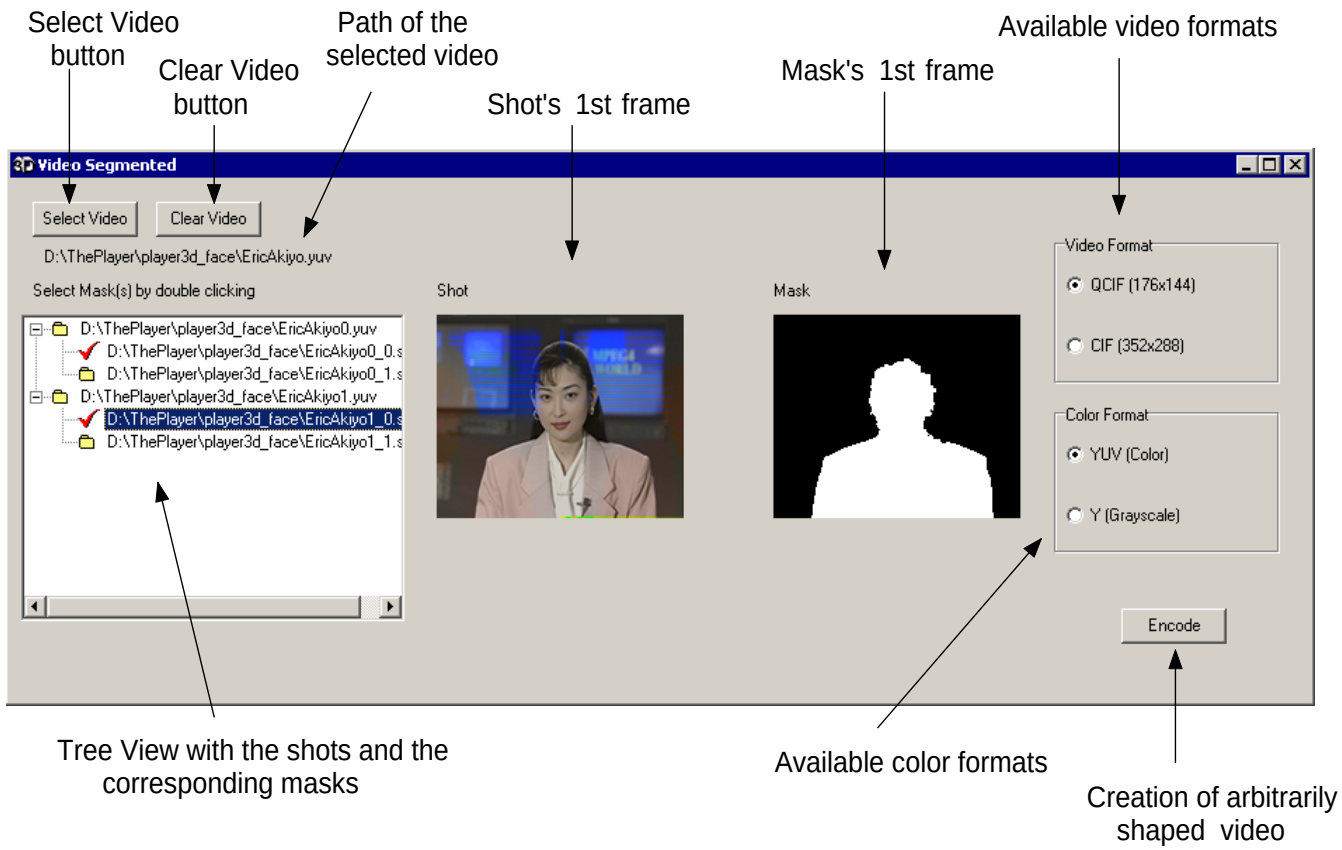



Fig. 14. The segmentation form in the Authoring Tool.

```
    keyValue [0.000 0.000 0.000 0.000 ,0.000 0.200 0.000 3.143 ]
}
. . .
ROUTE OrientTS120658180 .fraction_changed TO ORI120658180.set_fraction
ROUTE ORI120658180 .value_changed TO T120661744 .rotation
```

