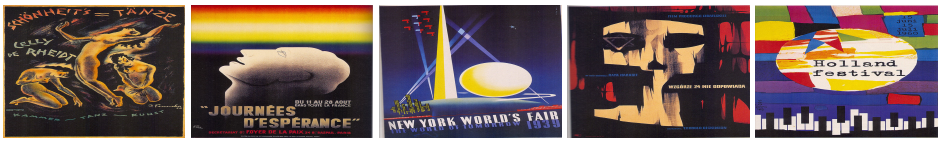
6. content annotation

video annotation requires a logical approach to story telling

learning objectives

After reading this chapter you should be able to explain the difference between content and meta information, to mention relevant content parameters for audio, to characterize the requirements for video libraries, to define an annotation logic for video, and to discuss feature extraction in samples of musical material.

Current technology does not allow us to extract information automatically from arbitrary media objects. In these cases, at least for the time being, we need to assist search by annotating content with what is commonly referred to as meta-information. In this chapter, we will look at two more media types, in particular audio and video. Studying audio, we will learn how we may combine feature extraction and meta-information to define a data model that allows for search. Studying video, on the other hand, will indicate the complexity of devising a knowledge representation scheme that captures the content of video fragments. Concluding this chapter, we will discuss an architecture for feature extraction for arbitrary media objects.



1

6.1 audio

The audio media type covers both spoken voice and musical material. In this section we will discuss audio signal, stored in a raw or compressed (digital) format, as well as similarity-based retrieval for musical patterns.

In general, for providing search access to audio material we need, following Subrahmanian (1998), a data model that allows for both mdeta-data (that is

information about the media object) and additional attributes of features, that we in prinicple obtain from the media object itself, using feature extraction.

audio data model

- *meta-data* – describing content
- *features* – using feature extraction

As an example of audi meta-data, consider the (meta-data) characterization that may be given for opera librettos.

example

singers – (Opera,Role,Person)
score – ...
transcript – ...

For signal-based audio content, we have to perform an analysis of the audio signal for which we may take parameters such as frequency, velocity and amplitude. For the actual analysis we may have to break up the signal in small windows, along the time-axis. Using feature extraction, we may characterize (signal-based) properties such as indicated below.

feature extraction

- *intensity* – watts/m^2
- *loudness* – in decibels
- *pitch* – from frequency and amplitude
- *brightness* – amount of distortion

For a more detailed treatment of signal-based audio content description, consult Subrahmanian (1998).

In the following we will first give an overview of musical search facilities on the web and then we will discuss similarity-based retrieval of musical patterns in somewhat more depth in the section on *research directions*, In section 6.3, we will have a closer look at feature extraction for arbitrary media types.

musical similarity

Although intuitively obvious, how can we characterize musical simularity? And perhaps more importantly, how can we compute the extent to which one piece of music or a melody line is similar to another piece of music or melody line. As concerns musical content, at least for most genres, it appears that

According to Selfridge (1998), we should focus primarily on *melody*, since

> *"It is melody that makes music memorable: we are likely to recall a tune long after we have forgotten its text."*

Other features, content-based as well as descriptive, may however be used as additional filters in the proces of retrieval.

Melodic searching and matching has been explored mainly in the context of bibliographic tools and for the analysis of (monophonic) repertories Hewlett and Selfridge-Field (1998). As described in section , many of these efforts have been made available to the general public through the Web. Challenges for the near future are, however, to provide for melodic similarity matching on polyphonic works, and retrieval over very large databases of musical fragments.

In this section we will look in somewhat more detail at the problem of melodic similarity matching. In particular, we will discuss representational issues, matching algorithms and additional analysis tools that may be used for musical information retrieval.

melodic similarity Consider the musical fragment *Twinkle, twinkle little star* (known in the Dutch tradition as *"Altijd is Kortjakje ziek"*), which has been used by Mozart for a series of variations Mozart (1787). Now, imagine how you would approach establishing the similarity between the original theme and these variations. As a matter of fact, we discovered that exactly this problem had been tackled in the study reported in Mongeau and Sankoff (1990), which we will discuss later. Before that, we may reflect on what we mean by the concept of a *melody*. In the aforementioned variations the original melody is disguised by, for example, decorations and accompaniments. In some variations, the melody is distributed among the various parts (the left and right hand). In other variations, the melody is only implied by the harmonic structure. Nevertheless, for the human ear there seems to be, as it is called in Selfridge (1998), a *'prototypical'* melody that is present in each of the variations.

When we restrict ourselves to pitch-based comparisons, melodic similarity may be established by comparing profiles of pitch-direction (up, down, repeat) or pitch contours (which may be depicted graphically). Also, given a suitable representation, we may compare pitch-event strings (assuming a normalized pitch representation such as position within a scale) or intervallic contours (which gives the distance between notes in for example semitones). Following Selfridge (1998), we may observe however that the more general the system of representation, the longer the (query) *string* will need to be to produce meaningful discriminations. As further discussed in Selfridge (1998), recent studies in musical perception indicate that pitch-information without durational values does not suffice.

representational issues Given a set of musical fragments, we may envisage several reductions to arrive at the (hypothetical) prototypical melody. Such reductions must provide for the elimination of confounds such as rests, repeated notes and grace notes, and result in, for example, a pitch-string (in a suitable representation), a duration profile, and (possibly) accented note profiles and harmonic reinforcement profiles (which capture notes that are emphasized by harmonic changes). Unfortunately, as observed in Selfridge (1998), the problem of which reductions to apply is rather elusive, since it depends to a great extent on the goals of the query and the repertory at hand.

As concerns the representation of pitch information, there is a choice between a base-7 representation, which corresponds with the position relative to the tonic in the major or minor scales, a base-12 representation, which corresponds with a division in twelve semitones as in the chromatic scale, and more elaborate encodings, which also reflect notational differences in identical notes that arise through the use of accidentals. For MIDI applications, a base-12 notation is most suitable, since the MIDI note information is given in semitone steps. In addition to relative pitch information, octave information is also important, to establish the rising and falling of melodic contour.

