Future Internet **2012**, *4*, 179-215; doi:10.3390/fi4010179

OPEN ACCESS

future internet

ISSN 1999-5903

www.mdpi.com/journal/futureinternet

Article

A Flexible Object-of-Interest Annotation Framework for Online Video Portals

Robert Sorschag

Institute of Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstrasse 9-11, Vienna A-1040, Austria; E-Mail: sorschag@ims.tuwien.ac.at; Tel.: +43-1-58801-18825

Received: 23 November 2011; in revised form: 2 February 2012 / Accepted: 20 February 2012 / Published: 22 February 2012

Abstract: In this work, we address the use of object recognition techniques to annotate *what* is shown *where* in online video collections. These annotations are suitable to retrieve specific video scenes for object related text queries which is not possible with the manually generated metadata that is used by current portals. We are not the first to present object annotations that are generated with content-based analysis methods. However, the proposed framework possesses some outstanding features that offer good prospects for its application in real video portals. Firstly, it can be easily used as background module in any video environment. Secondly, it is not based on a fixed analysis chain but on an extensive recognition infrastructure that can be used with all kinds of visual features, matching and machine learning techniques. New recognition approaches can be integrated into this infrastructure with low development costs and a configuration of the used recognition approaches can be performed even on a running system. Thus, this framework might also benefit from future advances in computer vision. Thirdly, we present an automatic selection approach to support the use of different recognition strategies for different objects. Last but not least, visual analysis can be performed efficiently on distributed, multi-processor environments and a database schema is presented to store the resulting video annotations as well as the off-line generated low-level features in a compact form. We achieve promising results in an annotation case study and the instance search task of the TRECVID 2011 challenge.

Keywords: video annotation; video sharing; object recognition

---

1. Introduction

Video sharing has become one of the most important activities in the internet and it accounts for a large share of today's internet traffic [1]. Professional videos and user-generated content of different genres are available on video portals like YouTube, YouKo, Vimeo, Hulu, and Metacafe [2–6]. Millions of users visit such portals every day [7] in order to watch videos on their computer screens and mobile phones. Despite this success, current video sharing and retrieval works quite differently from classical internet document retrieval for text documents, websites, and images. In order to understand the big picture of current video sharing and its development potentials it is necessary to get a deeper knowledge of the roles and aims of video portals, their users, and about the nature of video documents. Thus, we start with a brief overview of these topics before a flexible object-of-interest annotation framework is proposed for the assistance of various video portals.

Video documents consist of audio-visual content and they are accompanied by metadata. The entities of video documents are scenes, shots, and frames on a temporal level; objects on spatial level; and object related events on a semantic level [8]. Bibliographic metadata is used to describe general video attributes, such as the video title, its genre, owner, duration, and number of views on a video portal. Structural metadata includes the hierarchy of scenes and shots while content metadata describes information like the shown actors and audio transcripts of a video [9]. In the process of video *annotation*, metadata of all three types is generated using manual, interactive, or automatic annotation techniques. Most video *retrieval* techniques solely operate on metadata although the video content itself can also be used by text queries and visual queries [10]. More than a decade ago, studies about the formulations of text queries have been made in [9], and [11]. They distinguished between (a) queries about the videos as a whole; (b) queries about the topic content; (c) queries about sensory content (e.g., the appearance or location of objects); and (d) queries about the data and metadata. In contrast to text queries, visual queries use images or videos as input to retrieve similar videos or videos that show specific objects. The main difficulty of these visual queries is the generation or retrieval of appropriate query content itself.

