13

Multimedia Annotation Tools

Carsten Saathoff,[1] Krishna Chandramouli,[2] Werner Bailer,[3] Peter Schallauer[3] and Raphaël Troncy[4]
[1]*ISWeb – Information Systems and Semantic Web, University of Koblenz-Landau, Koblenz, Germany*
[2]*QMUL Queen Mary University of London, London, UK*
[3]*JOANNEUM RESEARCH – DIGITAL, Graz, Austria*
[4]*Centrum Wiskunde & Informatica, Amsterdam, The Netherlands*

As we have seen in previous chapters, many techniques exist to analyze, annotate and further interpret multimedia content. However, these techniques remain limited with respect to the information they can automatically generate, which makes it important to provide tools that assist users in correcting, refining, and creating annotations manually. These tools are generally referred to as *multimedia annotation tools*. These tools also often offer retrieval functionalities, since the typical purpose of annotations is to provide the means for semantically browsing or querying a repository.

In this chapter we introduce two tools for multimedia annotation, namely the *Semantic Video Annotation Tool* (SVAT) and the *K-Space Annotation Tool* (KAT). Though technically quite similar, these tools target different problems. On the one hand, SVAT is aimed at professional users in the audiovisual media production and archiving domain, and is therefore driven by a quite specific use case. The underlying model is the *MPEG-7 Detailed Audiovisual Profile* (Bailer and Schallauer 2006), and the whole tool is designed to handle videos in terms of manual and automatic annotation techniques.

KAT, on the other hand, is supposed to be a framework for efficient and rich semantic annotation of multimedia content. In contrast to SVAT, its underlying model is the *Core Ontology for Multimedia* (see Chapter 9), covering not only videos, but all kinds of multimedia data. Similarly, KAT is not geared toward either a specific media type or a specific use case, but provides a framework for implementing annotation tools for various kinds of scenarios. KAT therefore abstracts every kind of functionality as a plugin, which includes not only analysis, but also visualization of content items or collections, retrieval, and annotation.

Multimedia Semantics: Metadata, Analysis and Interaction, First Edition.
Edited by Raphaël Troncy, Benoit Huet and Simon Schenk.
© 2011 John Wiley & Sons, Ltd. Published 2011 by John Wiley & Sons, Ltd.

www.it-ebooks.info

However, both tools try to facilitate efficient annotation by providing the means to integrate automatic and manual annotation in a framework that reduces the effort required of the user, while supporting the annotation of multimedia content with semantic metadata.

This chapter is structured as follows. In Section 13.1 we briefly discuss the current state of the art in multimedia annotation tools. We then introduce SVAT in Section 13.2 and discuss its core functionality. We continue with KAT in Section 13.3, where we focus on semi-automatic annotation and, in particular, discuss the implications of using a generic multimedia ontology like the COMM. We conclude the chapter in Section 13.4.

13.1 State of the Art

There are already several annotation tools available, both for images and videos. In this section we discuss the most prominent tools available, specifically with a focus on semantic media annotation tools as collected by the W3C Multimedia Semantics Incubator Group,[1] and on audiovisual media production and authoring tools. Obviously, these two areas are also targeted by both tools we present in this chapter.

One of the most prominent annotation tools on the Semantic Web is *Ontomat Annotizer* (Handschuh and Staab 2003), which was mainly focused on annotating web pages, but never emphasized the multimedia aspect involved in this task. Based on Ontomat the tool, *M-Ontomat Annotizer* (Bloehdorn et al. 2005) was developed, a tool that focuses on generating prototypical knowledge for the automatic annotation of multimedia content. However, the tool was never extended to a real annotation tool that also stored the annotations and made them available to other applications. KAT was heavily inspired by M-Ontomat Annotizer, and both built originally on Ontomat as a framework. We discuss this aspect in Section 13.3.1.

AKTive Media (Chakravarthy and Lanfranchi 2006) is a tool integrating automatic and manual annotation of multimedia content. It involves annotation with different domain ontologies, and includes methods to support a user in handling complex ontologies. It also supports region-level annotation and relational annotation, to be able to relate entities depicted within the content. During annotation it tries to gather additional contextual information by querying web services and showing related information, for example by using information extraction techniques.

A tool targeted at semantic image annotation and the publishing of the metadata on the web is *PhotoStuff* (Halaschek-Wiener et al. 2005). This tool does not support any other media types, nor does it integrate any advanced analysis. However, it does extract EXIF and Dublin Core metadata from image files if present and transforms it into an ontology-based representation.

Another interesting tool is *Caliph*, which is usually mentioned in conjunction with its retrieval engine *Emir* (Lux et al. 2003). The tool is based on MPEG-7 and is capable of extracting low-level features and indexing them for typical content-based image retrieval. Furthermore, it allows for semantic annotation using the appropriate MPEG-7 descriptor and provides a graphical user interface for creating relational annotations. However, since the tool is based on MPEG-7, the metadata created hardly integrates with other available metadata from the Semantic Web, and even linking metadata across different media items becomes hard.

