



Deliverable D.6.1

Report on software development

Month 24

Project Start:	01/02/2007
Project Duration:	36 Months
Priority area	2.3.6
Contract No.:	FP6-045547
Website:	

Due-Date:	15/01/2009
Delivery:	15/02/2009
Lead Partner:	CVC
Project Leader	UvA
Dissemination Level:	Public
Status:	Draft
Approved:	
Version:	0.9

Table of Contents

Table of Contents	2
1. Introduction	4
2. Work done	4
3. MPEG7	6
3.1. What is Mpeg7	6
3.2. Mpeg7 java library	7
3.2.1. Implemented part	7
3.3. Test: From logic predicates to Mpeg7	9
3.4. Description of the implemented part of Mpeg7	13
3.4.1. The Header	13
3.4.2. The video path	13
3.4.3. Video Segment List	13
3.4.4. Video Segment	14
3.4.5. Video Segment description	14
3.4.6. Segment Time	14
3.4.7. MediaTimePoint	14
3.4.8. MediaDuration	15
4. Integrated software prototype	16
4.1. Software	18
4.1.1. Shot Segmentation	18
4.1.2. Visual Feature Annotator	19
4.1.3. Visual Classifier Software	19
4.1.4. Audio Classifier	19
4.1.5. Demo GUI	19
4.2. Data	19
4.2.1. Video Data	19
4.2.2. Shot segmentation	19
4.2.3. Visual features	19
4.2.4. Visual/Audio Concepts	19
4.2.5. Annotations	19
4.3. Implementation	20
4.3.1. System requirements	20
5. System Tests	21
6. Second Prototype	25
6.1. New Execution Scheme	25
6.2. Human Detection	25
6.3. OCR	29
6.3.1. OCR pre-processing module	30
7. Thesaurus of detectors	32
7.1. Machine-And-User Tagging	34
7.1.1. Search-based Machine Tagging	34
7.1.2. Tag Relevance Estimation from User Tagging	35
7.2. Results	36
8. Conclusions	36

1. Introduction

The objective of this work package is to consolidate and validate the textual, audio, speech, motion and visual features in a common representation and indexing structure to enable wide and simple usage by the other system components. The consolidation is split into two parts: one part contains the processing of video into shots, audio segments, and speech recognition; the other part of the consolidated software is the thesaurus of audiovisual detectors of semantic concepts.

Thus, on the one hand, the integrated system will be tested in the labs and evaluated with respect to the accuracy of both interpretation and retrieval, and the degree of interaction. On the other hand, since the representation of audiovisual features has to be used in the query ontology, the most effective way is to translate MPEG 7 descriptors in OWL so that they can easily be included and referred in the Query Ontology. In addition to translation of MPEG 7 visual descriptors in OWL this task will investigate methodologies to establish links and relationships between concepts defined in the query ontology and the visual descriptors so that query should be performed both on high semantic concepts and low level visual features.

2. Work done

On task 6.1, software consolidation, in this first two years of the project we focused on creating a prototype of the VidiVideo final software: First, the MPEG7 format to serve as standard for communication between the different modules was defined, second, a functional first prototype was implemented and tested, and finally we have started working on a second version. In the following sections each of these tasks are described in detail.

On task 6.2, thesaurus of detectors, Inesc-Id have been developing an ontology of audio related concepts, currently including 105 concepts: 75 basic concepts (e.g. birds, horses, buses), 6 aggregated concepts (e.g. animals), 4 music-related concepts (e.g. vocal music), 6 concepts derived from

gender/background classification (e.g. child voice), 10 concepts derived from speaker clustering (e.g. dialogue), 2 concepts derived from language identification (e.g. English), 1 concept related to telephone bandwidth audio detection, and 1 concept derived from the transcription (e.g. count down). It is not clear, however, that we will have training examples for all these concepts. The audio concepts related to the identification of a particular speaker were not considered in this total, although their feasibility depends mostly on the availability of training material. This task is described in detail in section 7.

On the other hand, in order to achieve the goal of 1.000 concept detectors for video, at UvA the focus was set on leveraging existing expert-labeled data sets, like MediaMill Challenge and LSCOM, for new video domains. The partners at UvA have been working on this baseline of 525 concept detectors, extending it up to 576 concepts based on the annotations provided by ICT-CAS for TRECVID 2007 and 2008 and the concepts detected on the PASCAL VOC Challenge 2008 (see deliverable 6.4). In addition the UvA has been working on leveraging non-expert users labelled data, in particular tagged pictures on Flickr.

Finally, on task 6.3, Benchmark evaluation, the consortium decided to incorporate a well-known surveillance database for further benchmark evaluation of events detection, in particular the TRECVID 2008 Evaluation for Event Detection¹. This dataset for benchmark evaluation consists of airport surveillance video (called PILOT) where predefined events appear in 100 hours of airport surveillance video.

To allow for a quantitative evaluation of progress, UvA and Surrey participated jointly in the TRECVID and VOC-PASCAL benchmarks. Participation was focused on (i) the concept detection task to see how the performance of each individual detector is compared to other state-of-the-art systems and (ii) the interactive retrieval task where the aim is to have users find the best result for a

¹ <http://www.nist.gov/speech/tests/trecvid/2008/>

set of 24 information needs where for each need the interaction time is limited to 15 minutes.

As a result, the best results in 14 out of 20 categories for the PASCAL VOC challenge, and in 9 out of 20 categories in the TRECVID competition were obtained. Technical details on the results of this benchmark evaluation can be found in Deliverable D6.4.

3. MPEG7

The first achievement that we had to accomplish is to learn MPEG-7, because we were not familiar with this language. This phase was difficult because the information on MPEG-7 is still short and quite incomplete.

3.1. *What is Mpeg7*

Mpeg7 is a multimedia content description standard. MPEG-7 is formally called Multimedia Content Description Interface. Thus, it is not a standard which deals with the actual encoding of moving pictures and audio, like MPEG-1, MPEG-2 and MPEG-4. It uses XML to store metadata.

It was designed to standardize:

- a set of Description Schemes (short DS in the standard) and Descriptors (short D in the standard)
- a language to specify these schemes, called the Description Definition Language (short DDL in the standard)
- a scheme for coding the description

3.2. *Mpeg7 java library*

We developed a library to manage MPEG7 files. It is written in Java and enables to load MPEG7 files to java classes and vice-versa.

In short, we developed java classes that store information of MPEG7. Furthermore, we developed two conversion methods; one to load an xml MPEG7 file to instances of these java classes, and another one to store instances of these java classes to a file.

This library does only cover a very small part of MPEG7 –we focused only on the parts of the standard that we are using- however, it was designed so it can be extended in a very easy way.

3.2.1. Implemented part

As described above, not all MPEG7 specification was implemented. Figure 1 illustrates which elements are currently done. In words, the implemented parts are:

1. The MediaLocator of the video, which indicates where is a video file.
2. Video TemporalDecomposition to indicate the shots of the video.
3. Inside the shots, we have implemented two types of text annotation
 - a. KeyWord annotation which stores annotations written in keywords
 - b. Sentence Annotation, which stores annotation written in a tree structured like a sentence.

Figure 2 is an example of an MPEG7 which can be used by the library. The example does not contain KeyWords.