Video portals are one of the major success stories of the Web 2.0. Since YouTube went online in 2005 an immense number of videos made their way through the internet and more than 50% of all web users visit one of several hundred online portals on a daily basis [7]. In short, these portals try to provide as many videos as possible and to maximise the number of page visits and views. Complex server architectures and service models are used [1] to deal with the vast amount of data requests. User generated video portals like YouTube, and YouKo mainly rely on advertisement as their source of revenue while paid accounts are frequently used from domain specific portals such as Netflix where only professional content is shown [12]. Portals for user-generated content allow video uploads and users can annotate their own videos with free text tags and make comments on videos they have watched. Most portals can only be used to generate metadata that belongs to an entire video while a few systems such as Synvie [13] and the prototypical system of [14] enable users to make comments on video scenes and shots. The portals use own, proprietary metadata formats and differentiate between different video genres. Furthermore, there are often restrictions for the video content, codecs, and duration that is allowed for uploading.

Beside video consumption on the websites or in external video players, each portal provides some video retrieval and browsing services. Videos are presented with one or multiple keyframes and metadata, such as its title, duration, number of views, and a general description. The same keyframes, for instance the first frame of a video, are always used although [15] has shown that these keyframes are not suited to present the results of each video query. Video retrieval is usually possible through (a) direct navigation to videos using external links; (b) internal links such as personalized recommendations and related videos that are shown in the portal website; and (c) text-based searches. According to [16], internal links of option (b) account for about 60% of all views in YouTube. The presented recommendations are personalized for each user and they reflect her recent activities on the site. Related video links are shown for videos that stem from the same user of a selected video that are similarly tagged or share other similarities like internal user recommendations or comments. However, it has been shown that there is an extremely weak social connectivity between users of a video portal [17,18] and that views from user recommendations mainly account to direct links that are contained in Facebook postings, emails, or somewhere else. Such external links provide a way of advertising the videos, and thus they are especially important for recently uploaded videos with 10% to 30% of all views in the first 4 days of a YouTube video [19]. In the same study, the overall views from external links only accounted for about 7% views of all YouTube videos while a much higher impact of external links has been measured in YouKo. However, the current number of YouTube visits that consists of a single page view (bounce rate) and its constant increase over the last 20 months (up to 33% measured in November 2011 by Alexa [7]) indicate a higher number of views from external links today and supports their importance. The last class of video retrieval in online portals stems from search queries and goal-oriented browsing around a certain topic [16]. For this, it follows that only a small partition of all video views originates from a search oriented retrieval process, although no exact numbers are available. Most video portals only allow text-based video searches while visual queries and their combination with text queries [20] are mainly integrated in research prototypes [21–24] that stem from computer vision research.

Video portal *users* follow the same intentions as users of traditional broadcast television. On the one hand, they visit video portals with the simple, unarticulated want to be entertained by content that they find interesting. On the other hand, they are interested in specific information and content. According to [25] users can be classified in (a) browsers with no clear end goal that perform a series of unrelated searches within a session and jump across multiple topics; (b) surfers that have a moderate clarity of an end goal. Their actions may be somewhat exploratory in the beginning but increase the clarity in subsequent searches; and (c) searchers that are very clear about what they are searching for in the system and only need short sessions to retrieve an end result. Users further adapt to the provided technologies and thus there exists a relationship between the most used query types (about videos as a whole that can be answered with bibliographic metadata) and the availability of results [9]. In other words, this means that people tend to ask queries where they expect reasonable answers. Queries about the visual content usually involved actors and their actions while questions about objects and the video structure have only been raised by people that were involved with multimedia work.

The reasons why users want to share video content are manifold. One group of users produces content by themselves and simply like to share this content with other people. Another group wants to share or promote videos that they find interesting. The generation of tags and comments for video content is

motivated by the similar goals, as shown in an investigation of user-generated YouTube tags [26]. One interesting finding of this work is that tags are generally not used to enhance the video description but to improve the video popularity. [27] further discovered, that people have a higher motivation to annotate their content when they get the right incentives and affordances for annotation, which means that poorly annotated content is often a consequence of inappropriate annotation tools. Social tagging systems are described in [28] together with a taxonomy to compare different tagging architectures, as they have done it for Flickr and Delicious [29,30]. Another initiative to improve video annotations comes from annotation games, such as OntoTube and Yahoo's VideoTagGame, where users are motivated to tag videos for fame [31].