[1] http://www.w3.org/2005/Incubator/mmsem/wiki/Tools_and_Resources

www.it-ebooks.info

We are not able to discuss all available tools in this chapter, and obviously other tools exist. A more complete overview is provided by the W3C Multimedia Semantics Incubator Group. Web 2.0 platforms might also be considered annotation tools, as they typically provide the means for tagging content with keywords or geolocations. However, Web 2.0 tools usually employ proprietary data models and do not publish their data in a common format. But also the Semantic Web tools discussed typically use custom ontologies for representing the content and annotations internally, which are often rather tool-specific. The exception is Caliph, which is based on MPEG-7, but then suffers from the problems discussed in Chapter 9.

13.2 SVAT: Professional Video Annotation

The Semantic Video Annotation Tool targets professional users in audiovisual media production and archiving and is designed to cover the use case described in Section 2.3. The tool can be integrated with automatic audiovisual and semantic content analysis tools (cf. Chapters 4 and 12). It allows the user to view and validate results of automatic audiovisual and semantic content analysis as well as to manually structure content and to add textual and semantic annotation. If content analysis has not been done, annotation can start 'from scratch'.

The SVAT architecture uses a modular plugin system. All the components described below are plugins which use a framework for data access and synchronization with other components. The components to be loaded are listed in the main application's configuration file, so it is easy to extend the functionality by simply developing new plugins. This also makes it possible to enable and disable functionality in a flexible way depending on the application context. In addition, the user interfaces of components can be hidden/shown and moved freely in accordance with the user's needs.

SVAT uses the MPEG-7 Detailed Audiovisual Profile (Bailer and Schallauer 2006) as metadata representation and both default input and output format. If content analysis results are available in the input MPEG-7 document, they can be displayed and modified. Import from and export to other formats can be supported via plugins.

13.2.1 User Interface

The SVAT user interface (see Figure 13.1) contains:

- a full-featured video player;
- the main timeline for navigation and specification of a temporal section of interest (local time period), and a number of timeline views that display information of the local time period such as a shot editor timeline for navigation and shot editing, visualization of extracted keyframes and stripe images, visual activity, camera motion and speech to text (ASR) transcript;
- the video structure tree view, which displays one or more high-level segmentations of the content and is also synchronized with the other views; and
- the semantic annotation view, which allows annotation of textual information and named entities at segment level.

www.it-ebooks.info

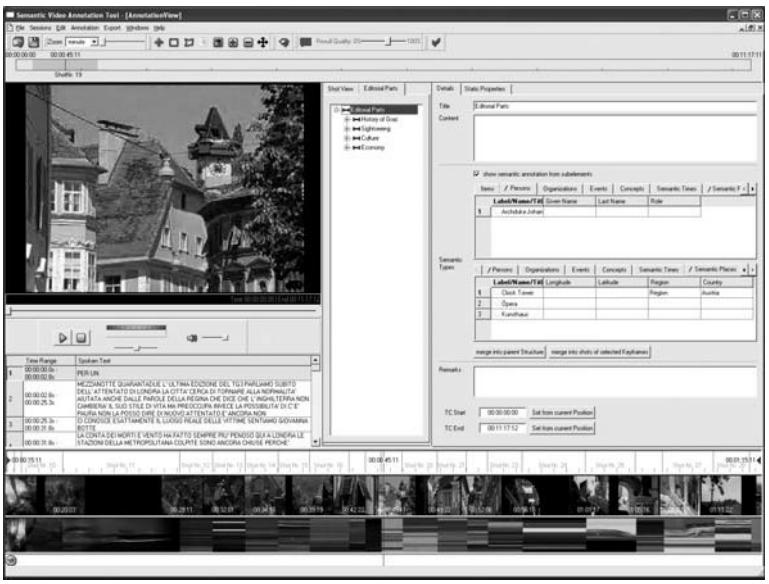

Figure 13.1 SVAT user interface.

All the components are synchronized with respect to the temporal position in the video; thus navigation is possible by clicking or clicking and dragging in any of the timeline views, by selecting segments in the tree view, by changing the temporal resolution of the timeline, or by selecting a phrase in the ASR transcript.

Main Timeline

The main timeline shows the time period from the start position to the end position of the video. By clicking the left mouse button at a specific position in the timeline, the actual position of the video can be changed. The position cursor (blue line) follows the mouse pointer as long as the left mouse button is held down. Each other window in the SVAT application will immediately be updated accordingly.

From the whole time period a temporal section of interest (local time period) can be selected in the toolbar. The zoom slider determines the length of the selected time period. This time period is indicated in the main timeline by a color marked range around the actual cursor. All windows below the main timeline display information only from this selected time period.

Shot Editor Timeline

Shots are the basic building blocks of the visual modality of a video. The visualization of the shots supports the user in efficiently navigating through the video and accessing

www.it-ebooks.info

certain time segments within the video. Shots are also an important basis for manual annotation, as many annotations are at the shot level. Knowledge of the shot structure facilitates building higher-level content structures such as scenes or chapters.

The shot editor timeline provides viewing and simple editing functionality for shot boundaries. If automatic content analysis has been done, the detected shot boundaries are shown in this view. Depending on the results of the automatic shot detection, missing shot boundaries (both hard cuts and gradual transitions) can be inserted and wrongly detected shot boundaries can be removed.

Keyframe, Stripe Image and Visual Activity Views

Keyframe extraction tools identify a number of frames in each shot, which are used as representative proxies of the shot's content and support effective navigation within the video. If automatically extracted keyframes are available, they are displayed in the keyframe view with the corresponding time code overlaid.