The AnimationStream node reads from an external source the selected FAP file. The Transform node inserted before the Face node, controls the position of the animated face in the scene. The Face node inserts the animated face and connects it with the FAP file defined earlier. The following group creates the "logo" which is located on the upper left corner and more specifically, the textured rotating box. First the position of the box (Transform node) and then the image to be applied as texture (appearance and texture fields) is defined. Finally the geometry and the dimensions of the object are defined (geometry node). In our case the object is a box. The final part contains the necessary nodes for creating the rotating motion. First, the period of the motion is defined (how fast the box will be rotated) and whether the rotation speed will be constant. This is controlled by the TimeSensor node and the loop and cycleInterval fields. The OrientationInterpolator node defines the intermediate positions of the motion. Finally, the ROUTE nodes connect the defined parameters of

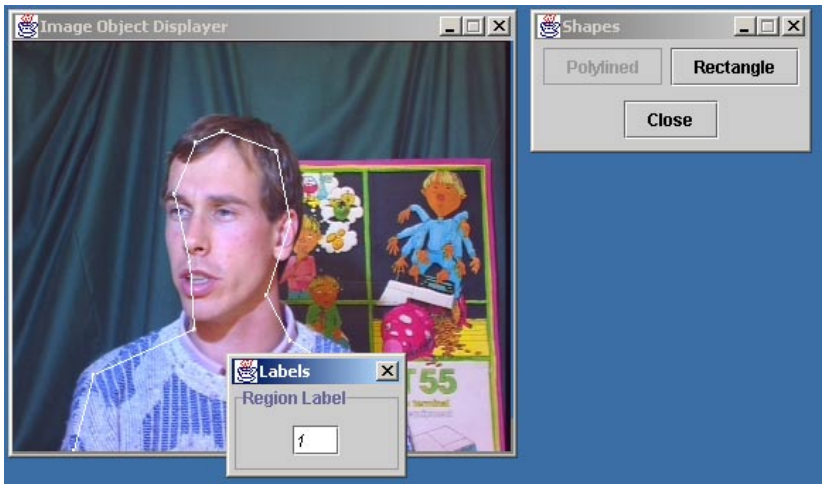



Fig. 15. Rough boundary around the foreground object.

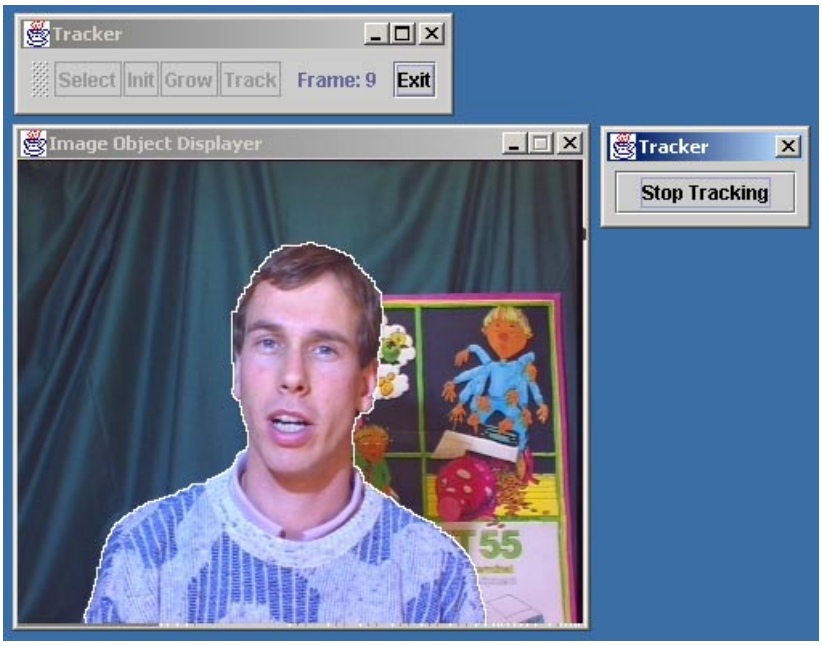



Fig. 16. Snapshot of the tracking procedure.

the movement to the textured object. The objects are uniquely characterized by the DEF nodes. For example, the texture box is object T120661744.