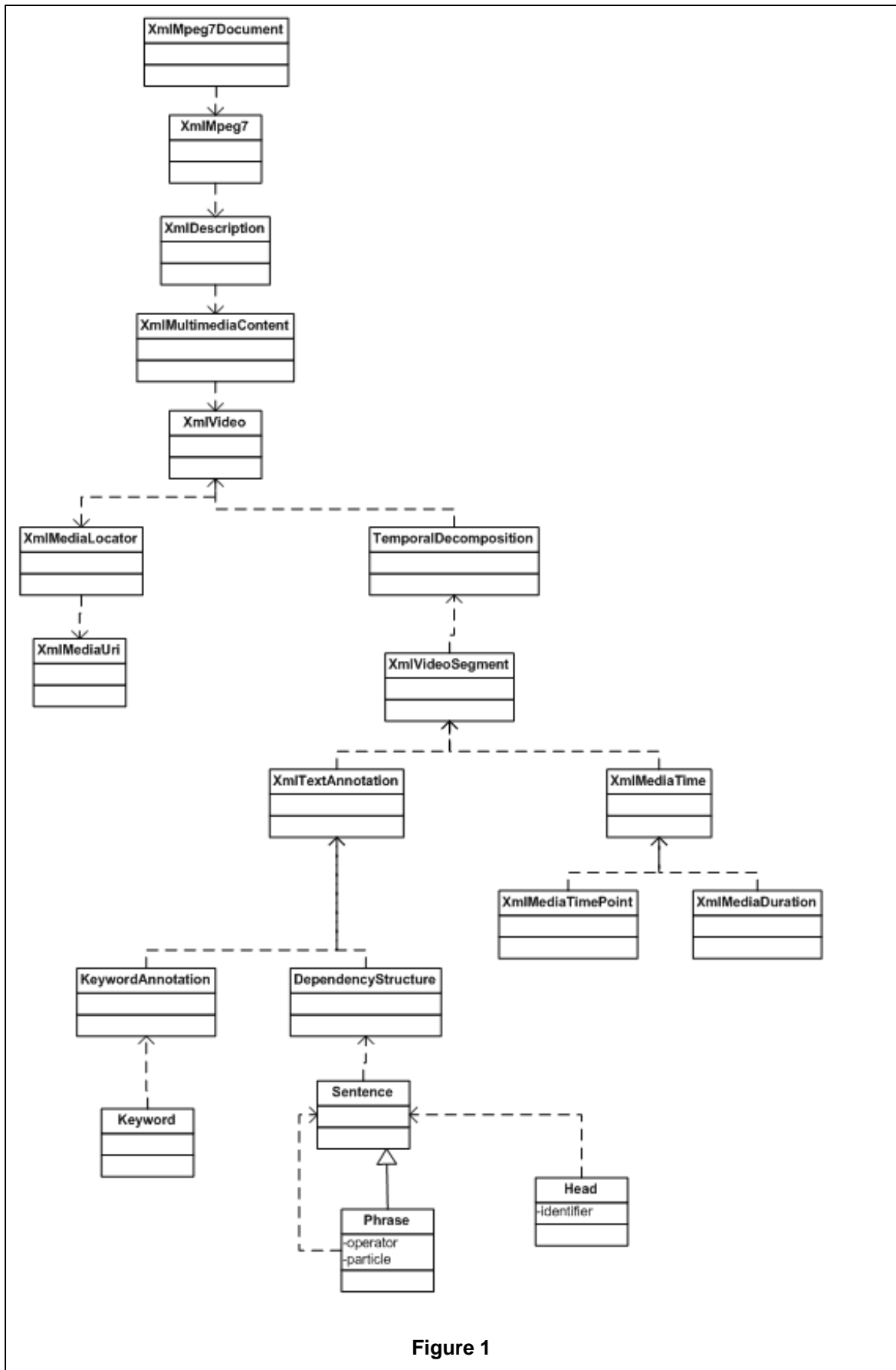


Figure 1

```

<?xml version="1.0" encoding="UTF-8"?>
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Description xsi:type="ContentEntityType">
<MultimediaContent xsi:type="VideoType">
<Video id="TRECVID2006_1">
<MediaLocator>
<MediaUri>20051102_142800_LBC_NAHAR_ARB.mpg</MediaUri>
</MediaLocator>
<TemporalDecomposition gap="false" overlap="true">
<VideoSegment id="shot1_1_RKF">
<TextAnnotation relevance="0.44" confidence="1">
<DependencyStructure xml:lang="en">
<Sentence>

<!-- agent1 throws a red box to agent2: throws(agent1,
agent2, box, red) -->
<Phrase operator="subject">
<Head id="agent1"/>
</Phrase>
<Phrase operator="object">
<Phrase>
<Head id="red"/>
</Phrase>
<Head id="box"/>
</Phrase>
<Phrase functionWord="to">
<Head id="agent2"/>
</Phrase>
<Head id="throw"/>
</Sentence>
</DependencyStructure>
</TextAnnotation>
<MediaTime>
<MediaTimePoint>T00:00:00:0F30000</MediaTimePoint>
<MediaDuration>
PT00H00M00S1001N30000F</MediaDuration>
</MediaTime>
</VideoSegment>
</TemporalDecomposition>
</Video>
</MultimediaContent>
</Description>
</Mpeg7>

```

Figure 2

3.3. Test: From logic predicates to Mpeg7

Once the Mpeg7 library was done we needed to test it, so we decided to use it on CVC's surveillance sequences, which were annotated in the form of

temporal logic predicates², which are used for semantic understanding of a video sequence and to generate natural language descriptions of it. We successfully developed a Java application that converted these annotations into MPEG-7 format using our library. Figure 3 shows an example of a description file for a crosswalk scene, and

Figure 4 indicates the corresponding Mpeg7 file using Keywords.

```
470 ! pedestrian (Agent1)
470 ! appear (Agent1, upper_left)
492 ! walk (Agent1, upper_sidewalk)
583 ! turn (Agent1, right, upper_crosswalk)
591 ! stop (Agent1, upper_crosswalk)
615 ! object (Object1)
615 ! leave_object (Agent1, Object1)
630 ! pedestrian (Agent2)
630 ! appear (Agent2, upper_right)
642 ! walk (Agent2, upper_sidewalk)
656 ! walk (Agent1, upper_sidewalk)
687 ! abandoned_object (Object1, upper_crosswalk)
692 ! meet (Agent1, Agent2, upper_crosswalk)
800 ! vehicle (Agent3)
822 ! danger_of_runover (Agent3, Agent1)
825 ! stop (Agent1)
828 ! brake_up (Agent3)
828 ! danger_of_runover (Agent3, Agent2)
838 ! back_up (Agent2)
842 ! stop (Agent2)
852 ! accelerate (Agent3)
862 ! vehicle (Agent4)
862 ! appear (Agent4, left)
872 ! exit (Agent3, right)
873 ! chase (Agent1, Agent2)
```

Figure 3

```
<?xml version="1.0" encoding="UTF-8"?>
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="VideoType">
      <Video>
```

² F-Limette – An inference engine for Fuzzy Metric Temporal Horn Logic
http://cogvisys.iaks.uni-karlsruhe.de/Vid-text/f_limette/index.html

```

    <MediaLocator>
    <MediaUri>E:/cvc/videoDeCVC/h264_vlc.mov</MediaUri>
    </MediaLocator>
      <TemporalDecomposition>
        <VideoSegment>
          <TextAnnotation>
            <KeywordAnnotation>
              <Keyword>pedestrian</Keyword>
            </KeywordAnnotation>
          </TextAnnotation>
          <MediaTime>
            <MediaTimePoint>T00:00:33:8F14</MediaTimePoint>
          </MediaTime>
        </VideoSegment>
        <VideoSegment>
          <TextAnnotation>
            <KeywordAnnotation>
              <Keyword>appear</Keyword>
            </KeywordAnnotation>
          </TextAnnotation>
          <MediaTime>
            <MediaTimePoint>T00:00:33:8F14</MediaTimePoint>
          </MediaTime>
        </VideoSegment>
        <VideoSegment>
          <TextAnnotation>
            <KeywordAnnotation>
              <Keyword>walk</Keyword>
            </KeywordAnnotation>
          </TextAnnotation>
          <MediaTime>
            <MediaTimePoint>T00:00:35:2F14</MediaTimePoint>
          </MediaTime>
        </VideoSegment>
        <VideoSegment>
          <TextAnnotation>
            <KeywordAnnotation>
              <Keyword>turn</Keyword>
            </KeywordAnnotation>
          </TextAnnotation>
          <MediaTime>
            <MediaTimePoint>T00:00:41:9F14</MediaTimePoint>
          </MediaTime>
        </VideoSegment>
        <VideoSegment>
          <TextAnnotation>
            <KeywordAnnotation>
              <Keyword>stop</Keyword>
            </KeywordAnnotation>
          </TextAnnotation>
          <MediaTime>
            <MediaTimePoint>T00:00:42:3F14</MediaTimePoint>
          </MediaTime>
        </VideoSegment>
        <VideoSegment>
          <TextAnnotation>
            <KeywordAnnotation>
              <Keyword>object</Keyword>
            </KeywordAnnotation>
          </TextAnnotation>
          <MediaTime>
            <MediaTimePoint>T00:00:42:3F14</MediaTimePoint>
          </MediaTime>
        </VideoSegment>
      </TemporalDecomposition>
    </MediaLocator>
  </MediaLocator>

```

```

        </KeywordAnnotation>
    </TextAnnotation>
    <MediaTime>
<MediaTimePoint>T00:00:43:13F14</MediaTimePoint>
    </MediaTime>
</VideoSegment>
<VideoSegment>
    <TextAnnotation>
        <KeywordAnnotation>
            <Keyword>leave_object</Keyword>
        </KeywordAnnotation>
    </TextAnnotation>
    <MediaTime>
<MediaTimePoint>T00:00:43:13F14</MediaTimePoint>
    </MediaTime>
</VideoSegment>
<VideoSegment>
    <TextAnnotation>
        <KeywordAnnotation>
            <Keyword>pedestrian</Keyword>
        </KeywordAnnotation>
    </TextAnnotation>
    <MediaTime>
<MediaTimePoint>T00:00:45:0F14</MediaTimePoint>
    </MediaTime>
</VideoSegment>
<VideoSegment>
    <TextAnnotation>
        <KeywordAnnotation>
            <Keyword>appear</Keyword>
        </KeywordAnnotation>
    </TextAnnotation>
    <MediaTime>
<MediaTimePoint>T00:00:45:0F14</MediaTimePoint>
    </MediaTime>
</VideoSegment>
<VideoSegment>
    <TextAnnotation>
        <KeywordAnnotation>
            <Keyword>walk</Keyword>
        </KeywordAnnotation>
    </TextAnnotation>
    <MediaTime>
<MediaTimePoint>T00:00:45:12F14</MediaTimePoint>
    </MediaTime>
</VideoSegment>
<VideoSegment>
    <TextAnnotation>
        <KeywordAnnotation>
            <Keyword>walk</Keyword>
        </KeywordAnnotation>
    </TextAnnotation>
    <MediaTime>
<MediaTimePoint>T00:00:46:12F14</MediaTimePoint>
    </MediaTime>
</VideoSegment>
<VideoSegment>

```

```

        <TextAnnotation>
            <KeywordAnnotation>
                <Keyword>chase</Keyword>
            </KeywordAnnotation>
        </TextAnnotation>
        <MediaTime>
<MediaTimePoint>T00:01:02:5F14</MediaTimePoint>
        </MediaTime>
    </VideoSegment>
</TemporalDecomposition>
</Video>
</MultimediaContent>
</Description>
</Mpeg7>

```

Figure 4

3.4. Description of the implemented part of Mpeg7

In the following we describe, briefly, the MPEG7 implemented part.