Stripe images are spatiotemporal representations of the visual content and hence are an effective way to navigate within video footage. A stripe image is created by extracting one column from each of the frames of the video and putting them together in temporal order. This visualization clearly shows shot boundaries as discontinuities, and allows the user to easily track camera motion and the movements of larger objects.

The visual activity view shows spatiotemporal activity of the video content over time as line graph, if it has been extracted during automatic content analysis. An average value for a short time period (typically a second) is displayed. The measure represents the frame to frame visual activity, caused by either global (camera) motion or local object motion.

Camera Motion View

This view displays the results of automatic camera motion detection and allows the user to add new camera motion annotations on a shot basis. The camera motion detection algorithm identifies segments containing salient camera movements and describes this motion both qualitatively and quantitatively. The original camera motion, however, cannot be reconstructed unambiguously from the analysis of the video, since for example it is not possible to discriminate a translation of the camera (e.g. a track left) from a rotation (e.g. a pan left), if the scenery is far away from the camera. The result of camera motion detection is thus described in terms of four types of camera motion (pan, tilt, zoom, roll) and a rough quantization of the amount of motion. Camera motion annotation can guide the user and support manual annotation. In addition, camera motion also conveys some semantics of the content; for example, zooming in on an object or person can be an indicator of its importance. The view displays the most dominant camera motion for each time segment if there is significant motion. The color indicates the type of motion and if sufficient space is available at the current zoom level, the type of motion is also displayed as text.

www.it-ebooks.info

Speech to Text Transcript

The ASR component displays the speech to text transcripts if available. The view is time-synchronized with the video player and selects the segment of the current time. Double-clicking on a row makes the video player jump to the start of the segment. The columns and contents of the view that are displayed can be configured.

Video Structure Tree

If the video has been processed with an automatic analysis tool, the video is automatically segmented into shots. This shot list is displayed by the video structure tree view, including the automatically extracted keyframes within the shots. If there are no shots present on loading, they can be manually created by other components within the application (like the shot editor timeline), or the video structure view can be used without shot information.

In order to structure the video, different views can be created. A view is the representation of a logical structure of the content (e.g. the scenes of a movie, the stories of a news broadcast, the takes in rushes). It is possible to create more than one view. This feature allows more than one segmentation to be described. For example, it can be used to display automatically generated segmentations, such as one based on audiovisual features and one on the lexical analysis of the speech transcript. Another use is to describe content structure in terms of different criteria, scenes of a movie according to a script, a segmentation based on appearance of a certain character, etc.

Within the created view, the video can be structured into arbitrary segments. The types of segments in the structure, as well as the accepted types of sub-segments, are defined by an MPEG-7 classification scheme loaded by SVAT. The content structure helps to group sections of the video that belong together and also allows semantic types to be annotated on the created segments.

13.2.2 Semantic Annotation

Detailed annotation can be done on shots or any other segment nodes in the tree structure. The user interface (Figure 13.2) allows the following detailed annotations to be edited: title, content synopsis, named entities and comments.

Named entities can be annotated in the Details tab. With the option to duplicate the semantic annotation control (with the context menu), two different types can be annotated and viewed separately. Annotation can be done on both semantic annotation controls and both are kept up to date.

There is also an option to show the semantic annotation from sub-elements (e.g. the shots within a structure). These entries are displayed as read-only within the Label/Name/Title column. It is also possible to merge the semantic annotation of one segment into the parent segment and also into the shots of selected keyframes that are displayed in a separate component within the application. The supported types of semantic entities correspond to those defined by the MPEG-7 SemanticDS (MPEG-7 2001): items (objects), persons, organizations, events, concepts, time and place. The properties of these types are also supported.

www.it-ebooks.info

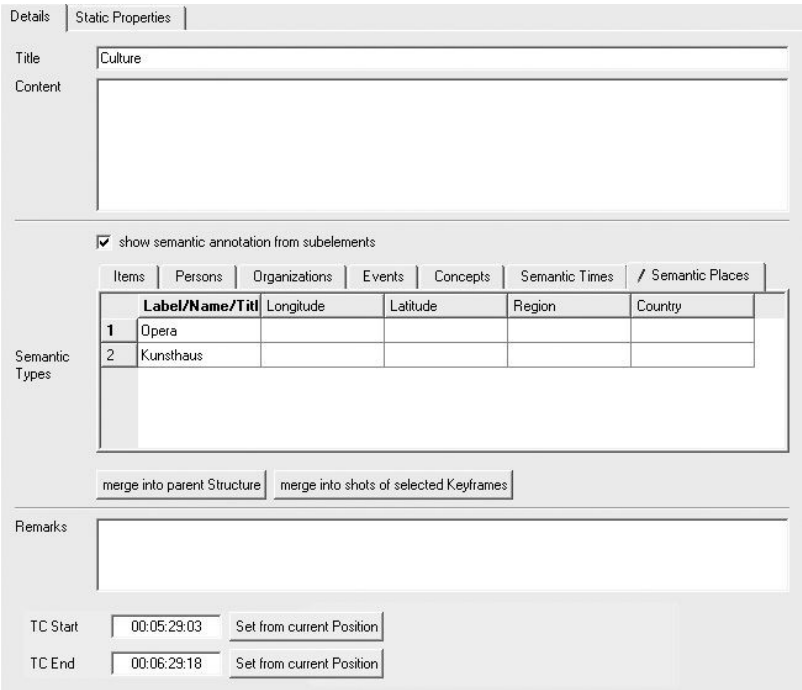

Figure 13.2 Detailed annotation interface for video segments.