When we restrict ourselves to directional profiles (up, down, repeat), we may include information concerning the slope, or degree of change, the relation of the current pitch to the original pitch, possible repetitions, recurrence of pitches after intervening pitches, and possible segmentations in the melody. In addition, however, to support relevant comparisons it seems important to have information on the rhythmic and harmonic structure as well.

3

example(s) – *napster*

Wasn't it always your dream to have all your music free? Napster[1] was the answer. (But not for long.) Napster is, as we learn in the WikiPedia[2], *an online music service which was originally a file sharing service created by Shawn Fanning. Napster was the first widely-used peer-to-peer music sharing service,*

[1] www.napster.com
[2] en.wikipedia.org/wiki/Napster

and it made a major impact on how people, especially college students, used the Internet. Its technology allowed music fans to easily share MP3 format song files with each other, thus leading to the music industry's accusations of massive copyright violations. The service was named Napster after Fanning's nickname. However, Napster has been forced to become commercial. So the question is: is there life after napster?[3] Well, there is at least open source![4]

research directions – *musical similarity matching*

An altogether different approach at establishing melodic similarity is proposed in Mongeau and Sankoff (1990). This approach has been followed in the Meldex system McNab et al. (1997), discussed in section . This is a rather technical section, that may be skipped on first reading. The approach is different in that it relies on a (computer science) theory of finite sequence comparison, instead of musical considerations. The general approach is, as explained in Mongeau and Sankoff (1990), to search for an optimal correspondence between elements of two sequences, based on a distance metric or measure of dissimilarity, also known more informally as the *edit-distance*, which amounts to the (minimal) number of transformations that need to be applied to the first sequence in order to obtain the second one. Typical transformations include *deletion*, *insertion* and *replacement*. In the musical domain, we may also apply transformations such as *consolidation* (the replacement of several elements by one element) and *fragmentation* (which is the reverse of consolidation). The metric is even more generally applicable by associating a weight with each of the transformations. Elements of the musical sequences used in Mongeau and Sankoff (1990) are pitch-duration pairs, encoded in base-12 pitch information and durations as multiples of 1/16th notes.

The matching algorithm can be summarized by the following recurrence relation for the dissimilarity metric. Given two sequences $A = a_1, \ldots, a_m$ and $B = b_1, \ldots, b_n$ and $d_{ij} = d(a_i, b_j)$, we define the distance as

$$d_{ij} = min \begin{cases} d_{i-1,j} + w(a_i, 0) & \text{deletion} \\ d_{i,j-1} + w(0, b_j) & \text{insertion} \\ d_{i-1,j-1} + w(a_i, b_j) & \text{replacement} \\ d_{i-k,j-1} + w(a_{i-k+1}, \ldots, a_i, b_j).\ 2 \leqslant k \leqslant i & \text{consolidation)} \\ d_{i-1,j-k+1} + w(a_i, b_{j-k+1}, \ldots b_j)\ 2 \leqslant k \leqslant j & \text{fragmentation} \end{cases}$$

with

$$\begin{array}{ll} d_{i0} = d_{i-1,0} + w(a_i, 0),\ i \geqslant 1 & \text{deletion} \\ d_{0j} = d_{0,j-1} + w(0, b_i),\ j \geqslant 1 & \text{insertion} \end{array}$$

and $d_{00} = 0$. The weigths $w(\_, \_)$ are determined by the degree of dissonance and the length of the notes involved.

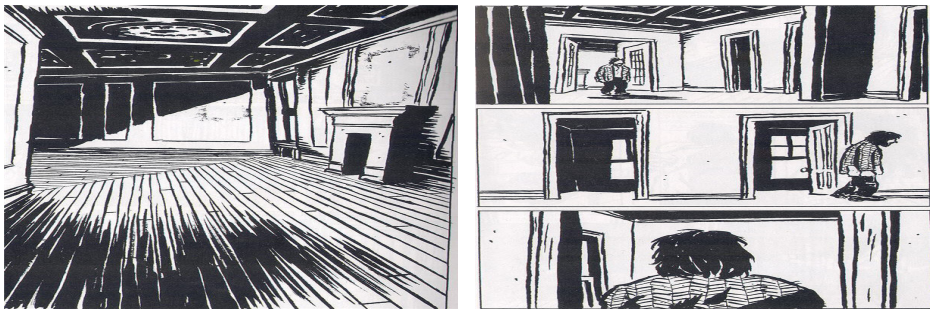
The actual algorithms for determining the dissimilarity between two sequences uses dynamic programming techniques. The algorithm has been generalized to

[3] www.afternapster.com
[4] opennap.sourceforge.net

look for matching phrases, or subsequences, within a sequence. The complexity of the algorithm is $O(mn)$, provided that a limit is imposed on the number of notes involved in consolidation and fragmentation.

Nevertheless, as indicated in experiments for the Meldex database, the resulting complexity is still forbidding when large databases are involved. The Meldex system offers apart from the (approximate) dynamic programming algorithm also a state matching algorithm that is less flexible, but significantly faster. The Meldex experiments involved a database of 9400 songs, that were used to investigate six musical search criteria: (1) exact interval and rhythm, (2) exact contour and rhythm, (3) exact interval, (4) exact contour, (5) approximate interval and rhythm, and (6) approximate contour and rhythm. Their results indicate that the number of notes needed to return a reasonable number of songs scales logarithmically with database size McNab et al. (1997). It must be noted that the Meldex database contained a full (monophonic) transcription of the songs. An obvious solution to manage the complexity of searching over a large database would seem to be the storage of prototypical themes or melodies instead of complete songs.

indexing and analysis There are several tools available that may assist us in creating a proper index of musical information. One of these tools is the Humdrum system, which offers facilities for metric and harmonic analysis, that have proven their worth in several musicological investigations Huron (1997). Another tool that seems to be suitable for our purposes, moreover since it uses a simple pitch-duration, or *piano-roll*, encoding of musical material, is the system for metric and harmonic analysis described in Temperley and Sleator (1999). Their system derives a metrical structure, encoded as hierarchical levels of equally spaced beats, based on preference-rules which determine the overall likelihood of the resulting metrical structure. Harmonic analysis further results in (another level of) *chord spans* labelled with roots, which is also determined by preference rules that take into account the previously derived metrical structure. As we have observed before, metrical and harmonic analysis may be used to eliminate confounding information with regard to the 'prototypical' melodic structure.