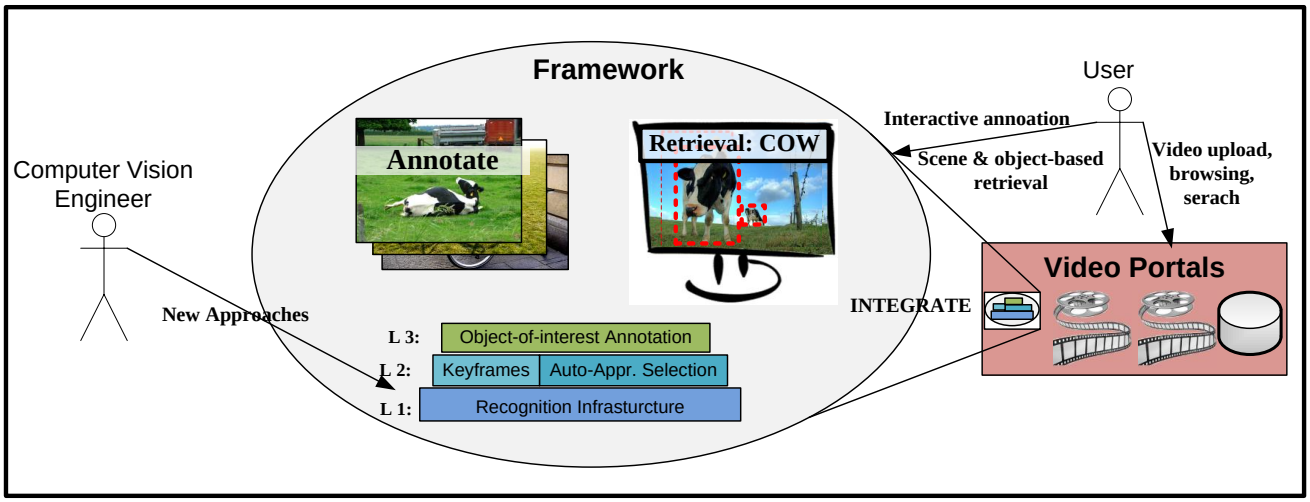
Content-based analysis techniques have been investigated in many works for the use in video annotation and retrieval. Amongst other things, researchers proposed ways to generate better and more annotations using tag recommendation, tag propagation, and tag ordering [32]. Although a large number of retrieval techniques and systems exist that use low-level content features independently or combined with text queries (see Section 2), content based analysis techniques are not presented in current video sharing portals. The proposed object-of-interest annotation framework in this work tries to learn from related systems and focuses on solutions with a high potential for industrial applicability. Thus, this framework represents an interface between video portals and computer vision research targeting object recognition.

The framework further offers the possibility to improve video sharing portals in regard to their capability of scene retrieval and object based text queries. It exploits object recognition approaches without presenting new methods or fixed analysis workflows. Recognition approaches of all kinds including global and local detector-descriptor chains, matching strategies and machine learning techniques can be integrated and the framework can be easily extended to incorporate computer vision advances. Moreover, we designed this framework for the use in various video platforms instead of proposing another stand-alone system. Despite its flexibility in matter of recognition approach extension and its compatibility with existing video portals, a few cornerstones mark the framework's efficient nature: Firstly, videos are represented by a few keyframes per scene. Secondly, object annotations and low-level features are stored in a simple and compact form with sufficient information to answer text queries on scene level. Thirdly, an appropriate recognition approach is automatically selected for each annotated object. Furthermore, the framework was designed for the use in large-scale video collections with distributed analysis on multi-core and multi-machine architectures.

Figure 1 shows a high-level perspective of the framework in combination with the roles of video portals, their users, and computer vision engineers as well as an overview of the proposed object-of-interest annotation. This annotation process first pre-processes all videos of the portal to select relevant keyframes and to extract visual features. User interaction is then required to initiate the annotation of new objects in uploaded videos starting from a single example. In this process, keyframes that are supposed to contain the same object are returned to the user for feedback. After this training data generation, the object is automatically annotated in all videos of the collection. The relevance of scene and object level retrieval results should then be visible to the user at the first glance, as shown for the query "cow" in Figure 1. The evaluations of this work are focused on the object-of-interest

annotation in real-world scenarios and on automatic approach selection to show the capabilities of the proposed framework.