Global annotation and production information are related to the entire content. The user interface (see Figure 13.3) allows one to set the title, subtitle, place, production date, creators, abstract, rights and information related to a content management system (CMS) (date imported, CMS reference and tape reference).

The user interface for the annotation of creators is flexible: one can set as many creators as needed, with each having a role that can be selected from a list of roles. Different creators can be specified with the same role.

Controlled vocabulary can be used for the annotation of named entities in both detailed and global annotation. In this case only the URI of the named entity is annotated in the MPEG-7 document, thus establishing a link to the controlled vocabulary. By default MPEG-7 classification schemes are used. SVAT also allows the linking of named entities stored in an external knowledge base. An interface for accessing KIM (Kiryakov et al. 2004) is available, but any other knowledge base can be integrated using the plugin concept.

13.3 KAT: Semi-automatic, Semantic Annotation of Multimedia Content

The K-Space Annotation Tool aims to provide a framework for semi-automatic, semantic annotation of multimedia content, with an emphasis on rich semantic annotations. Compared to pure keyword or concept annotation, rich annotation specifically denotes the

www.it-ebooks.info

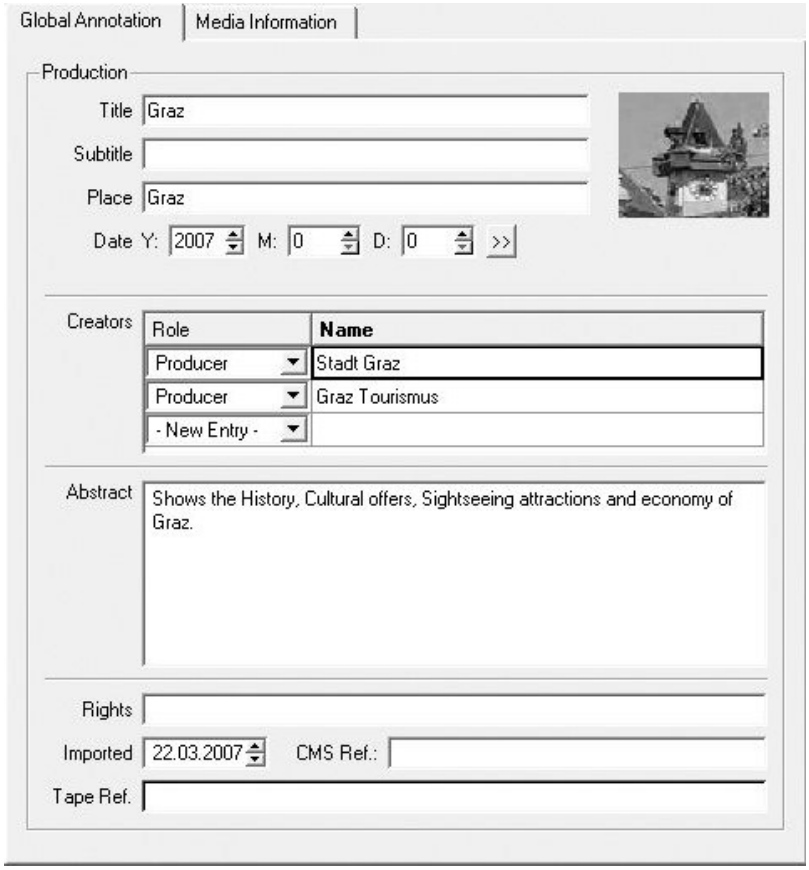

Figure 13.3 Global annotation dialogue.

employment of relations between concepts and individuals, for example, the explicit representation of events as first class objects, linking individuals to such events and describing the roles they play in the context of specific events. This provides the means not only to query for depicted objects, but also to query for content depicting very specific events, to allow for querying for related content and to exploit the relations between real-life entities for browsing multimedia repositories. A KAT screenshot showing an annotated image is shown in Figure 13.4.

KAT is designed with the well-known model–view–controller paradigm in mind. The model is provided by the Core Ontology for Multimedia (cf. Chapter 9) and an underlying RDF repository, Sesame 2.[2] The COMM provides an MPEG-7 based, generic model for multimedia content and can easily be specialized or extended for more specific scenarios. However, due to the formal, ontology-based modeling modules, the generic handling of special content is also possible, since the relationship between different content types is made explicit. For instance, for a medical application, X-ray images stored in a specialized