6.2 video

Automatic content description is no doubt much harder for video than for any other media type. Given the current state of the art, it is not realistic to expect content description by feature extraction for video to be feasible. Therefore,to realize content-based search for video, we have rely on some knowledge representation schema that may adequately describe the (dynamic) properties of video fragments.

In fact, the description of video content may reflect the story-board, that after all is intended to capture both time-independent and dynamically changing properties of the objects (and persons) that play a role in the video.

In developing a suitable annotation for a particular video fragment, two questions need to be answered:

video annotation

- what are the interesting aspects?
- how do we represent this information?

Which aspects are of interest is something you have to decide for yourself. Let's see whether we can define a suitable knowledge representation scheme.

One possible knowledge representation scheme for annotating video content is proposed in Subrahmanian (1998). The scheme proposed has been inspired by knowledge representation techniques in Artificial Intelligence. It captures both static and dynamic properties.

video content

> video v, frame f
> f has associated objects and activities
> objects and activities have properties

First of all, we must be able to talk about a particular video fragment v, and frame f that occurs in it. Each frame may contain objects that play a role in some activity. Both objects and activities may have properties, that is attributes that have some value.

property

> property: name = value

As we will see in the examples, properties may also be characterized using predicates.

Some properties depend on the actual frame the object is in. Other properties (for example sex and age) are not likely to change and may considered to be frame-independent.

object schema

> (fd,fi) – frame-dependent and frame-independent properties

Finally, in order to identify objects we need an object identifier for each object. Summing up, for each object in a video fragment we can define an *object instance*,

that characterizes both frame-independent and frame-dependent properties of the object.

object instance: (oid,os,ip)

- *object-id* – oid
- *object-schema* – os = (fd,fi)
- *set of statements* – ip: name = v and name = v IN f

Now, with a collection of object instances we can characterize the contents of an entire video fragment, by identifying the frame-dependent and frame-independent properties of the objects.

Look at the following example, borrowed from Subrahmanian (1998) for the *Amsterdam Drugport* scenario.

frame	objects	*frame-dependent properties*
1	Jane	has(briefcase), at(path)
-	house	door(closed)
-	briefcase	
2	Jane	has(briefcase), at(door)
-	Dennis	at(door)
-	house	door(open)
-	briefcase	

In the first frame Jane is near the house, at the path that leads to the door. The door is closed. In the next frame, the door is open. Jane is at the door, holding a briefcase. Dennis is also at the door. What will happen next?

Observe thatwe are using predicates to represent the state of affairs. We do this, simply because the predicate form *has(briefcase)* looks more natural than the other form, which would be *has = briefcase*. There is no essential difference between the two forms.

Now, to complete our description we can simply list the frame-independent properties, as illustrated below.

object	*frame-independent properties*	value
Jane	age	35
	height	170cm
house	address	...
	color	brown
briefcase	color	black
	size	40 x 31

How to go from the tabular format to sets of statements that comprise the object schemas is left as an (easy) exercise for the student.

Let's go back to our *Amsterdam Drugport* scenario and see what this information might do for us, in finding possible suspects. Based on the information given in the example, we can determine that there is a person with a briefcase, and another person to which that briefcase may possibly be handed. Whether this is the case or not should be disclosed in frame 3. Now, what we are actually looking for is the possible exchange of a briefcase, which may indicate a drug transaction.

So why not, following Subrahmanian (1998), introduce another somewhat more abstract level of description that deals with *activities*.

activity

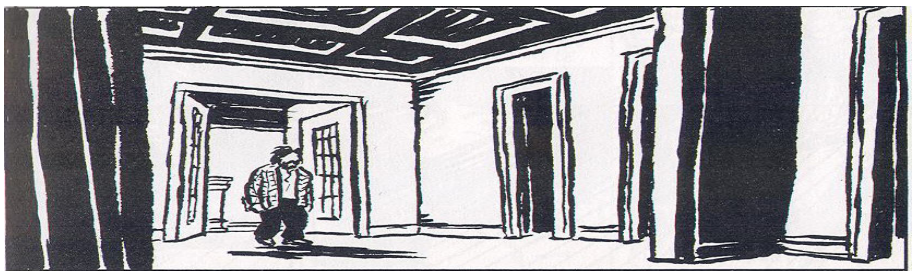
- activity name – id
- statements – *role* = *v*

An activity has a name, and consists further simply of a set of statements describing the *roles* that take part in the activity.

example

{ giver : Person, receiver : Person, item : Object }
giver = Jane, receiver = Dennis, object = briefcase

For example, an *exchange* activity may be characterized by identifying the *giver*, *receiver* and *object* roles. So, instead of looking for persons and objects in a video fragment, you'd better look for activities that may have taken place, by finding a matching set of objects for the particular roles of an activity. Consult Subrahmanian (1998) if you are interested in a further formalization of these notions.



5

video libraries

Assuming a knowledge representation scheme as the one treated above, how can we support search over a collection of videos or video fragments in a video library.

What we are interested in may roughly be summarized as

video libraries

which videos are in the library
what constitutes the content of each video
what is the location of a particular video

Take note that all the information about the videos or video fragments must be provided as meta-information by a (human) librarian. Just imagine for a moment how laborious and painstaking this must be, and whata relief video feature extraction would be for an operation like *Amsterdam Drugport*.