As can be seen from the above, the text based description format for MPEG-4 is very complicated. It is almost impossible to develop an MPEG-4 scene from scratch using only text. The user should be aware of a complicated syntax and a great number of MPEG-4 BIFS node names and at the same time keep track of all object names defined. The presented authoring tool allows non-expert MPEG-4 users to create complicated scenes by converting this text-based description to a more native, graphical description.

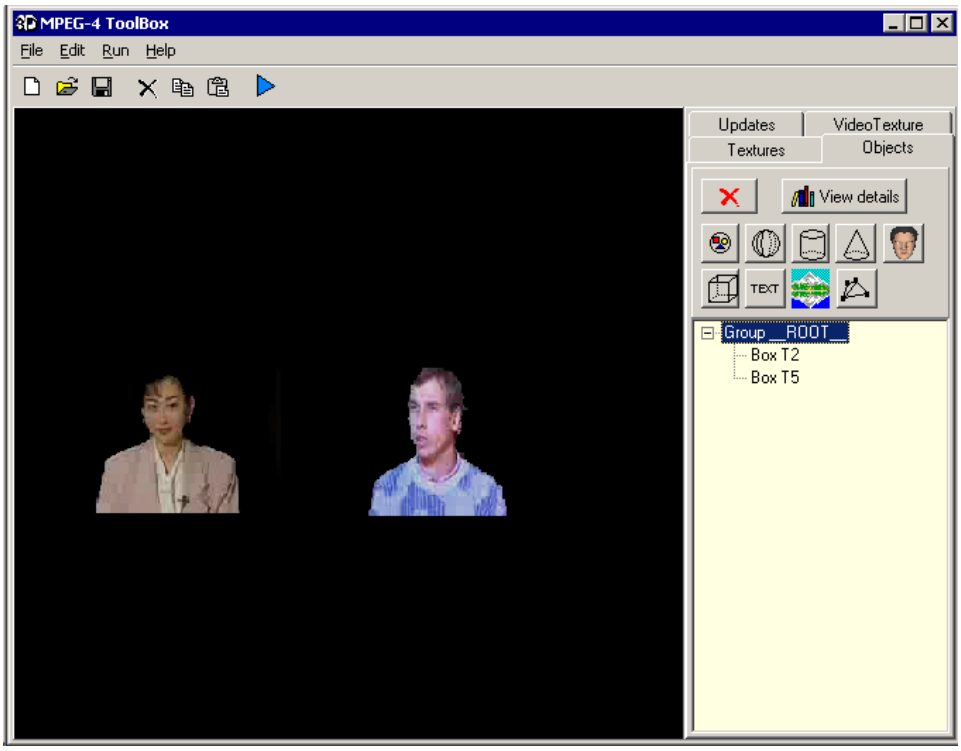



Fig. 17. The result of the arbitrarily shaped video objects creation textured on two boxes as shown in the Authoring Tool.

## VII. Conclusions

In this paper an authoring tool for the MPEG-4 multimedia standard integrated with image sequence analysis algorithms was presented. The tool maps BIFS features and functionalities to common Window controls allowing users to efficiently create or edit and finally play MPEG-4 compliant scenes using an external MPEG-4 player. The authoring tool is integrated with a shot detection algorithm along with a semi-automatic method for moving object segmentation and tracking. The user can perform these image sequence analysis procedures in order to create arbitrarily shaped video objects and insert them into the scene. Experimental results demonstrated that it is possible to create complex scenes using unique MPEG-4 features such as object-based coding, Updates and Facial Animation. The image sequence analysis algorithms were integrated as separate modules. This shows the flexibility of the architecture of the presented authoring tool to efficiently combine different modules and integrate the results in the same MPEG-4 compatible scene.