3.4.1. The Header

All files should start with:

```

<?xml version="1.0" encoding="UTF-8"?>
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Description xsi:type="ContentEntityType">
        <MultimediaContent xsi:type="VideoType">
            <Video>

```

Inside the video tag, all the content is described.

3.4.2. The video path

In order to store where the video, that the MPEG7 is describing, can be found, we should insert the following tag as the first child of the video tag –where sample.avi must be changed to the corresponding path-

```

<MediaLocator> <MediaUri>sample.avi</MediaUri>
</MediaLocator>

```

3.4.3. Video Segment List

After the MediaLocator, <TemporalDecomposition> should be the next child of video. Inside this tag, the shots are described.

```
<TemporalDecomposition>
```

3.4.4. Video Segment

Inside the TemporalDecomposition a Video Segments are listed. The video segments contain the time when the segment starts, and the description of the content.

```
<VideoSegment>
```

3.4.5. Video Segment description

The first child of the VideoSegment is the description of the data of the segment. In the following we expose an example of the text annotation. In case that there is more than one keyword, they should be listed after the first one.

```
<TextAnnotation>  
  <KeywordAnnotation>  
    <Keyword>plane</Keyword>  
  </KeywordAnnotation>  
</TextAnnotation>
```

3.4.6. Segment Time

The second child of VideoSegment indicates when the video starts, and which is the duration of the segment.

```
<MediaTime>  
<MediaTimePoint>T00:00:46:12F14</MediaTimePoint>  
<MediaDuration>PT00H00M00S1001N30000F</MediaDuration>  
</MediaTime>
```

In the following we describe how MediaTimePoint is written

3.4.7. MediaTimePoint

MediaTimePoint describes a time stamp of the media using Gregorian date and day time without specifying the TZD.

The format is YYYY-MM-DDThh:mm:ss:nnnFNNN. Where following lexicals are used for digits of the corresponding date/time elements:

- Y: Year, can be a variable number of digits,

- M: Month,
- D:Day,
- h: hour,
- m: minute,
- s: second,
- n: number of fractions, nnn can be any number between 0 and NNN-1 (NNN and with it nnn can have an arbitrary number of digits).
- N: number of fractions of one second which are counted by nnn. NNN can have a arbitrary number of digits and is not limited to three.

Also delimiters for the time specification (T) and the number of fractions of one second are used (F).

Beside the specification of Gregorian Date and day time, counting the number (nnn) of fractions (FNNN) of a second can allow a higher precision than one second. If counting fractions is used, the number of fractions making up one second (FNNN) shall be specified along with the counted number of fraction (nnn). Thus the FNNN defines the value range of the counted number of fractions (nnn) value. So the value range of 'nnn' is limited from 0 to FNNN-1.

3.4.8. MediaDuration

Describes the duration of a time interval according to days and day time of a notion of time encoded in the media without specifying a difference in the TZD. The time interval is defined as a half open time interval with the closed end being at the beginning. A simpleType representing a duration in time using a lexical representation of days (nD), time duration and a fraction specification (TnHnMnSnN) including the specification of the number of fractions of one second (nF): PnDTnHnMnSnNnF.

4. Integrated software prototype

When the first phase was finished CVC started a second integration phase, the aim of which was to create a single piece of software which integrated all the different elements. We started by recollecting the different software used in the previous phase and installed it in our local machines, then proceeded to design two specific execution schemes, one for the classifier module to learn the concepts (learning phase – Figure 5) and another one for the classification itself, the results of which are what will be read by the GUI and delivered to the final user (execution phase – Figure 6). How does the GUI access and store these results depends on the interface itself and therefore belongs to WP7 (see deliverable 7.1 for further reference, such as a more detailed diagram for the specific case of the CVC GUI).