To query the collection of video fragments, we need a query language with access to our knowledge representation. It must support a variety of retrieval

operations, including the retrieval of segments, objects and activities, and also property-based retrievals as indicated below.

query language for video libraries

- *segment retrievals – exchange of briefcase*
- *object retrievals* – all people in v:[s,e]
- *activity retrieval* – all activities in v:[s,e]
- *property-based* – find all videos with object oid

Subrahmanian (1998) lists a collection of video functions that may be used to extend SQL into what we may call VideoSQL. Abstractly, VideoSQL may be characterized by the following schema:

VideoSQL

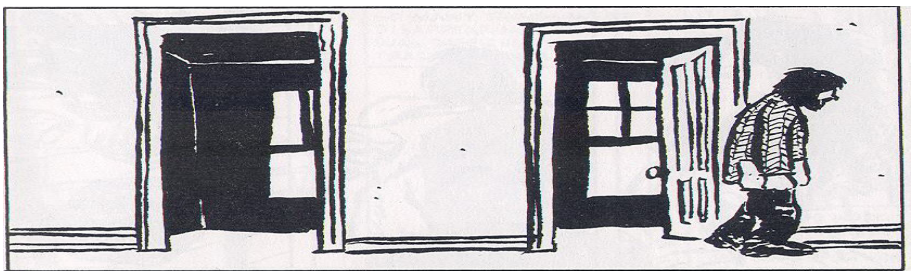
```
SELECT – v:[s,e]
FROM – video:<source><V>
WHERE – term IN funcall
```

where *v:[s,e]* denotes the fragment of video v, starting at frame s and ending at frame e, and *term IN funcall* one of the video functions giving access to the information about that particular video. As an example, look at the following VideoSQL snippet:

example

```
SELECT  vid:[s,e]
FROM video:VidLib
WHERE (vid,s,e) IN VideoWithObject(Dennis) AND
      object IN ObjectsInVideo(vid,s,e) AND
      object != Dennis AND
      typeof(object) = Person
```

Notice that apart from calling video functions also constraints can be added with respect to the identity and type of the objects involved.



example(s) – *video retrieval evaluation*

The goal of the TREC[5] conference series is to encourage research in information retrieval by providing a large test collection, uniform scoring procedures, and a forum for organizations interested in comparing their results. Since 2003 their is an independent *video* track devoted to research in automatic segmentation, indexing, and content-based retrieval of digital video. In the TRECVID[6] 2004 workshop, thirty-three teams from Europe, the Americas, Asia, and Australia participated. Check it out!

7

research directions– *presentation and context*

Let's consider an example. Suppose you have a database with (video) fragments of news and documentary items. How would you give access to that database? And, how would you present its contents? Naturally, to answer the first question, you need to provide search facilities. Now, with regard to the second question, for a small database, of say 100 items, you could present a list of videos thatb matches the query. But with a database of over 10.000 items this will become problematic, not to speak about databases with over a million of video fragments. For large databases, obviously, you need some way of visualizing the results, so that the user can quickly browse through the candidate set(s) of items.

Pesce (2003) provide an interesting account on how *interactive maps* may be used to improve search and discovery in a (digital) video library. As they explain in the abstract:

> *To improve library access, the Informedia Digital Video Library uses automatic processing to derive descriptors for video. A new extension to the video processing extracts geographic references from these descriptors.*
>
> *The operational library interface shows the geographic entities addressed in a story, highlighting the regions discussed in the video through a map display synchronized with the video display.*

[5] trec.nist.gov
[6] www-nlpir.nist.gov/projects/trecvid

So, the idea is to use geographical information (that is somehow available in the video fragments themselves) as an additional descriptor, and to use that information to enhance the presentation of a particular video. For presenting the results of a query, candidate items may be displayed as icons in a particular region on a map, so that the user can make a choice.

Obviously, having such geographical information:

> *The map can also serve as a query mechanism, allowing users to search the terabyte library for stories taking place in a selected area of interest.*

The approach to extracting descriptors for video fragments is interesting in itself. The two primary sources of information are, respectively, the spoken text and graphic text overlays (which are common in news items to emphasize particular aspects of the news, such as the area where an accident occurs). Both speech recognition and image processing are needed to extract information terms, and in addition natural language processing, to do the actual 'geocoding', that is translating this information to geographical locations related to the story in the video.

Leaving technical details aside, it will be evident that this approach works since news items may relevantly be grouped and accessed from a geographical perspective. For this type of information we may search, in other words, with three kinds of questions:

- *what* – content-related
- *when* – position on time-continuum
- *where* – geographic location

and we may, evidently, use the geographic location both as a search criterium and to enhance the presentation of query results.

mapping information spaces Now, can we generalize this approach to other type of items as well. More specifically, can we use maps or some spatial layout to display the results of a query in a meaningful way and so give better access to large databases of multimedia objects. According to Dodge and Kitchin (2002), we are very likely able to do so:

> *More recently, it has been recognized that the process of spatialization – where a spatial map-like structure is applied to data where no inherent or obvious one does exist – can provide an interpretable structure to other types of data.*

Actually, we are taking up the theme of *visualization*, again. In Dodge and Kitchin (2002) visualizations are presented that (together) may be regarded as an *atlas of cyberspace*.

atlas of cyberspace

> *We present a wide range of spatializations that have employed a variety of graphical techniques and visual metaphors so as to provide striking and powerful images that extend from two dimension 'maps' to three-dimensional immersive landscapes.*

As you may gather from chapter 7 and the *afterthoughts*, I take a personal interest in the (research) theme of *virtual reality interfaces for multimedia information systems*. But I am well aware of the difficulties involved. It is an area that is just beginning to be explored!

8

6.3 feature extraction

Manual content annotation is laborious, and hence costly. As a consequence, content annotation will often not be done and search access to multimedia object willnot be optimal, if it is provided for at all. An alternative to manual content annotation is (semi) automatic feature extraction, which allows for obtaining a description of a particular media object using media specific analysis techniques.

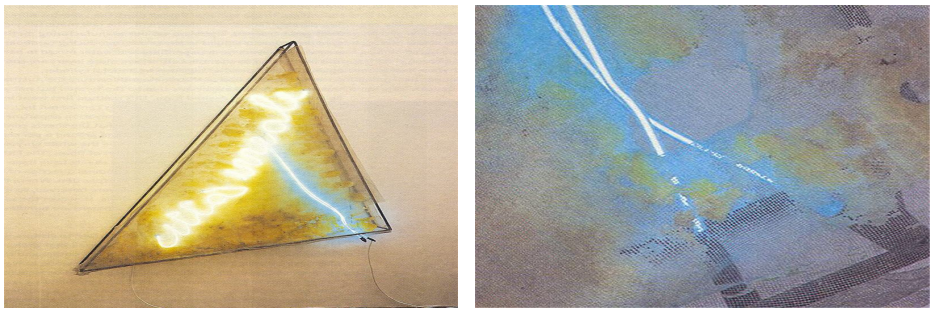
The Multimedia Database Research group at CWI has developed a framework for feature extraction to support the *Amsterdam Catalogue of Images* (ACOI). The resulting framework for feature extraction is known as the ACOI framework, Kersten et al. (1998).