The presented parts of the corresponding Text Description Files show that it is almost impossible for the non-expert to build even simple MPEG-4 scenes from scratch using only text. We found that while content developers were satisfied with the efficiency and the effectiveness of the system,

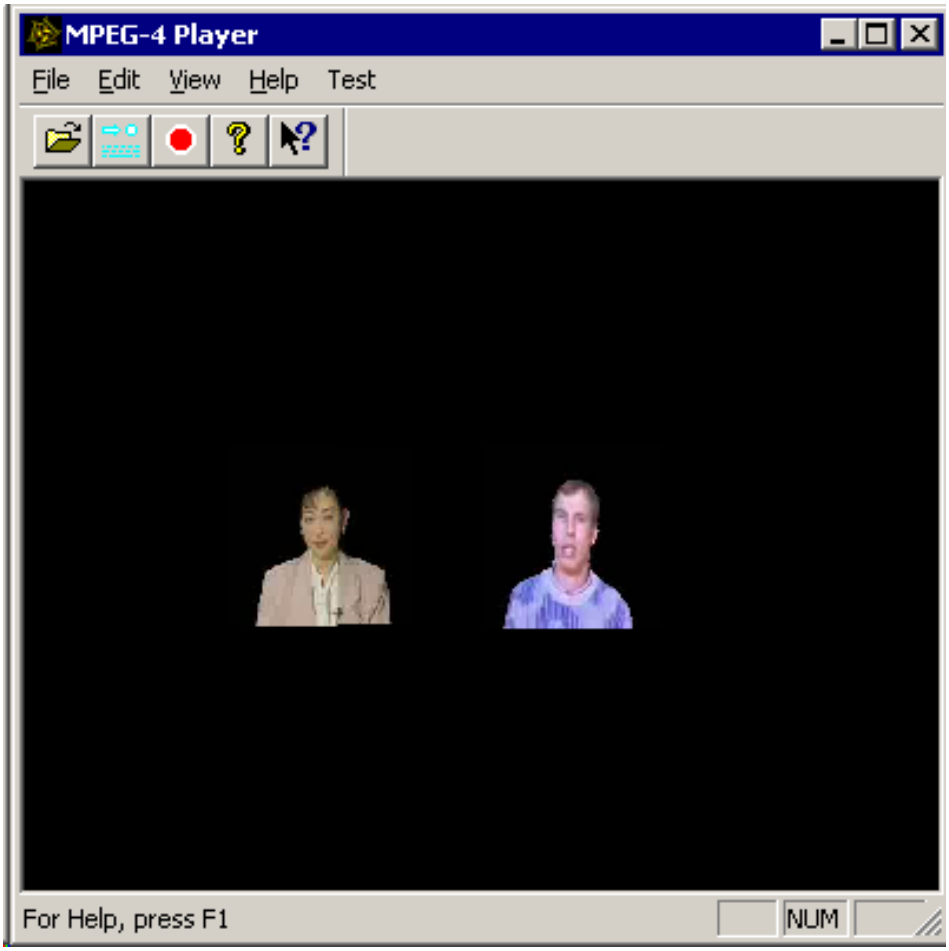



Fig. 18. The result of the shot detection and segmentation procedures as shown in the player.

those that were not familiar with the MPEG-4 standard had problems understanding the terminology used. Thus, further development and refinement is needed before the tool can be useful for large-scale deployment.

Another important feature of the authoring tool is that it produces scenes which are totally MPEG-4 compliant. These scenes can be visualized using the IM1-3D player developed by the MPEG-4 group without any modifications. Thus, the tool may be used to create MPEG-4 compliant applications without introducing proprietary features.