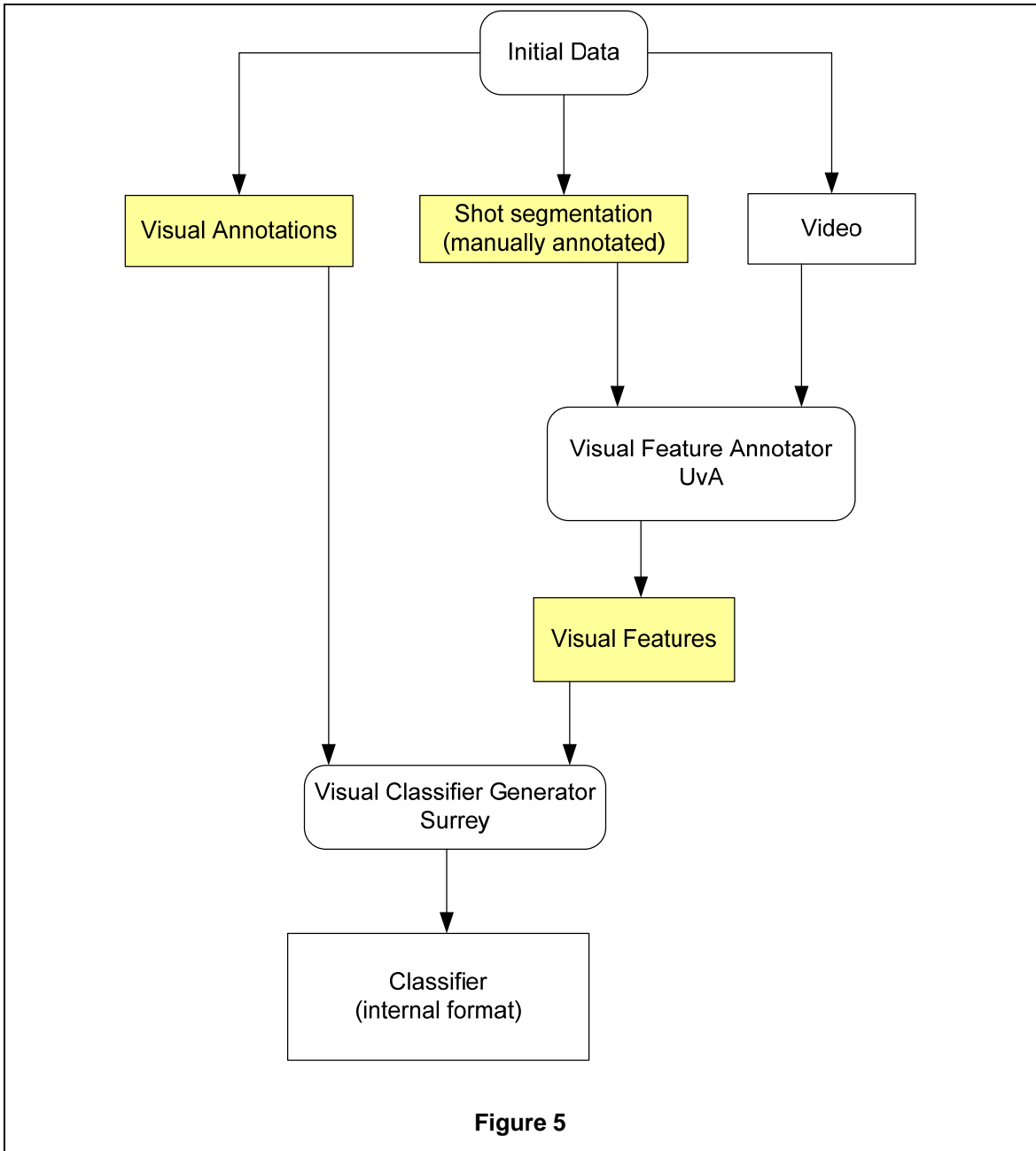


Figure 5

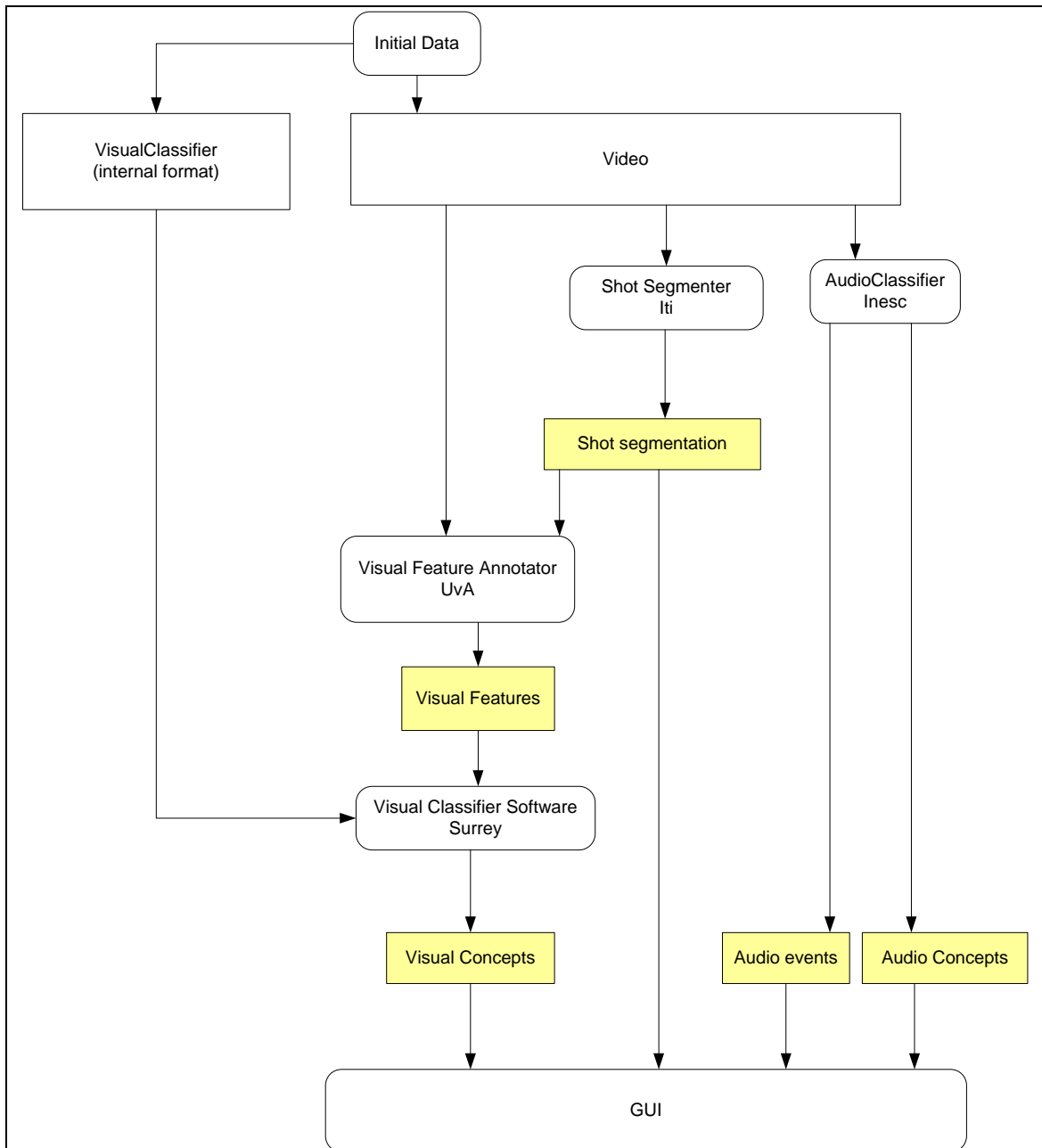


Figure 6

4.1. Software

4.1.1. Shot Segmentation

This software receives, as input videos, and outputs the shot segmentation segmentation for each one.

4.1.2. Visual Feature Annotator

This software, using the shot segmentation and the video, obtains visual features from the shots.

4.1.3. Visual Classifier Software

This software, given annotations (ground truth) from one video, generates classifiers for new videos. To do so, it uses the visual features (for the annotated videos) created by the visual analysis.

4.1.4. Audio Classifier

This software is the equivalent part of Visual Analysis and Visual Classifier, for audio features.

4.1.5. Demo GUI

Finally a Graphic User Interface is developed to be able to search into the videos with the visual and audio concepts that have been generated by the previous software.

4.2. Data

4.2.1. Video Data

These are the videos to be used

4.2.2. Shot segmentation

MPEG7 file with the shots of a video

4.2.3. Visual features

MPEG7 file with the low level features of the video shots

4.2.4. Visual/Audio Concepts

MPEG7 file with visual or audio concepts that appear in each shot of the video

4.2.5. Annotations

MPEG7 files with the same format as Visual/Audio Concepts, but with manually annotated information (ground truth).

4.3. Implementation

Following the designed schemes, we developed several software/scripts in order to install/execute all the software with a single command. However we found some problems (that we could solve with the appreciated help of all the partners) and a major inconvenient: Certh's shot segmentation software had to be run in Windows whereas other software had to be run in Linux. Due to this, at first two pieces of software had to be created, one for each system, and the data (videos and generated metadata) transfer had to be done manually.

After agreeing on a common linux platform in the VidiVideo Lisbon October meeting, Certh partners migrated their software, and it was integrated successfully.

4.3.1. System requirements

Currently the whole system can work automatically on an Ubuntu 8.04 32-bit machine that has the following packages installed (or internet connection in order to install them):

- **Shot segmentation** requires:
 - opencv – Computer vision libraries
 - libsvm – for Super Vector Machine optimizations
- **Audio Classifier** requires:
 - rpm – Red Hat package manager, to allow installing RH software
 - gfortran – The GNU fortran 95 compiler.
 - octave – language for numerical computations
 - ffmpeg – for decoding the videos.
 - gawk – GNU's implementation of 'awk' language
 - sox – for audio file format conversion
 - libxerces27 – for XML parsing
 - libapr1 – Apache's Portable Runtime Library
 - libaprutil1 – Apache's Portable Runtime Utility Library
 - libldap-2.3-0 – runtime libraries for OpenLDAP servers and clients
 - gotoblas – a High-Performance BLAS set
 - audimus – Speech recognition API by Inesc-ID, provided by them.

The integrated package does include for the moment an installer shellsript to get everything needed, and in the future we expect to include it all on a self-bootable disc and/or a virtual machine system, such as VMWare³.

5. System Tests

After we designed the first prototype, we decided to create a huge test, with about 25 videos for the learning and another 25 from the TRECvid 2007 set belonging to the news and documentaries domains, plus 2 surveillance videos (the indoor and outdoor HERMES sequences from CVC) to classify, in order to test how the software worked for the three areas. Doing this test we encountered many little problems that made the realization of the test take about two months. This was mostly because of long execution times.

During the test executions other errors were found, the gravest one being that the visual classifier software did not seem to learn any concepts: all the shots had confidence equal zero for all the concepts. With the collaboration from Surrey, the cause for this was found and solved, together with some hardcoded paths. Finally the 27-video test was finished, although with mixed results: some concepts were detected quite correctly, such as “Vegetation”, “Building”, “Crowd”... while some others were detected poorly, such as “Animal”, “Explosion/Fire”, “Mountain”... (Figures 7 and 8).