The ACOI framework is intended to accomodate a broad spectrum of classification schemes, manual as well as (semi) automatic, for the indexing and retrieval of arbitrary multimedia objects. What is stored are not the actual multimedia objects themselves, but structural descriptions of these objects (including their location) that may be used for retrieval.

The ACOI model is based on the assumption that indexing an arbitrary multimedia object is equivalent to deriving a grammatical structure that provides a namespace to reason about the object and to access its components. However there is an important difference with ordinary parsing in that the lexical and grammatical items corresponding to the components of the multimedia object must be created dynamically by inspecting the actual object. Moreover, in general, there is not a fixed sequence of lexicals as in the case of natural or formal languages. To allow for the dynamic creation of lexical and grammatical items the ACOI framework supports both *black-box* and *white-box* (feature) detectors. Black-box detectors are algorithms, usually developed by a specialist in the media domain, that extract properties from the media object by some form of analysis. White-box detectors, on the other hand, are created by defining logical or mathematical expressions over the grammar itself. Here we will focus on black-box detectors only.

The information obtained from parsing a multimedia object is stored in a database. The feature grammar and its associated detector further result in updating the data schemas stored in the database.

formal specification Formally, a feature grammar G may be defined as $G = (V, T, P, S)$, where V is a collection of variables or non-terminals, T a collection of terminals, P a collection of productions of the form $V \rightarrow (V \cup T)$ and S a start symbol. A token sequence ts belongs to the language $L(G)$ if $S \xrightarrow{*} ts$. Sentential token sequences, those belonging to $L(G)$ or its sublanguages $L(G_v) = (V_v, T_v, P_v, v)$ for $v \in (T \cup V)$, correspond to a complex object C_v, which is the object corresponding to the parse tree for v. The parse tree defines a hierarchical structure that may be used to access and manipulate the components of the multimedia object subjected to the detector. See Schmidt et al. (1999) for further details.



9

anatomy of a feature detector

As an example of a feature detector, we will look at a simple feature detector for (MIDI encoded) musical data. A special feature of this particular detector,that I developed while being a guest at CWI, is that it uses an intermediate representation in a logic programming language (Prolog) to facilitate reasoning about features.

The hierarchical information structure that we consider is defined in the grammar below. It contains only a limited number of basic properties and must be extended with information along the lines of some musical ontology, see Zimmerman (1998).

feature grammar

detector song; *# # to get the filename*
detector lyrics; *# # extracts lyrics*
detector melody; *# # extracts melody*
detector check; *# # to walk the tree*

```
atom str name;
atom str text;
atom str note;

midi: song;

song: file lyrics melody check;

file: name;

lyrics: text*;
melody: note*;
```

The start symbol is a *song*. The detector that is associated with *song* reads in a MIDI file. The musical information contained in the MIDI file is then stored as a collection of Prolog facts. This translation is very direct. In effect the MIDI file header information is stored, and events are recorded as facts, as illustrated below for a *note_on* and *note_off* event.

```
event('twinkle',2,time=384, note_on:[chan=2,pitch=72,vol=111]).
event('twinkle',2,time=768, note_off:[chan=2,pitch=72,vol=100]).
```

After translating the MIDI file into a Prolog format, the other detectors will be invoked, that is the *composer*, *lyrics* and *melody* detector, to extract the information related to these properties.

To extract relevant fragments of the melody we use the melody detector, of which a partial listing is given below.

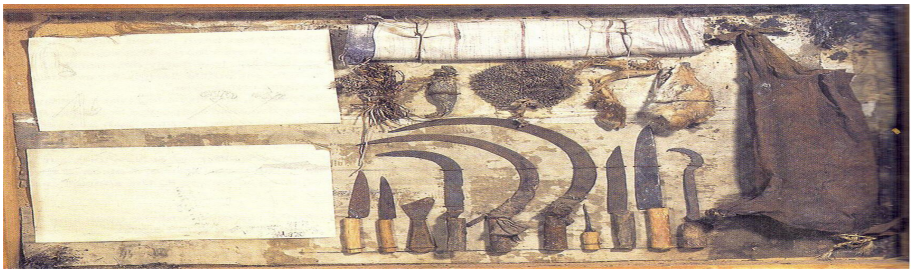
melody detector

```
int melodyDetector(tree *pt, list *tks ){
char buf[1024]; char* _result;
void* q = _query;
int idq = 0;

  idq = query_eval(q,"X:melody(X)");
  while ((_result = query_result(q,idq)) ) {
        putAtom(tks,"note",_result);
        }
  return SUCCESS;
}
```

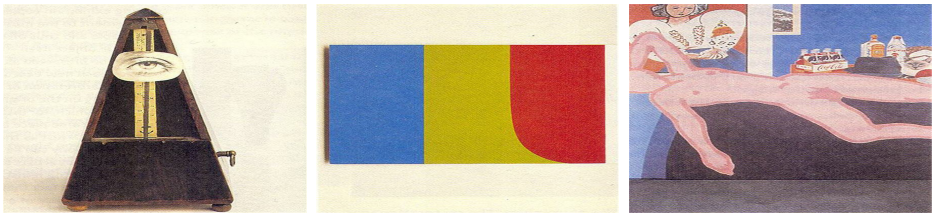
The embedded logic component is given the query `X:melody(X)`, which results in the notes that constitute the (relevant fragment of the) melody. These notes are then added to the tokenstream. A similar detector is available for the lyrics.

Parsing a given MIDI file, for example *twinkle.mid*, results in updating the database.