The presented paper, also, highlights and exemplifies the manner in which non-expert MPEG-4 users may create and manipulate MPEG-4 content. Specifically, the tool developed is intended to help MPEG-4 algorithm and system developers integrate their algorithms and make them available through a user friendly interface. It may also help as a beginning for the development of new tools. Finally the tool may serve as a benchmark for the comparison of other, proprietary or not, authoring tools to

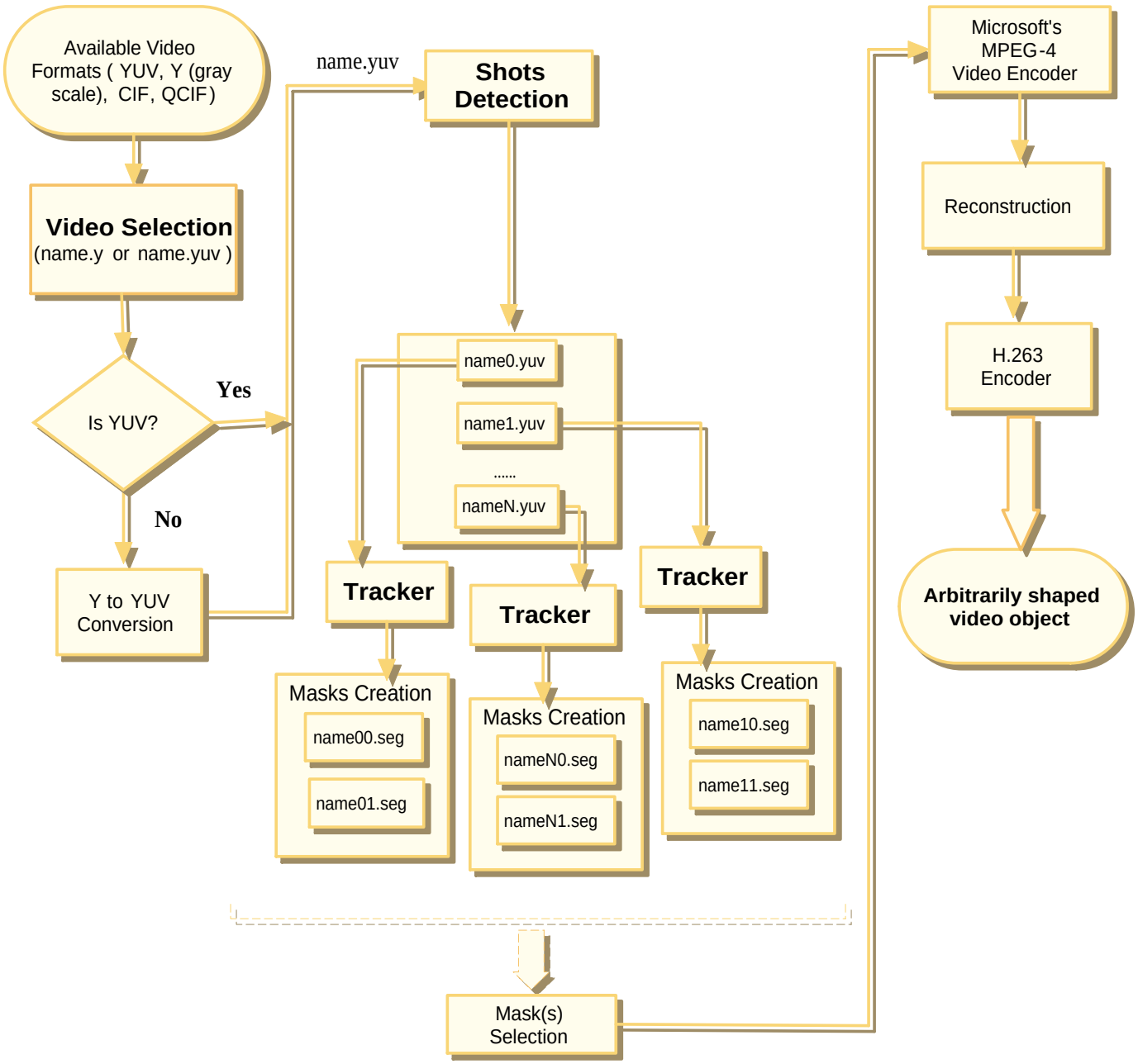



Fig. 19. Segmentation procedure for the creation of h.263 video format.

one with the capabilities of the MPEG-4 system.