[2] http://www.openrdf.org

www.it-ebooks.info

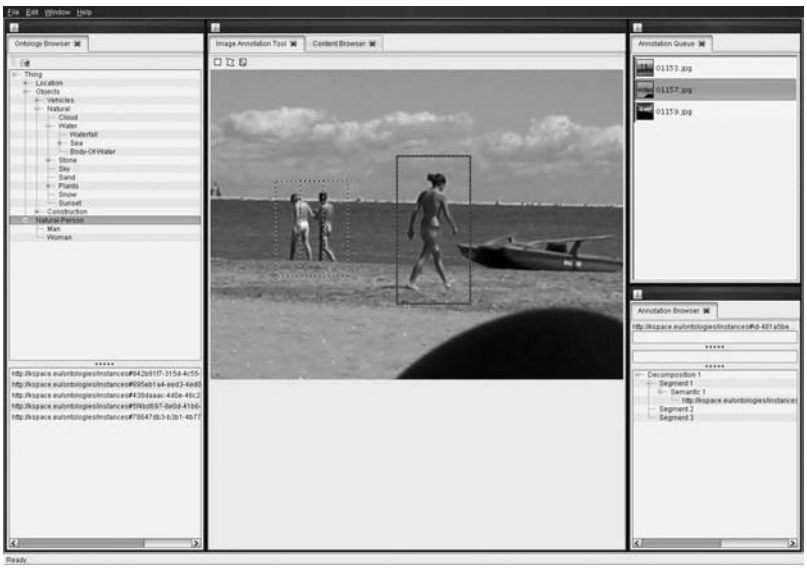

Figure 13.4 KAT screenshot during image annotation.

file format might be important. A generic content browser might not know how to display this file format, but it still knows that the files contain images and might display a generic image icon instead of a thumbnail. Further, a content-independent structure view, such as the annotation browser (cf. Section 13.3.3), can still display the decompositions and annotations, since the representation of a decomposition and an annotation is defined independently of the specific content type, but for images in general. Making these kinds of relations between concepts explicit, and abstracting from specific content and scenarios, is a great advantage of using the COMM and Semantic Web standards for the underlying model in KAT.

KAT is published as open source software under the terms of the LGPLv3 license and hosted on *launchpad* at the URL `http://launchpad.net/kat`. We further provide up-to-date documentation in the Semantic Multimedia Wiki at `http://semantic-multimedia.org/`.

13.3.1 History

The KAT code is based on the Ontomat-Annotatizer (Handschuh and Staab 2003) tool, but has been greatly extended and refactored. In the current version, basically all of the original Ontomat code has been modified, so that at the time of writing the relationship is historical rather than technical.

The development of KAT was inspired by M-Ontomat-Annotizer (Petridis et al. 2006), which provided the means to link ontology concepts with low-level features based on the aceMedia ontology infrastructure (Bloehdorn et al. 2005). These features were called visual prototypes and could be employed for automatic annotation of images. However, M-Ontomat-Annotizer never provided the means to actually annotate the content, either

www.it-ebooks.info

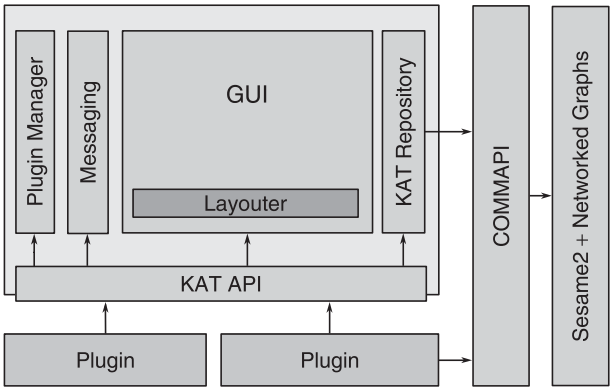

Figure 13.5 Overview of KAT architecture.

manual or automatically, or to retrieve content. The aim of KAT was to integrate feature extraction, automatic methods exploiting extracted features and semantic annotation into a single framework – to provide a tool for semi-automatic annotation, where user input is used by automatic methods to provide or propose annotations. The overall target is to work toward efficient annotation of multimedia content, that is, to reduce the time required by a user to annotate his content with rich semantics.

13.3.2 Architecture

An overview of KAT architecture is presented in Figure 13.5. KAT consists of three main building blocks. The core consists of a set of services and an API that is used by the plugins. The plugins implement all functionality relevant to the end-user, such as the processing of content, retrieval, and visualization. Finally, the COMMAPI and the underlying storage layer provide the common model to all plugins. Basically, all actions that might be performed within KAT are either visualization or modification of COMM objects.

The most important element of the core is the graphical user interface. Within it, plugins might register views that are displayed at certain locations. The available locations are determined by the layouter used. The default layouter provides four locations where a view might be displayed. These locations are indicated in Figure 13.6.

The remaining functionality provided by the core is as follows:

- the `MessageDispatcher`, which delivers messages and handles subscriptions to certain message types;
- the `PluginManager`, which controls import, loading and removal of plugins;
- the `KATRepository`, which provides convenient methods for creating, retrieving and storing COMM objects and other metadata that might be added to the RDF repository;
- the KAT API, which provides the interface for plugins to the KAT core functionality.

We discuss these components below in greater detail.

A plugin implements the functionality that is relevant to an end user: the analysis, visualization, modification or retrieval of content and its metadata. The core component

www.it-ebooks.info

Figure 13.6 Available view positions in the default layout.

of a plugin is the controller, the only class that is directly instantiated by the KAT core. The basic functionality provided by the KAT core to a plugin is as follows:

- Installation, loading, execution and removal of plugins based on stored preferences or user request.
- Registration of views that are displayed in one of the locations of the current layout.
- Execution of workers. Workers correspond to resource-consuming tasks (like most analyses), where usually only one should run at a time in order to not slow down a typical desktop machine too much. Workers are executed by the KAT core sequentially.

In other words, the only required component for a plugin is the controller. The internal structure of a plugin is not further defined. However, controllers, views and workers have to implement the corresponding interfaces.