10

implementation The embedded logic component is part of the *hush* framework, Eliens (2000). It uses an object extension of Prolog that allows for the definition of native objects to interface with the MIDI processing software written in C++. The logic component allows for the definition of arbitrary predicates to extract the musical information, such as the melody and the lyrics. It also allows for further analysis of these features to check for, for example, particular patterns in the melody.



11

example(s) – *modern art: who cares?*

The artworks shown above are taken from Hummelen and Sillé (1999), which bundles the experiences and insights resulting from studying the preservation of contemprary art, under the title: *modern art, who cares?* This project was a precursor to the INCCA[7] that provided the input to our *multimedia casus*, which is introduced in chapter 10.

Both the INCCA project and the related *Open Archives Initiative*[8], focus on making meta-information available on existing resources for the preservation of contemporary art and cultural heritage in general, including reports, case studies and recordings of artworks, that is images, videos and artists interviews.

12

[7] www.incca.org
[8] www.openarchives.org

research directions– *media search*

There is a wealth of powerful search engines on the Web. Technically, search engines rely either on classification schemes (as for example Yahoo) or content-based (keyword) indexing (as for example Excite or AltaVista). Searching on the Web, nowadays, is moderately effective when text-based documents are considered. For multimedia objects (such as images or music) existing search facilities are far less effective, simply because indexing on category or keywords can not so easily be done automatically. In the following we will explore what search facilities there are for music (on the web). We will first give some examples of search based on keywords and categories, then some examples of content-based search and finally we will discuss a more exhaustive list of musical databases and search facilities on the Web. All search facilities mentioned are listed online under *musical resources*.

keywords and categories For musical material, in particular MIDI, there are a number of sites that offer search over a body of collected works. One example is the Aria Database, that allows to search for an aria part of an opera based on title, category and even voice part. Another example is the MIDI Farm, which provides many MIDI-related resources, and also allows for searching for MIDI material by filename, author, artist and ratings. A category can be selected to limit the search. The MIDI Farm employs voting to achieve collaborative filtering on the results for a query. Search indexes for sites based on categories and keywords are usually created by hand, sometimes erreonously. For example, when searching for a Twinkle fragment, Bach's variations for Twinkle were found, whereas to the best of our knowledge there exist only Twinkle variations by Mozart Mozart (1787). The Digital Tradition Folksong Database provides in addition a powerful lyrics (free text) search facility based on the AskSam search engine. An alternative way of searching is to employ a meta-search engine. Meta-search engines assist the user in formulating an appropriate query, while leaving the actual search to (possibly multiple) search engines. Searching for musical content is generally restricted to the lyrics, but see below (and section).

content-based search Although content-based search for images and sound have been a topic of interest for over a decade, few results have been made available to the public. As an example, the MuscleFish Datablade for Informix, allows for obtaining information from audio based on a content analysis of the audio object. As far as content-based musical search facilities for the Web are concerned, we have for example, the Meldex system of the New Zealand Digital Library initiative, an experimental system that allows for searching tunes in a folksong database with approximately 1000 records, McNab et al. (1997). Querying facilities for Meldex include queries based on transcriptions from audio input, that is humming a tune! We will discuss the approach taken for the Meldex system in more detail in research directions section, to assess its viability for retrieving musical fragments in a large database.

music databases In addition to the sites previously mentioned, there exist

several databases with musical information on the Web. We observe that these databases do not rely on DBMS technology at all. This obviously leads to a plethora of file formats and re-invention of typical DBMS facilities. Without aiming for completeness, we have for example the *MIDI Universe*, which offers over a million MIDI file references, indexed primarily by composer and file length. It moreover keeps relevant statistics on popular tunes, as well as a hot set of MIDI tunes. It further offers access to a list of related smaller MIDI databases. Another example is the aforementioned Meldex system that offers a large collection of tunes (more than 100.000), of which a part is accessible by humming-based retrieval. In addition text-based search is possible against file names, song titles, track names and (where available) lyrics. The Classical MIDI Archive is an example of a database allowing text-based search on titles only. Results are annotated with an indication of "goodness" and recency. The Classical Themefinder Database allows extensive support for retrieval based on (optional) indications of meter, pitch, pitch-class, interval, semi-tone interval and melodic contour, within a fixed collection of works arranged according to composer and category. The index is clearly created and maintained manually. The resulting work is delivered in the MuseData format, which is a rich (research-based) file format from which MIDI files can be generated, Selfridge (1997). A site which collects librarian information concerning music resources is the International Inventory of Music Resources (RISM), which offers search facilities over bibliographic records for music manuscripts, librettos and secondary sources for music written after c.a. 1600. It also allows to search for libraries related to the RISM site. Tune recognition is apparently offered by the Tune Server. The user may search by offering a WAV file with a fragment of the melody. However, the actual matching occurs against a melodic outline, that is indications of rising or falling in pitch. The database contains approx. 15.000 records with such pitch contours, of which one third are popular tunes and the rest classical themes. The output is a ranked list of titles about which the user is asked to give feedback.

discussion There is great divergence in the scope and aims of music databases on the Web. Some, such as the RISM database, are the result of musicological investigations, whereas others, such as the MIDI Farm, are meant to serve an audience looking for popular tunes. With regard to the actual search facilities offered, we observe that, with the exception of Meldex and the Tune Server, the query facilities are usually text-based, although for example the Classical Themefinder allows for encoding melodic contour in a text-based fashion.

6.4 development(s) – expert recommendations

Leaving all responsibility for interaction to the user is usually not a good choice, in particular when an information system contains complex, highly interrelated information. Despite the wealth of recommendation systems, it still seems to be an open problem how to generate a related collection of recommendations, that is an organized sequence of recommended items that me be used as a guided

tour, for example an overview of artworks and related information from a museum collection.

In Eliens & Wang (2007) we wrote: there is a great wealth of recommender systems, and a daunting number of techniques for producing recommendations, based on content, user behavior or social groups. See the AAAI 2004 Tutorial[9] on recommender systems and techniques for an (extensive) overview.