# References


[1] "Tutorial issue," *Signal Processing:Image Communication, Tutorial issue on MPEG-4*, vol. 15, no. 4-5, 2000.

[2] R. Koenen, "MPEG-4 Multimedia for our Time," *IEEE Spectrum*, vol. 36, pp. 26–33, Feb. 1999.

[3] L. Chiariglione, "MPEG and Multimedia Communications," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, pp. 5–18, Feb. 1997.

[4] F. Pereira, "MPEG-4:Why, what, how and when?," *Signal Processing:Image Communication*, vol. 15, pp. 271–279, 2000.

[5] S. Boughoufalah, J. C. Dufourd, and F. Bouilhaguet, "MPEG-Pro, an Authoring System for MPEG-4," in *ISCAS 2000-IEEE International Symposium on Circuits and Systems*, (Geneva, Switzerland), May 2000.

[6] V. K. Papastathis, I. Kompatsiaris, and M. G. Strintzis, "Authoring tool for the composition of MPEG-4 audiovisual scenes," in *International Workshop on Synthetic Natural Hybrid Coding and 3D Imaging*, (Santorini, Greece), September 1999.

[7] H. Luo and A. Eleftheriadis, "Designing an interactive tool for video object segmentationa and annotation," in *ACM Multimedia-99*, March 1999.

[8] P. Correia and F. Pereira, "The role of analysis in content-based video coding and interaction," *Special Issue on Video Sequence Segmentation for Content-Based Processing and Manipulation, Signal Processing Journal*, vol. 26, no. 2, 1998.

[9] B. Erol and F. Kossentini, "Automatic key video object plane selection using the shape information in the MPEG-4 compressed domain," *IEEE Trans. on Multimedia*, vol. 2, pp. 129–138, June 2000.

[10] B. Erol, S. Shirani, and F. Kossentini, "A concealment method for shape information in MPEG-4 coded video sequences," *IEEE Trans. on Multimedia*, vol. 2, no. 3, pp. 185–190, 2000.

[11] IBM Hotmedia Website, "http://www-4.ibm.com/software/net.media/," 2000.

[12] Veon Website, "http://www.veon.com," 2000.

[13] iVAST Website, "http://www.ivast.com," 2000.

[14] Envivio Website, "http://www.envivio.com," 2000.

[15] P. Daras, I. Kompatsiaris, T. Raptis, and M. G. Strintzis, "MPEG-4 Authoring Tool for the Composition of 3D Audiovisual Scenes," in *ISCAS 2001-IEEE International Symposium on Circuits and Systems*, (Sydney, Australia), May 2001.

[16] P. Daras, I. Kompatsiaris, T. Raptis, and M. G. Strintzis, "Authoring Tool for the Composition of 3D Audiovisual Scenes using the MPEG-4 Standard," in *DCV'01-Digital and Computational Video*, (Tampa, Florida, U.S.A.), Feb 2001.

[17] Y. Avrithis, A. Doulamis, N. Doulamis, and S. Kollias, "A Stochastic Framework for Optimal Key Frame Extraction from MPEG Video Databases," *Comp. Vision and Image Understanding*, vol. 75, pp. 3–24, Jul. 1999.

[18] B.-L. Yeo and B. Liu, "Rapid Scene Analysis on Compressed Video," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 5, pp. 533–544, Dec. 1995.

[19] G. Tziritas and C. Labit, "Motion Analysis and Image Sequence Coding," *Elsevier*, 1994.

[20] A. Mitiche and P. Bouthemy, "Computation and analysis of image motion: a synopsis of current problems and methods," *Intern. Journal on Computer Vision*, vol. 19, pp. 29–55, Jul 1996.