The model and storage layer of KAT is based on the Core Ontology for Multimedia (Arndt et al. 2008) and the Sesame 2 RDF repository in conjunction with the Networked Graphs Sail (Schenk and Staab 2008). KAT is basically a tool to visualize and modify COMM graphs and the corresponding content. Therefore, the underlying model for all components, and specifically the plugins, is the COMM it self. Communication among components is based on passing around URIs of COMM objects. The semantics of the URI is clearly defined by the COMM based on the available metadata in the repository. The type of message then clearly defines what should be done with the content item or COMM graph, respectively. The Networked Graphs Sail is used to structure the RDF repository into parts that are easier to handle, to allow for the addition of provenance and administrative metadata, and to provide transparent read and write access to more than one underlying repository at once.

www.it-ebooks.info

13.3.3 Default Plugins

KAT contains a set of default plugins for image annotation. The most important plugins are the *Ontology Browser, Content Browser*, and *Image Annotation Tool*. The Ontology Browser can load and display OWL ontologies. It might be used to create instances of ontology concepts, but does not currently support modification of the ontology. Using drag and drop, the ontology browser is capable of annotating arbitrary content items. The content browser is used to import content, and to display results of queries against the repository. Queries are currently triggered by selecting an ontology concept or individual. In that case, the content browser will display all content annotated with the concept or individual selected. Importing content is done via a file browser. The selected content is added to the repository and displayed with thumbnails in a list. Using the content browser, content can be selected for analysis by one of the installed analysis plugins, chosen from a drop-down box. Double-clicking a content item opens it in the respective annotation tool. For images this is the Image Annotation Tool. The plugin allows for creating regions using bounding boxes or polygons and for annotating these regions using drag and drop. Furthermore, it can visualize regions already present.

Within K-Space, several analysis plugins implementing functionality described in Chapters 10–12 have been implemented, enabling detection of highlights and events in videos, identification of semantic concepts in images, and the like. Furthermore, the *Audio-Visual Annotation Tool* allows the playback of video and audio content, visualization of shots and keyframes, and semantic annotation of videos and audio files.

13.3.4 Using COMM as an Underlying Model: Issues and Solutions

As already explained, the COMM is the common model for all KAT components. It provides the formal model that is used to classify and annotate the content and provides the basic metadata required by plugins to decide whether content should be processed or how it should be processed. From a programming point of view, the COMMAPI abstracts from the storage in the underlying Sesame repository. However, depending on the plugin, more complex queries or custom COMM descriptors might require direct interaction with the repository. In order to acquire an instance of the internal repository correctly configured with respect to remote repositories, one can use the appropriate methods of the `KATRepository` object. This is the internal repository manager, which takes care of configuring the repositories correctly and provides some primitives to be used for constructing correct queries or storing to the correct remote repository.

The Networked Graphs Sail provides transparent access to multiple repositories, and is used to structure the content using graphs for provenance tracking. Since we also require write access to remote repositories, KAT assumes that every remote repository is also a Sesame 2 installation, since generic write access to remote repositories is not available for RDF at the time of writing.

In order to handle transparent read and write access to remote repositories, the `DistributedSail`, which is part of the Networked Graphs package, is employed. Each repository is added to the configuration of the distributed sail and mapped to a namespace. All graphs within this namespace are retrieved from the given repository, and write operations are also performed on that repository. This, however, requires the creation

www.it-ebooks.info

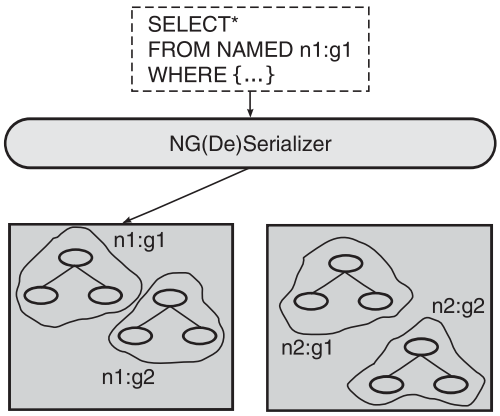

Figure 13.7 Using named graphs to map COMM objects to repositories.

of adequate graphs. In order to achieve this transparently, we provide the `NGSerializer` and `NGDeserializer` which handle (de)serialization of COMM graphs. These classes implement the `Serializer` and `Deserializer` interfaces provided as part of the COMMAPI.

The implementation creates a namespace for the repository and when a new COMM object is serialized, which is not yet present in the repository, a dedicated named graph is created within the namespace. This allows the mapping of COMM objects to the repository storing them. This approach is illustrated in Figure 13.7. For querying and writing we just have to specify the graph name, and the `DistributedSail` will map the graphs to the correct repository.

In a similar fashion we can also store ontologies. We create a named graph for each imported ontology, and store appropriate metadata for display. This allows easy selection of all available ontologies, including the source repository. We can also easily extract all statements belonging to a single ontology.

As specified in Chapter 9, COMM is designed around the numerous MPEG-7 patterns used to describe spatial, temporal, spatio-temporal and media source decompositions of multimedia contents into segments. The main issue in implementing COMM for annotating media documents (e.g. for audiovisual documents) for a reasonable video duration is the memory requirements. In Table 13.1 the number of triples generated for the same video exported with different metadata is presented. Here COMM is constructed: (1) with media information (MediaURI, MediaTimeDescriptor, etc.); (2) with media information and semantic annotation; (3) with media information, semantic annotation (for whole video, events, spatial objects) and MPEG-7 low-level visual features (Colour Layout Descriptor, CLD).