In van Setten (2005) a distinction is made between the following types of prediction techniques:

- social-based – dependent on (group) rating of item(s)
- information-based – dependent on features of item(s)
- hybrid methods – combining predictors

Social-based prediction techniques include collaborative filtering (CF), item-item filtering, popularity measures, etcetera. Information-based prediction techniques include information filtering, case-based reasoning and attribute or feature comparison. Finally, as hybridization techniques, van Setten (2005) distinguishes between weighted combination, switching, mixed application and meta-approaches such as feature combination and cascaded application.

The approach we presented in Eliens & Wang (2007), the R3 framework, for *rate – regret – recommend*, has aspects of social-based as well as information-based methods and may be characterized as hybrid since it uses a weighting scheme to select between experts for advice.

For clarity, it is worthwhile to delineate briefly what we understand by the phrases *rate, recommend, regret*, and how the R3 framework fits within the wider scope of recommendation techniques:

definition(s)

- *rating* – a value representing a user's interest
- *recommendation* – item(s) that might be of interest to the user
- *regret* – a function to measure the accuracy of recommendations

In our approach, we initially proceeded from the assumption that a rating is already present, and more in particular a rating that implies a sequential order on the presentation of a (limited) number of items. Later, however, we will explore how to relax this assumption and apply the R3 framework to sequences that are generated on the basis of content-based user preferences, to allow for an incremental adaptation of recommendations.

An interesting way to generate guided tours based on user tracking and expert advice, is suggested by a variant of decision theory introduced in Cesa-Bianchi and Lugosi (2006).

In classical prediction theory a prediction is a sequence of elements $x_1, x_2, \ldots$ that results from a stationary stochastic process. The risk of the prediction is taken to be the expected value of the accumulated *loss* function, measuring the discrepancy between predicted values and actual outcomes. Cesa-Bianchi and Lugosi (2006) introduce a variant of prediction theory in which no assumption

[9] www.dfki.de/∼jameson/aaai04-tutorial

is made with respect to the nature of the source of predictions. Instead, the *forecaster* is considered to be an entity that gives a prediction for an element based on *advice* of one or more *experts*. These experts might be actual sequences stored in a database. The deviation of the forecaster with the actual outcome is measured using a *regret* function, and the prediction task may hence be formulated as minimimizing the *regret* function by choosing the best expert for advice for each element of a prediction sequence.

For example, for the prediction of a bitstring of length n, the forecaster is a vector of n expert indices, that give advice for the bitvalue, *0* or *1*, in that position. In the general case, in which we have no information on the error rate of the experts' advice, we may use a weighting factor $0 \leqslant \beta_i \leqslant 1$ for each expert i, to indicate the credibility of the experts' advice. After each prediction, obtained by taking the majority decision of the experts, according to the weighting scheme, we may verify which experts fail to give the right advice, and decrease their weight, thus eliminating the influence of their advice in the long run.

In digital dossiers to be discussed in chapter 10, we explored the use of guided tours as a means to present the information in a story-like way, relieving the user of the often cumbersome task to interact, Dodge and Kitchin (2000). Guided tours, in the digital dossier, may take one of the following forms:

guided tour(s)

- automated (viewpoint) navigation in virtual space,
- an animation explaining, for example, the construction of an artwork, or
- the (narrative) presentation of a sequence of concept nodes.

In practice, a guided tour may be constructed as a combination of these elements, interweaving, for example, the explanation of concepts, or biographic material of the artist, with the demonstration of the positioning of an artwork in an exhibition space.

A pre-condition for the construction of guided tours based on user tracking is that navigation consists of a small number of discrete steps. This excludes the construction of arbitrary guided tours in virtual space, since it is not immediately obvious how navigation in virtual space may be properly discretized. In this case, as we will discuss later, a guided tour may be constructed using a programmed agent showing the user around.

For navigation in the concept graph, as well as for the activation of the media presentation gadget, the discretization pre-condition holds, and a guided tour may be composed from a finite number of discrete steps, reflecting the choice of the user for a particular node or interaction with the presentation gadget.

For example, in the *abramovic* dossier, the user has the option to go from the *Main* node to either *Artworks*, *Video Installations* or *Interviews*, and from there on further to any of the items under the chosen category. Tracking the actual sequences of choices of a user would suffice to create a guided tour, simply by re-playing all steps.

To obtain more interesting tours, we may track the navigation behavior of several experts for a particular task, for example retrieving information about an artwork installation. In case the experts disagree on a particular step in the

tour, we may take the majority decision, and possibly correct this by adjusting the weight for one or more experts. When we have a database of tours from a number of experts, we may offer the user a choice of tours, and even allow to give priority to one or more of his/her favorite experts, again simply by adjusting the weighting scheme.

As a technical requirement, it must be possible to normalize interaction sequences, to eliminate the influence of short-cuts, and to allow for comparison between a collection of recordings. For the actual playback, as a guided tour, a decision mechanism is needed that finds the advice at each decision point, from each expert, to select the best step, according to a decision rule that takes the weighting scheme into account.

In a more mathematical way, we may state that for each node n we have a successor function *S(n)*, that lists the collection of nodes connected with n, which we may write as $S(n) = \{n_1, \ldots, n_k\}$, where the suffix $i \leq k$ is an arbitrary integer index over the successor nodes. To take a history of navigation into account, we let s$\overline{p}$ be a string of integers, representing the choices made, encoding the navigation path. So, for a node $n_{\overline{p}}$, with history $\overline{p}$, the collection of successor nodes is $S_{\overline{p}}(n) = \{n_{\overline{p}1}, \ldots, n_{\overline{p}k}\}$.

Now assume that we have a weight function w, that assigns to each expert e_i a weight $0 \leq \beta_i \leq 1$, indicating the relevance of expert i. Then for a particular node n we may assume to have an advice $\alpha_i = x$, with weight β_i and x in $S(n)$. If an expert has no advice for this node, we may simply assume its weight to be 0. For a collection of experts, the final advice will be $\alpha(n) = \alpha_i(n)$ with weight β_i and $w(e_i) > w(e_j)$ for $i \neq j$. If no such advice $\alpha_i(n)$ exists, we may query the user to decide which expert has preference, and adapt the weights for the experts accordingly. This procedure can be easily generalized to nodes $n_{\overline{p}}$ with history $\overline{p}$.