[21] C. K. Cheong and K. Aizawa, "Structural motion segmentation based on probabilistic clustering," in *IEEE Int. Conf. on Image Processing*, vol. I, pp. 505–508, 1996.

[22] F. Moscheni, S. Bhattacharjee, and M. Kunt, "Spatiotemporal segmentation based on region merging," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-20, pp. 897–915, Sep. 1998.

[23] G. Adiv, "Determining three-dimensional motion and structure from optical flow generated by several moving objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, pp. 384–401, Jul. 1985.

[24] J. M. Odobez and P. Bouthemy, "Direct incremental model-based image motion segmentation for video analysis," *Signal Processing*, vol. 66, pp. 143–155, 1998.

[25] J. Wang and E. Adelson, "Representing moving images with layers," *IEEE Trans. on Image Processing*, vol. IP-3, no. 5, pp. 625–638, 1994.

[26] I. Grinias and G. Tziritas, "A semi-automatic seeded region growing algorithm for video object localization and tracking," *Signal Processing : Image Communication*, vol. 16, pp. 977–986, Aug 2001.

[27] R. Adams and L. Bischof, "Seeded Region Growing," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 641–647, Jun. 1994.

[28] R. Koenen, "MPEG-4 Overview - (V.16 La BauleVersion)," *ISO/IEC JTC1/SC29/WG11 N3747*, October 2000.

[29] G. Akrivas, N. Doulamis, A. Doulamis, and S. Kollias, "Scene Detection Methods for MPEG- encoded Video Signals," in *Proceedings of the MELECON 2000 Mediterranean Electrotechnical Conference*, (Nicosia, Cyprus), May 2000.

[30] Z. Lifshitz, "Status of the Systems Version 1, 2, 3 Software Implementation," tech. rep., ISO/IEC JTC1/SC29/WG11 N3564, July 2000.

[31] J. Signès, Y. Fisher, and A. Eleftheriadis, "MPEG-4's Binary Format for Scene Description," *Signal Processing:Image Communication, Special issue on MPEG-4*, vol. 15, no. 4-5, pp. 321–345, 2000.

[32] ISO/IEC 14472-1, "The Virtual Reality Modeling Language," *http://www.vrml.org/Specifications/VRML97*, 1997.

[33] B. MacIntyre and S. Feiner, "Future multimedia user interfaces," *Multimedia Systems*, vol. 4, no. 5, pp. 250–268, 1996.

[34] University of Genova. Digital Signal Processing Laboratory, "http://www-dsp.com.dist.unige.it/ snhc/fba_ce/facefrmt.htm," 2000.

[35] S. A. J. Winder, "MPEG-4 Video Reference Software," tech. rep., ISO/IEC 14496 (MPEG-4) Video Reference Software, Jul. 2001.