³ <http://www.vmware.com/es/overview/>

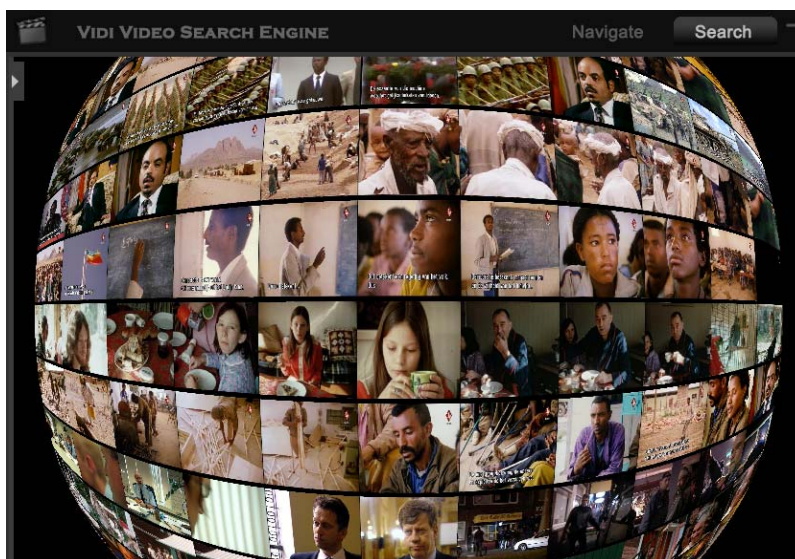
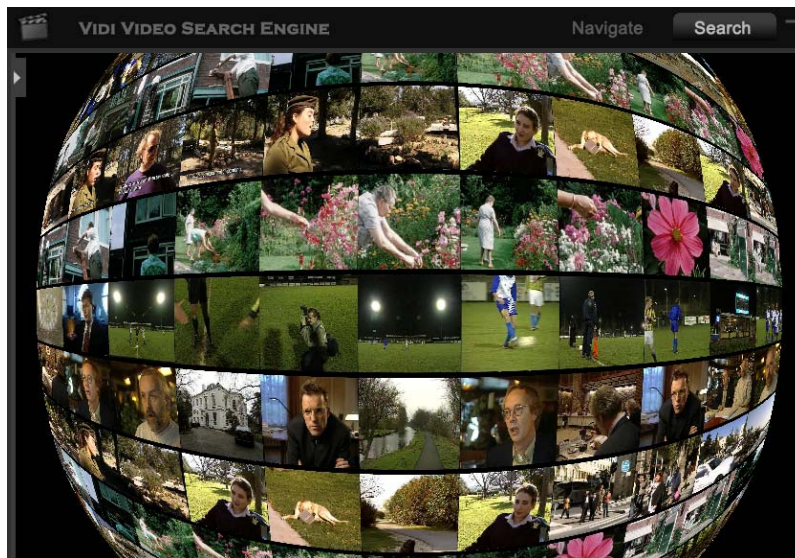
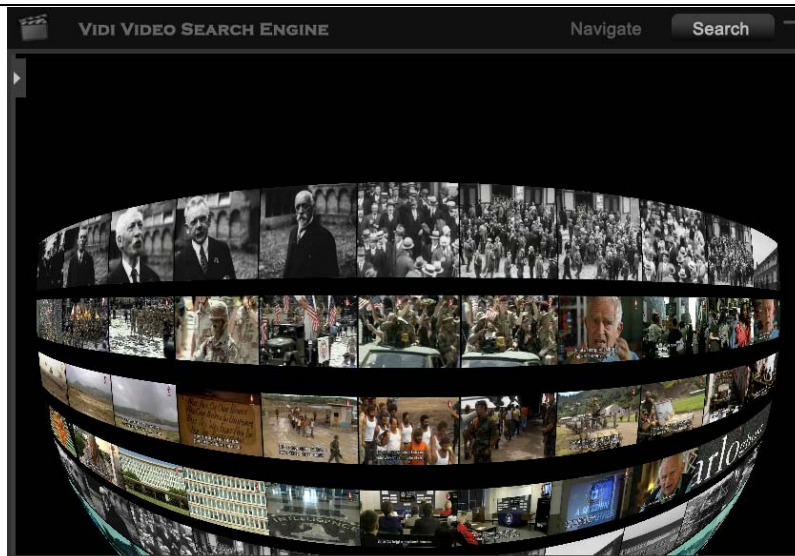


Figure 7 – Search results for “Crowd”, “Vegetation” and “Face” concepts

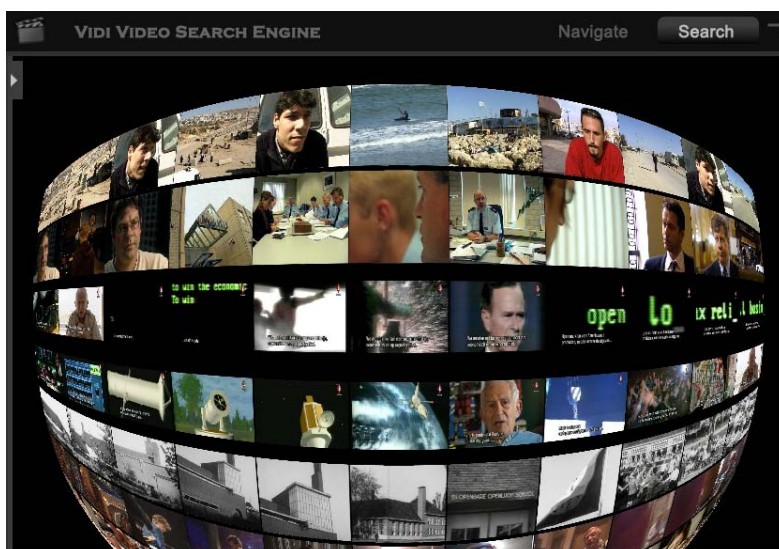
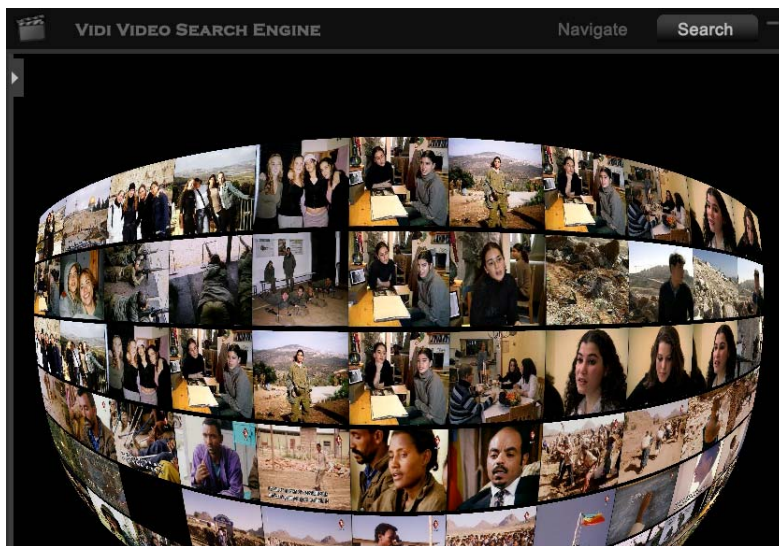


Figure 8 – Search results for “Road”, “Meeting” and “Mountain” concepts

New tests will be done with different parameters for the different modules in order to improve results. Increasing the parameters (such as number of iterations for the classifier), though, will result in increased execution times, which were the main problem faced in these tests. Table 1 shows a mean of the different execution times for each module in each test:

<u>Learning (25 videos, 50 iterations per classifier)</u>		
Prev. Feature extraction	00:06:26	
Feature extraction	05:23:21	~15-20 min. per video
post Feature extraction	00:00:09	
Prev Create Classifiers	00:00:19	
Create Classifiers	04:06:25	~2 min. per concept
TOTAL	09:36:40	
<u>Classifying (27 videos, 50 iterations per classifier)</u>		
Shot segmentation	25d 02:24:00	~1 day per video (news & documentaries)
post Shot segmentation	00:00:01	
Prev. Feature extraction	00:05:38	
Feature extraction	05:59:06	~15-20 min. per video
post Feature extraction	00:00:14	
Prev Classify	00:00:01	
Classify	04:44:15	~10 min. per video
post Classify	00:00:08	
Prev Audio Events	00:05:17	
Audio Events	01:10:19	~2 min. per video
post Audio Events	00:00:00	
metadata conversión	00:10:32	
TOTAL	25d 14:40:25	

Table 1

We can clearly observe that time cost-wise currently the shot segmentation module is way too costly compared to the other ones, at least for the long sequences (news and documentary domains) that formed the bulk of these tests.

Finally, on the surveillance domain a couple of points must be noted: First, the segmentation software took much less time for these videos (about 3 hours each), because of the much shorter length of these sequences. Second, the results weren't good due to the lack of actual shot segmentation: The images don't vary substantially in a continuous surveillance sequence, and thus the current segmentation algorithms, based on sudden image content changes,

don't work. Since visual features are extracted per shot, being there only one shot per video not enough features are extracted, and the classifiers can't work correctly. Therefore, a semantical shot segmenter, based on the sequence's different agents' actions, will be needed for this domain.

6. Second Prototype

Now that the first integrated system is complete, we are proceeding towards a second one, more efficient and with new modules incorporated such as a human detector from CVC and an OCR system for detecting concepts on written text, as agreed by the partners on the Lisbon meeting. Also, individual improvements on the different modules will be applied: for instance, the visual concepts classifier may be changed from the current visual features based one to a kernel based system on which UvA and Surrey are currently working on in order to improve the final results.

6.1. *New Execution Scheme*

Now that the system is functional, we will apply a series of improvements into the current execution scheme, such as multi-threaded processing for independent modules (all the tasks are currently executed one after the other) or common video data and metadata directories to avoid excess of data movement and duplication.

6.2. *Human Detection*

This human detection software by CVC member Marco Pedersoli is based on Histograms of Oriented Gradients (HOG) that can provide a state of the art detection rate with a reduced computational cost.

The method consists of a sliding window detector that scans the image in every possible position, location and scale looking for a learned pattern.

The computation of the HOG features is speeded-up using a recursive computation, where from the image gradient pixels, blocks of histograms of gradient orientation are built from the minimum to the maximum size needed for the multi-scale scan.

The detector is a Support Vector Machine (SVM) trained on 2400 positive examples which are images of humans cropped and resized to a standard size of 128x64 pixels and 10000 negative examples that are randomly taken from images not containing humans.