To cope with possible shortcuts, for example when a choice is made for a node at three levels deep, we must normalize the path, by inserting the intermediate node, in order to allow for comparison between experts.

Now assume that we have expert navigation paths with cycles, for example $n_{\overline{p}} \rightarrow n_{\overline{p}1} \rightarrow n_{\overline{p}13}$, where actually $n_{\overline{p}} = n_{\overline{p}13}$, which happens when we return to the original node. In general such cycles should be eliminated, unless they can be regarded as an essential subtour. However, in this case, they could also be offered explicitly as a subtour, if they have length $\geqslant 4$. When offering guided tours for which several variants exist, we may allow the user to simply assign weights to each of the experts from which we have a tour, or allow for incrementally adjusting the weight of the experts, as feedback on the actual tour presented.

In the CHIP[10] project (Cultural Heritage Information Personalization), the aim is to develop a recommender system that generates a collection of artworks in accordance with the users' preferences based on the rating of a small sample of artworks. The properties on which the recommendation is based include *period*, *artist*, and *genre*. The recommender system will also be used to generate guided tours, where apart from the already mentioned properties the *location* (the proximity in the actual museum) will be taken into account.

[10] www.chip-project.org

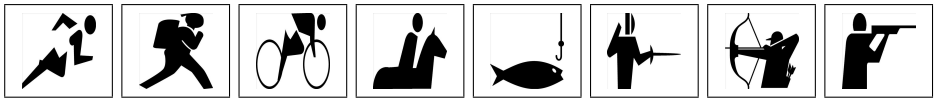
Using a weighting scheme on the properties, that is a difference metric on the properties, a graph can be created, giving a prioritized accessibility relation between each artwork and a collection of related artworks. By changing the weight for one of the properties, for example *location*, in case the tour is generated for the actual museum, the priority ordering may be changed, resulting in a different tour.

In contrast to the successor function for nodes in the concept graph of the digital dossier, we may assume to have a weighted successor function $S_w(n) = (n_1, \omega_1), \ldots, (n_k, \omega_k)$, with $\omega_i = w(n_i)$ the weight defined by the relevance of the node n_i, with respect to the attributes involved. In a similar way as for the digital dossier, user tracking may be deployed to incrementally change the weight of the arcs of the graph, reflecting the actual preference of the user when deviating from an existing guided tour.

recommendation(s) in Second Life Our virtual campus in Second Life already allows for performing simple statistics, by recording the presence of users at particular spots in the virtual world, using sensors and listeners installed in 3D objects. Since the LSL script-based counters appear to be rather volatile, tracking data are sent to a web server and stored in a database. This mechanism can easily be extended to a more encompassing form of user tracking, recording for a particular user not only presence at particular spots, but also the duration of presence, the actual proximity to objects, and the proximity to other users, as well as explicitly spoken comments or actions such as the donation of (Linden) money.

This does of course not explain nor how ratings come into existence, nor what features are considered relevant, or even how guided tours should be generated. However, as we have demonstrated in Ballegooij and Eliens (2001), see section 8.2, based on a rudimentary tagging scheme, we may in response to a query generate a guided tour taking the topographical constraints of the virtual world into account, for example to make a user familiar with the (virtual replica of the) actual workspace. It seems that this approach can be generalized to one that uses alternative descriptive methods, as long as they support feature-based information retrieval[11].

Obviously, both user tracking and recommendations may be fruitfully used in the realization of serious (corporate) games, as well as to support exploratory activity in non-serious games and (corporate) awareness systems.



[11] www.cs.vu.nl/∼eliens/research/rif.html

questions

content annotation

1. (*) How can video information be made accessible? Discuss the requirements for supporting video queries.

concepts

2. What are the ingredients of an *audio data model*
3. What information must be stored to enable search for video content?
4. What is *feature extraction*? Indicate how feature extraction can be deployed for arbitrary media formats.

technology

5. What are the parameters for *signal-based (audio) content*?
6. Give an example of the representation of *frame-dependent* en *frame-independent* properties of a video fragment.
7. What are the elements of a query language for searching in video libraries?
8. Give an example (with explanation) of the use of *VideoSQL*.

projects & further reading As a project, think of implementing musical similarity matching, or developing an application retrieving video fragments using a simple annotation logic.

You may further explore the construction of media repositories, and finding a balance between automatic indexing, content search and meta information.

For further reading I advice you to *google* recent research on video analysis, and the online material on search engines[12].

the artwork

1. works from Weishar (1998)
2. faces – from www.alterfin.org, an interesting site with many surprising interactive toys in *flash*, javascript and html.
3. mouth – Annika Karlson Rixon, entitled *A slight Acquaintance*, taken from a theme article about the body in art and science, the Volkskrant, 24/03/05.
4. story – page from the comic book version of *City of Glass*, Auster (2004), drawn in an almost tradional style.
5. story – frame from Auster (2004).
6. story – frame from Auster (2004).
7. story – frame from Auster (2004).
8. *white on white* – typographical joke.
9. modern art – *city of light* (1968-69), Mario Merz, taken from Hummelen and Sillé (1999).

[12] www.searchtools.com/tools/tools-opensource.html

10. modern art – *Marocco* (1972), Krijn Griezen, taken from Hummelen and Sillé (1999).
11. modern art – *Indestructable Object* (1958), Man Ray, *Blue, Green, Red I* (1964-65), Ellsworth Kelly, *Great American Nude* (1960), T. Wesselman, taken from Hummelen and Sillé (1999).
12. signs – sports, van Rooijen (2003), p. 272, 273.

Opening this chapter are examples of design of the 20th century, posters to announce a public event like a theatre play, a world fair, or a festival. In comparison to the art works of the previous chapter, these designs are more strongly *expressive* and more simple and clear in their *message*. Yet, they also show a wide variety of styles and rethorics to attract the attention of the audience. Both the faces and the mouth are examples of using body parts in contemporary art. The page of the comic book version of *City of Glass*, illustrates how the 'logic' of a story can be visualised. As an exercise, try to annoyaye the sequence of frames from the *City of Glass* can be described using the annotation logic you learned in this chapter. The modern art examples should interesting by themselves.