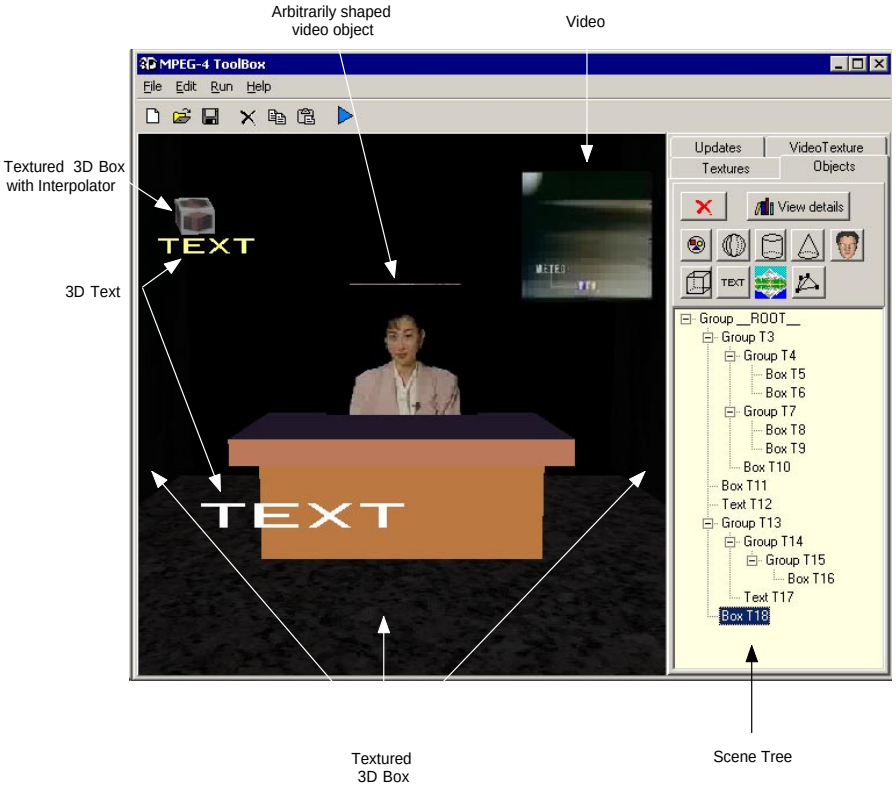



Fig. 20. The virtual studio scene using arbitrarily shaped video objects in the authoring tool.


Fig. 21. The virtual studio scene in the IM1 3D player.

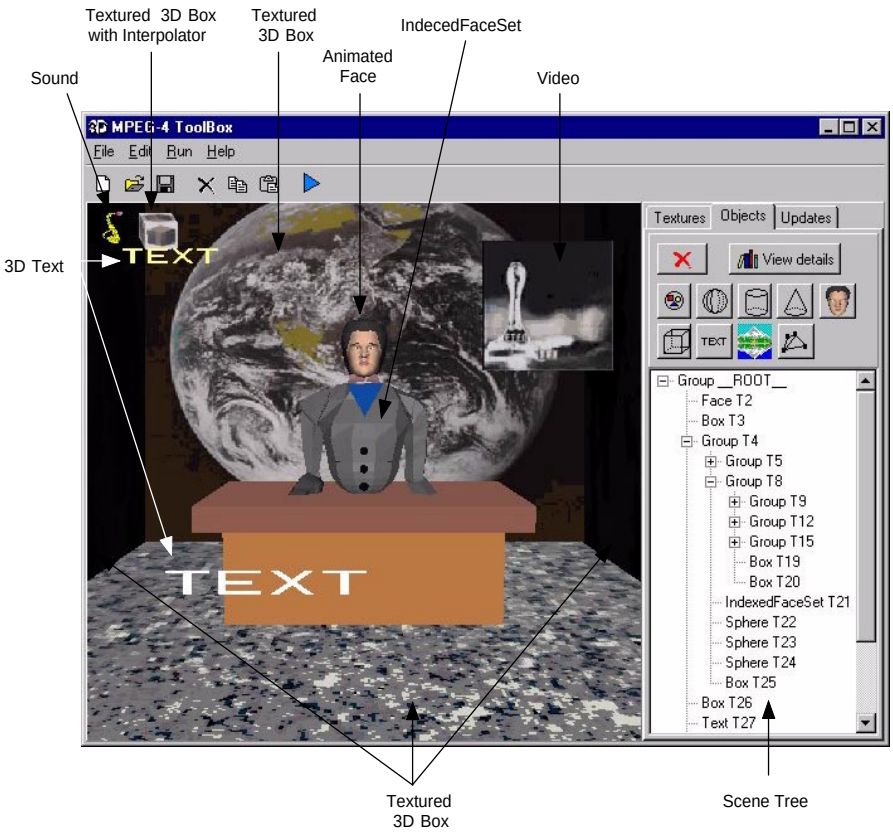



Fig. 22. The virtual studio scene in the authoring tool.


Fig. 23. The virtual studio scene in the IM1 3D player.