To improve the detection rate we retrain the detector adding the windows that have been incorrectly classified. We repeat the iteration until no further improvements are obtained.

We have applied this system to a few videos from the documentary and surveillance sets to test its performance in order to be added into the system. The first results were not very good, probably because of the image size in the case of documentaries and news, but the detector is still being worked on. Figure 9 shows some of the results for the news and documentaries domains, whereas Figure 10 shows the ones from surveillance sequences.

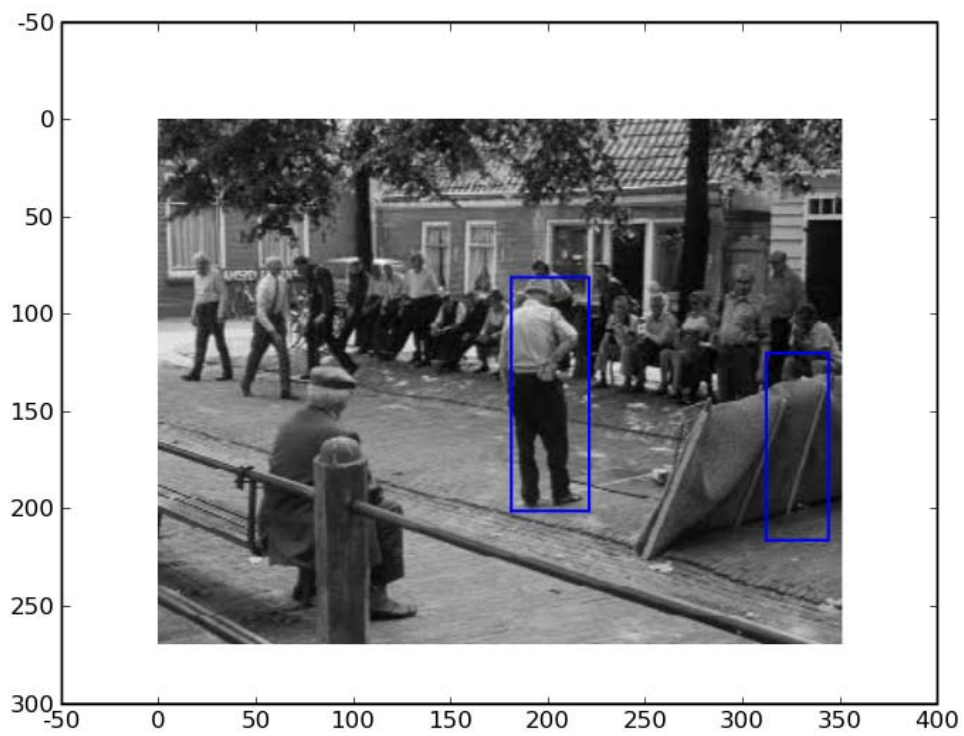
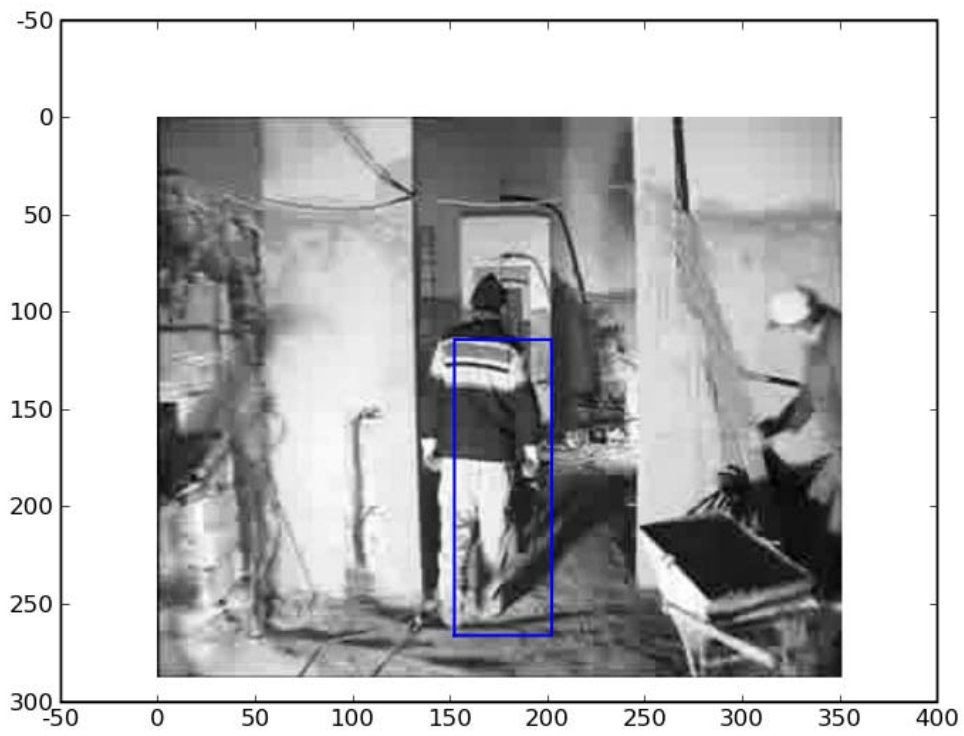


Figure 9

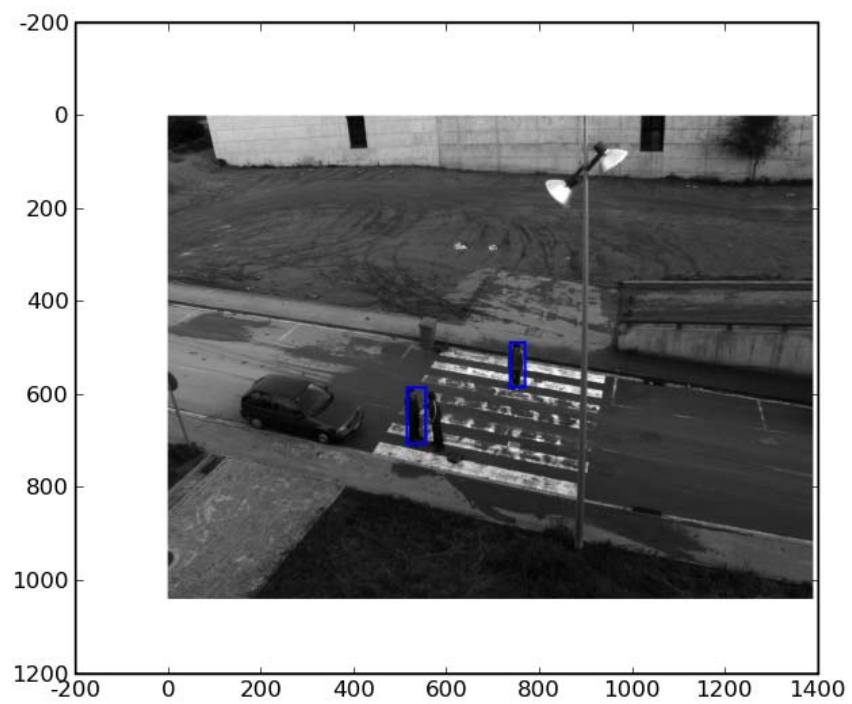
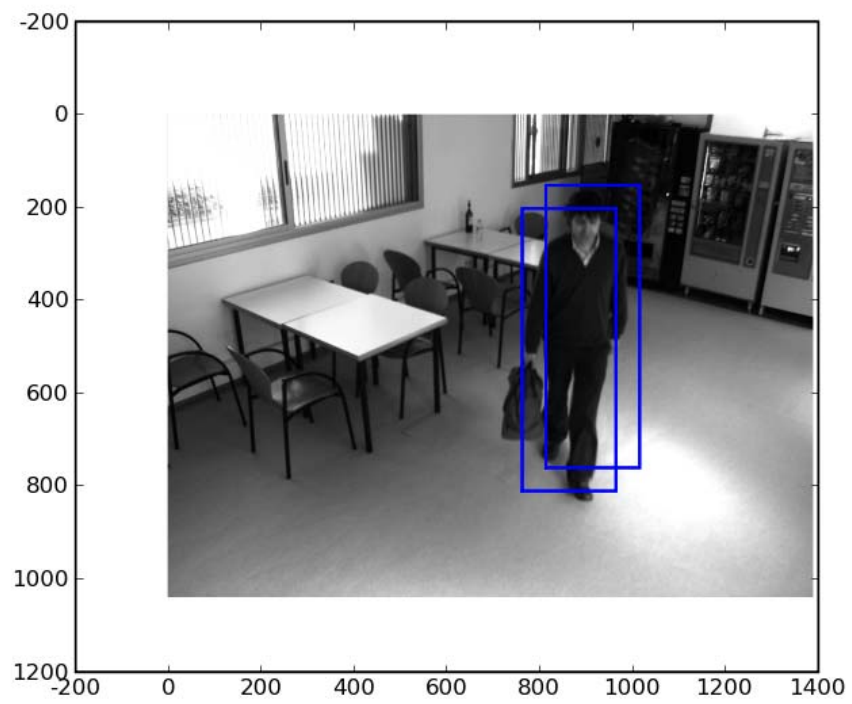


Figure 10

6.3. OCR

Tesseract⁴ is a free optical character recognition engine. It was originally developed at Hewlett-Packard from 1985 until 1995. After ten years with no development, Hewlett Packard and UNLV released it in 2005. It is currently developed by Google and released under the Apache License, Version 2.0. Tesseract can process English, French, Italian, German, Spanish, Brazilian Portuguese and Dutch and it can be trained to work in other languages as well. It is considered one of the most accurate free software OCR engines currently available, and that is why we chose it.