In Table 13.2 the number of triples generated for different videos with different numbers of shots and keyframes is presented. The COMM object is constructed to contain metadata about the media (e.g. its title, its mime type, its duration) and about low-level visual features (represented in MPEG-7) which are temporally aligned to the video (media time).

The number of triples generated by COMM is highly influenced by both number of shots (subsequently keyframes) and the amount of metadata associated with each shot/keyframe.

www.it-ebooks.info

Table 13.1 Number of RDF triples for MPEG-7 export of the same video with different metadata

Parameter	Inst. 1	Inst. 2	Inst. 3
Total duration (s)	218	218	218
No. of shots	164	164	164
No. of keyframes	164	164	164
Total triples generated	41 454	50 337	70 681

Table 13.2 Number of RDF triples for MPEG-7 export of different videos with the same metadata

Parameter	Video 1	Video 2	Video 3	Video 4
Total duration (s)	305	236	198	290
No. of shots	17	129	131	170
No. of keyframes	17	129	131	170
Total triples generated	7 618	55 666	56 524	73 256

MPEG-7 facilitates the browsing and retrieval of audiovisual content by defining partitions and decompositions and variations of the audiovisual materials. Thus, COMM also enables the description of detailed decomposition of a multimedia document. The granularity of the decomposition can range from a region in a keyframe to an abstract description of a shot (without any keyframes). Thus, increasing the number of triples associated with a COMM object requires an increase in the memory required to handle the object.

However, this can be resolved by exploiting the decomposition patterns defined in COMM (inspired by MPEG-7). The following is the pseudocode for constructing a COMM object in a hierarchical approach.

```
construct VideoData for complete video
export VideoData using COMM Serializer
for shot = 1 to N do
  construct VideoData for shot segments
  for keyframes = 1 to M do
    construct VideoSegmentTemporalDecomposition for all keyframes extracted
    from the shot
  end for
  construct MediaTimeDescriptor for shot segments
  export VideoData and MediaTimeDescriptor to RDF repository using COMM
  Serializer.
end for
apply a patch in order to link complete video uri to shot decompositions.
```

As may be noted, the COMM object is always constructed representing a single 'VideoData' or 'VideoSegmentTemporalDecomposition', thus significantly reducing the memory constraints. At the time of writing, the approach was used to export TRECVid MPEG-7

www.it-ebooks.info

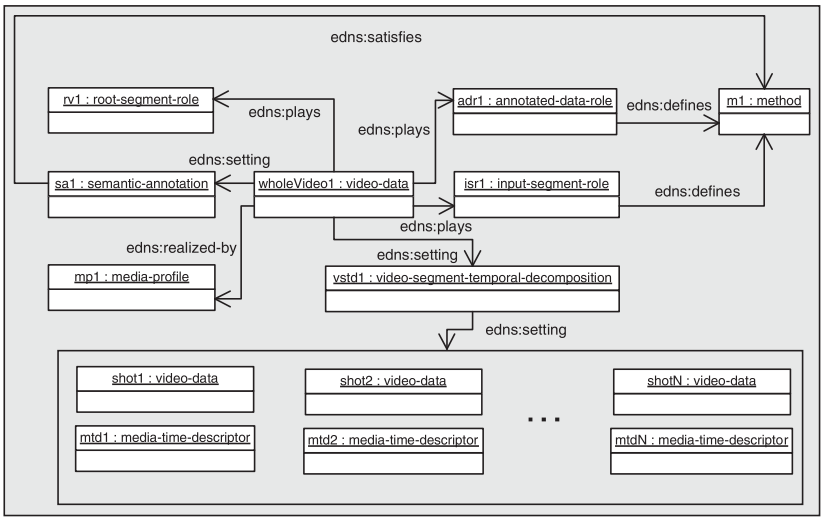

Figure 13.8 COMM video decomposition for whole video.

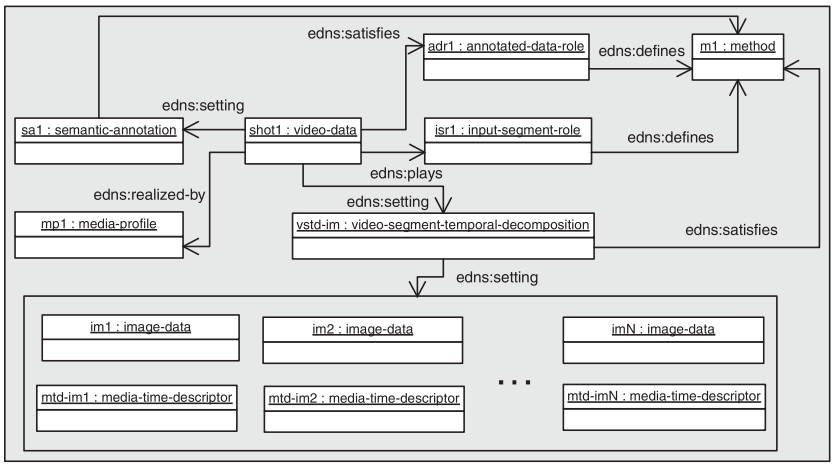

Figure 13.9 COMM video decomposition for video segment.

data to the RDF repository using COMM. The two-stage decomposition represented in the pseudocode is presented in Figures 13.8 and 13.9 for complete video and shot segments, respectively.