We tested tesseract on the documentary and news video set, with rather poor results (Figure 11). The problem was that tesseract is a raw OCR engine: It has no document layout analysis, which means it just reads plain text on screen, for example a scanned sheet of paper. It doesn't have any kind of image processing, so it cannot be used for complicated OCR tasks such as identifying characters from a low resolution video.

In order to be able to read complex character layouts, then, further investigation is needed and will be conducted on this matter through the use of other systems. Currently OCRopus⁵ is being considered for this task.

OCRopus is a free document analysis and OCR system released under the Apache License, Version 2.0 with a very modular design through the use of plugins. These plugins allow it to swap out components easily. It combines pluggable layout analysis, pluggable character recognition, and pluggable language modelling. It's based on tesseract and currently uses it as its backend for final character recognition after image processing.

⁴ <http://code.google.com/p/tesseract-ocr/>

⁵ <http://code.google.com/p/ocropus/>

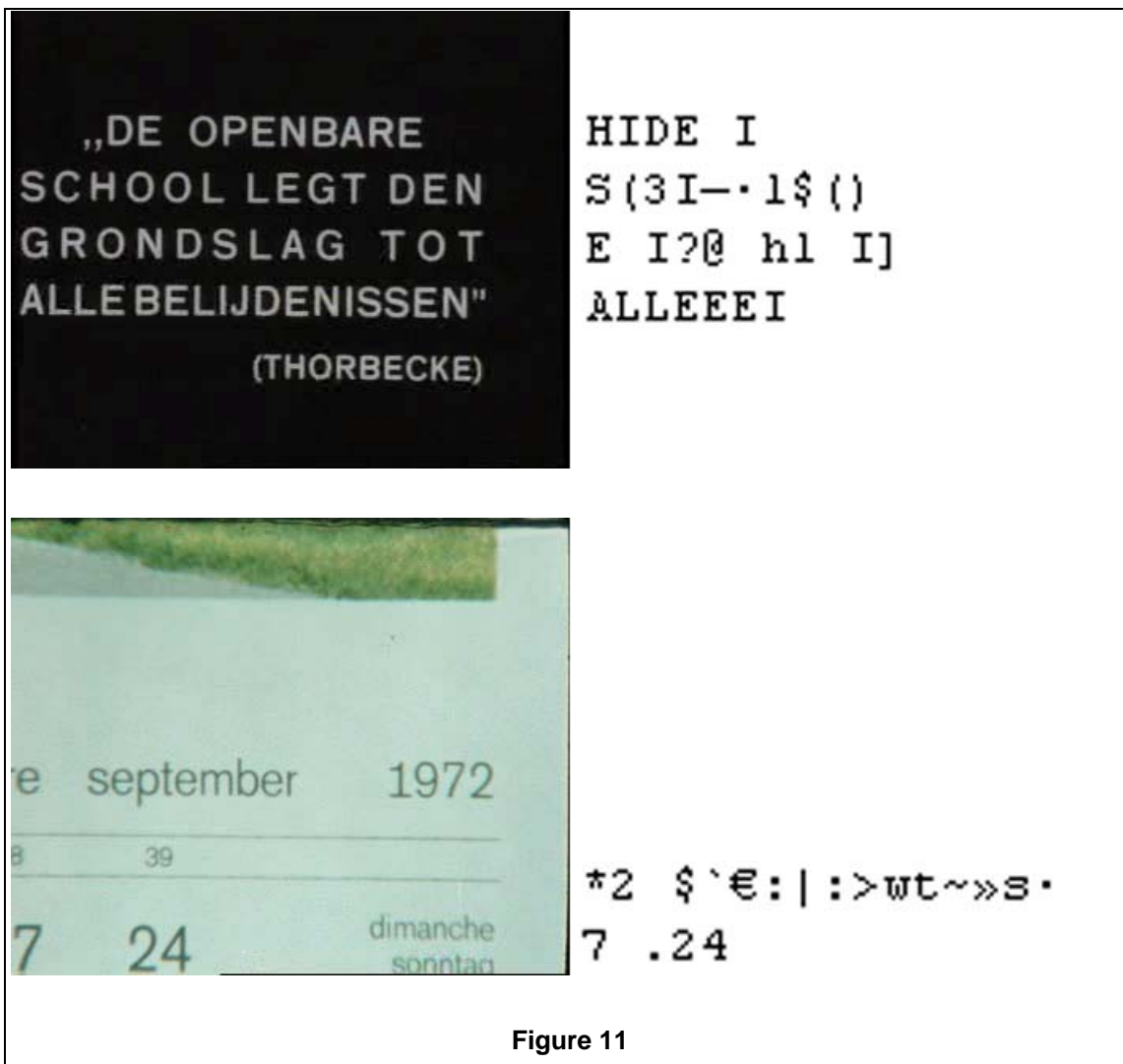


Figure 11

6.3.1. OCR pre-processing module

On the other hand, the partners from Unifi have been working on a pre-processing module which performs detection of areas of video frames that may contain text and eases the background/foreground separation. Its goal is to perform an initial image processing step that produces images that can then be fed to an OCR module, such as tesseract described above.

This module's works by detecting the previously smoothed image's corners: a corner can be defined as the intersection of two edges, and has been used to model the appearance of text. The detected corners are then clustered, considering the fact that superimposed text is of limited size w.r.t. the frame size, and clusters are analyzed to discard those that are unlikely containing text.

Given the fact that captions may appear with different sizes, positions, lengths, numbers, etc. the mean shift clustering technique has been used, considering the feature space as composed by the corner coordinates.

The clusters obtained by this process are then analyzed to discard those that are unlike to contain text because they contain a too low number of points. Finally, the last processing step is the binarization of the selected image patches to obtain the text itself.

The algorithm is able to cope also with several types of scene text, as shown in Figure 12.

Preliminary experiments, performed on a set of 700 frames sampled from news videos of several European broadcasters show that the system detects 90.76% of the captions, missing the remaining 9.24% with an average of 0.65 falsely detected clusters per frame, that anyway can be discarded as noise by the OCR.

The integration of this module with the OCR part of the VidiVideo project has just initiated. Once completed, analysis of the combined modules will be performed.



Figure 12

7. Thesaurus of detectors

In our strife towards a thesaurus of detectors, we have been focusing on gathering labeled image examples. As manual labeling is unfeasible, we have considered automatic image tagging. Machine tagging, though, is still very challenging: Images of the same concept vary significantly in terms of visual appearance, e.g., illumination, scale, and perspective. A large and diverse set of learning examples is imperative to model the visual diversity. On the other hand, social multimedia sharing systems have successfully motivated common users around the world to tag their visual content on the web. A good example of user tagging is Flickr. It hosts over 2 billion images, and receives around 3 million new uploaded photos per day. However, tags contributed by users are known to be ambiguous, limited in terms of completeness, and overly

personalized. This is not surprising due to the significant diversity of users' knowledge and cultural background. Given web-scale, yet unreliable, user-tagged photos, an interesting question is whether we can harness user tagging to solve the image tagging problem.

Many methods have been proposed to tackle the image tagging problem. We divide them according to their model-dependence into two types of approaches, namely model-based approaches and model-free approaches. Given a set of labeled images as training data, the model-based approaches focus on learning a mapping between low-level visual features (e.g., color and local descriptors) and high-level semantic concepts (e.g., airplane and classroom). Due to the expense of manual labeling, however, currently only a limited number of visual concepts can be modeled effectively using relatively small-scale datasets. Besides, the approaches are often computationally expensive, making them difficult to scale up. In contrast, the second type of approaches attempts to annotate an image in a model-free way by utilizing web photos. The approaches assume a large well-labeled database exists, such that one can find a visual duplicate for the unlabeled example. Then, automatic tagging is done by simply propagating tags from the duplicate to that image. However, since the database in reality is of limited-scale with noisy annotation, neighbor search is first conducted to find visual neighbors. De-noising methods are then used to select relevant tags, out of raw annotations of the neighbors, to annotate the unlabeled image.

Inspired by the initial success of the model-free methods in handling web-scale data, we propose to combine machine tagging and user tagging by employing the model-free methodology. The novelty of this work is that we introduce into the search-based framework an effective joint-modality tag relevance estimation method. By taking both textual and visual clues into account, the method accurately estimates tag relevance from 1.5 million user-tagged images.

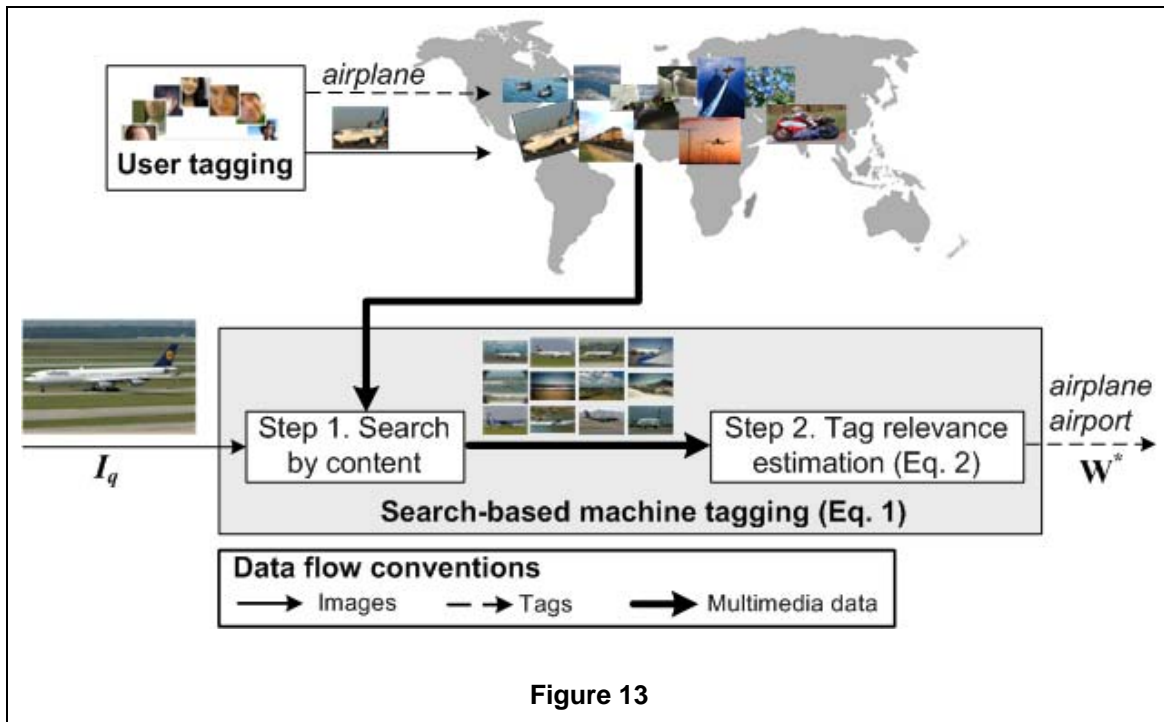
7.1. Machine-And-User Tagging

7.1.1. Search-based Machine Tagging

We aim for machine tagging methods that accurately predict relevant tags with respect to unlabeled images. We consider a tag relevant to an image if the tag accurately describes objective aspects of the visual content, or in other words, users with common knowledge relate the tag to the visual content easily and consistently. Although the relevance of a tag given the visual content can be subjective for a specific user, an objective criterion is desirable for the general content understanding problem. Given an unlabeled image I_q , if a well-labeled duplicate of I_q exists, machine tagging is solved by using tags of the duplicate. In reality, however, the assumption is often violated: either the duplicate does not exist, or it is unlabeled or mislabeled. An intuitive idea is then to approximate the duplicate by a set of visually similar examples.

Despite the semantic gap, given a very large dataset one might find relevant images in visual neighbors with a reasonable accuracy. Meanwhile, the increasing amount of web photos tagged by users around the world is creating a potentially unlimited-scale database. Though visual search results can be unsatisfactory, if relevant tags stand out from relevant images, one might still find prospective tags for tagging. Therefore, we decided to annotate images by harnessing worldwide usertagged photos within a search-based framework (Figure 13). This system works in 2 steps:

- *Step 1. Search by content.* We find k nearest neighbors of I_q from a user-tagged image database.
- *Step 2. Tag relevance estimation.* Given tags of the neighbor images, we then select most relevant tags to annotate I_q .



7.1.2. Tag Relevance Estimation from User Tagging

In previous work, methods to estimate the relevance of a tag given an image are mainly derived from the text retrieval field. The unreliability and sparsity of user tagging make the text-based methods inaccurate. Intuitively, if one can effectively exploit both textual and visual information, tag relevance estimation could be more accurate. However, visual features are often of high dimensionality and significant diversity exists in user tagging vocabulary. Hence, directly modeling co-occurrence of textual and visual modalities, e.g., using a multivariate Gaussian, tends to be problematic. As an alternative, we introduce a nonparametric joint-modality method based on a neighbor-voting algorithm. The algorithm originally aims for social image retrieval. Here, we extend it to the machine tagging scenario. The intuition behind neighbor voting is as follows: if different persons label visually similar images using the same tags, these tags are likely to reflect objective aspects of the visual content. Hence, the relevance of a tag with respect to an image can be inferred from tagging behavior of visual neighbors of that image.

Through the algorithm, unambiguous and objective tags receiving most neighbor voting stand out. We also take tag informativeness into account. Since

the text-based methods ignore visual clues, they are conducted in a global visual feature space. In contrast, our method computes tag relevance in a localized space, i.e., the neighborhood of image I_q . It is this restriction that provides a joint-modality mechanism, leading to more accurate and robust tag relevance estimation. Finally, to annotate an image, we choose all tags from its neighbors to form a candidate tag set. Each tag in the set is ranked in descending order according to its relevance and we select t top ranked tags as the final annotation. The choice of t is a tradeoff between annotation precision and recall, which needs to be determined experimentally.

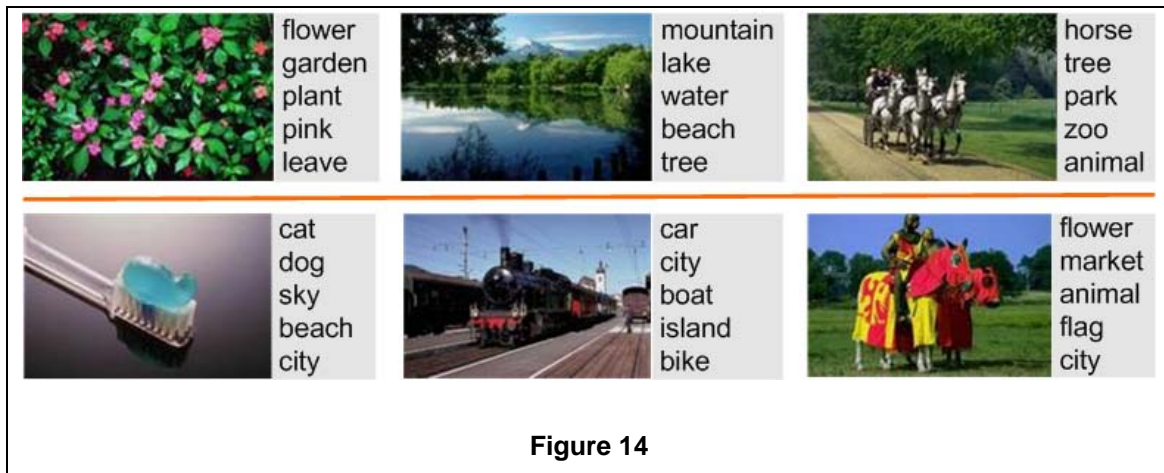
7.2. Results

Figure 14 shows images tagged automatically by our method. Good examples are at the top row and bad ones at the bottom row. Our viability is recognized by the community, and has resulted in acceptance of this work as an oral presentation at IEEE ICASSP 2009.

8. Conclusions

In conclusion, we can say that overall these 2 years have been satisfying:

- The deliverables have so far been completed on time.
- The integrated software is slowly but continuously improving, with a first fully functional prototype working on an Ubuntu system or virtual machine, although it currently requires big amounts of disk space and processing time.
- The thesaurus of detectors, both for video and audio, is expanding to the point of having reached around 680 concepts, two thirds of the required amount.



- We have done extensive experimentation with obtaining labeled examples from Flickr in an attempt to tackle one of the major bottlenecks in semantic indexing, getting sufficient examples.
- First benchmark, consisting of the TRECVID and PASCAL VOC challenges 2008, was passed successfully.

Pending work for the 3rd year is optimizing as much as possible the 2nd prototype of the system to make it work faster and taking less disk space on one hand, and adding new functionalities to it, such as OCR, on the other. Moreover, these new functionalities will allow us to detect new concepts and thus expand the thesaurus of detectors further ahead, in order to reach the 1000 mark, as much as possible relying on automatically captured labeled examples from Flickr. To reach the latter, more research is needed to get rid of all the Flickr tags which are not reflected in the visual content.