13.3.5 Semi-automatic Annotation: An Example

As mentioned earlier, the aim of KAT is to facilitate the annotation of multimedia content with rich semantic metadata using a combination of automatic and manual techniques.

www.it-ebooks.info

It is well known that creating rich semantic annotations manually is time-consuming and tedious, and thus means are desired to reduce this effort. We refer to techniques of this kind of as *efficient annotation*, which refers to the employment of methods to support the user in the task of annotating content. Depending on the scenario, the concrete techniques or tools used for such a task will vary, and KAT aims to provide the framework for linking these different tools in order to implement applications that can be used within more specific scenarios. In this section we will discuss two scenarios that have been used to inspire the development of KAT.

To support personal content management we can see clear benefits in using automatic image annotation techniques, such as presented in Chapters 11 and 12. As we have shown in a recent publication (Saathoff and Staab 2008), well-performing classifiers can be trained with relatively few training examples. Within KAT this might be used as an automatic analysis plugin that is iteratively trained using manual annotations provided by the user. Obviously, this kind of annotation plugin will usually ship with a set of already available classifiers. However, since it is known that training generic classifiers is hard, an iterative training will 'optimize' the classifiers for the specific content of each user. The manual annotations either might result from a set of manually labeled images, or could be drawn from automatic annotations that have been corrected by the user. On top of these basic annotations, further inferencing can be performed to infer additional annotations. Nevertheless, only simple and basic concepts might be recognized using such automatic methods, such as *sky* or *beach scene*. While this might be interesting in certain situations, annotations of this kind provide only very basic semantics (see also Jaimes and Chang 2000), and are far away from the user's perceived semantics when he browses his own content. In order to support the user in generating higher-level semantic annotations, one can, for instance, use clustering to group content with respect to certain events. A user might then annotate an event in one batch. This offers two advantages. The user does not have to annotate every single image and events are known to be an important primitive when organizing and browsing multimedia (Scherp et al. 2008). In order to free the user from handling ontology concepts directly, natural language processing might also be used, as discussed in Dasiopoulou et al. (2007).

A second scenario is the semi-automatic annotation of sports footage, in which different methods play a key role, such as the complementary resource analysis (see Chapter 12), highlight detection or key-frame extraction (cf. Chapters 4 and 5). The video analysis plugin described earlier is capable of producing temporal boundaries of the video depicting salient events. The shots are further analyzed to extract significant information from such shots, known as keyframes. In terms of high-level semantics, the shots depict an event and the keyframe represents the agents participating in these events. Thus, in order to annotate both events and agents participating in an event, the COMMAPI enables semantic labeling of both VideoSegment and Image classes. Using the video analysis tool, the shots depicting an event and keyframe representing the agents participating in an event can be semantically labeled with instances of any domain ontology. In order to describe this annotation, a two-tier VideoSegmentTemporalDecomposition is necessary. The first decomposition has to be applied to the whole video content in order to decompose it into video shots. Subsequently, the keyframes have to be identified by attaching one VideoSegmentTemporalDecomposition to each video shot. This decomposition objects only contain one segment (a still image), the keyframe.

www.it-ebooks.info

Additionally, the complementary resource analysis provides a structuring of the content in terms of important events (e.g. a shot on goal) and semantic annotations extracted from text. So, in addition to knowing that some highlight is present at a certain point, the algorithm also proposes the kind of event. This might then be refined or corrected by a user. In the case of a foul the algorithm might, for instance, not know who committed the foul and who was fouled, but it does detect that there is a foul and that two people were involved. The user now only has to specify the role a certain entity plated within this event, which can be done using drag and drop, for instance.

13.4 Conclusions

In this chapter we have presented an overview of available multimedia annotation tools and discussed two of them in detail. The Semantic Video Annotation Tool is targeted at professional audiovisual media production and archiving and provides a framework, based on MPEG-7, for annotating audiovisual media. It integrates different methods for automatic structuring of content and provides the means to semantically annotate the content. The K-Space Annotation Tool is a framework for semi-automatic, semantic annotation of multimedia content based on the COMM. The aim here is to provide a media-independent framework that might be specialized for certain scenarios. Building on the COMM facilitates interoperability between plugins even when specialized descriptors are introduced. The formal model of the COMM explicitly allows for handling specialized descriptions in a generic fashion using the inheritance relations.

Both tools try to tackle similar problems, namely semantically annotating multimedia content in a efficient manner, using different approaches. While one tool employs an existing standard and focuses on a specific, rather broad domain, the other tries to employ Semantic Web standards and formal ontologies across different domains and media.

We have discussed issues that arise when using formal ontologies for data-intensive applications such as multimedia annotation and provided solutions to this problem. However, compared to the known standards and data formats such as MPEG-7 (see also Chapter 8), the greater flexibility provided by Semantic Web standards comes at a cost of increased complexity and currently still reduced scalability. Both certainly have to be tackled in future research, in order to provide tools and frameworks that allow for the employment of Semantic Web standards in real applications.

www.it-ebooks.info

www.it-ebooks.info