Figure 1. A high-level perspective of the proposed annotation framework.



The remainder of this work is structured as follows. Section 2 reviews the related work in the area of object-based video annotation, object recognition, and recognition infrastructures. Section 3 describes the proposed object-of-interest annotation. In Section 4 the configurable object recognition infrastructure is introduced and it is discussed how video portals might integrate the framework. Section 5 presents an annotation prototype of the framework and the results of experimental evaluations. Finally, Section 6 draws conclusions and gives future research directions.

2. Related Work

Content-based analysis methods for video annotation and retrieval are proposed from several research directions. In the following, we review this research with an emphasis on scene-based and object-based methods. Furthermore, we discuss the state-of-the-art of object recognition, automatic approach selection, and recognition infrastructures. These topics are directly related to the proposed framework and a comparative discussion is given where possible.

2.1. Content-Based Annotation and Retrieval

Several content-based video retrieval systems have been proposed in the literature, such as the QBIC system [21,33], and the systems JACOB, VideoQ, Violone, Netra-V, VisualGREP, InsightVideo, ANVIL, and VARS that are reviewed in [22,34]. Most of these systems stem from the 1990s and are focused on query by example techniques that have been adapted from image retrieval [25]. They usually extract low-level features (color, shape, texture) from keyframes and only a few of these systems incorporate motion [35]. Specific video domains, for instance broadcast news and sport, have been investigated in [36,37] to annotate specifics like the sport type, playing fields, players and presenters, the audience and inserted text. Moreover, a few systems have been proposed that are specifically designed for large-scale video collections [38–40] that focus on scalability rather than most accurate analysis approaches. The

predominant inconvenience of query by example systems is the query generation itself. In contrast to text queries, it is quite difficult to obtain content examples that fit to specific retrieval goals that a user might have in mind.

Thus, recent research mainly focused on approaches for the generation of additional, richer metadata that is suitable for text-based video retrieval. In the simplest case, videos are thereby classified into a few predefined genres using text, audio, and visual cues [41]. Most approaches annotate videos with up to a few hundred concepts or labels. This task is also referred to as "video concept detection", "video semantic analysis", and "high-level feature extraction" [42]. Proposed techniques include support vector machines, Gaussian mixture models, maximum entropy methods, modified nearest-neighbour classifier, and multiple instance learning as described in the survey of Naphade *et al*. [43]. However, concepts consist of fuzzy semantics. Objects and object relations are given as concepts together with events and sometimes even with attributes like the size and speed of objects [44]. Concept annotation is usually performed globally for entire videos and without spatial information. A recent study about video indexing and retrieval [20] describes the state-of-the-art in video structure analysis, the used visual features, and analysis techniques.

Another trend in the computer vision society is the use of external knowledge and existing annotations for the generation of new metadata. Similar images or videos are obtained by searching (starting from a few keywords) to collect further tags for the investigated content, see [45,46]. Search-based approaches [47] are further used to collect a sufficiently large training set for machine learning environments, for instance in the video concept detection system TubeTagger [48]. [49] proposed an approach to employ the redundancy of online video portals in order to improve the annotation of videos. They performed near duplicate video detection is performed to detect identical and overlapping videos that have been uploaded to exchange their annotations.

Traditional video retrieval is mainly performed on a global level for entire videos. This trend is fostered by video portals that only allow global annotations and that mainly consist of short, user-generated videos. However, some annotation and retrieval techniques are performed on scene and shot level. For instance, the ShotTagger [50] allows shot-based video tagging and the authors presented a user study that indicates that the resulting retrieval is more convenient for users than the standard approach. The ViGOR system [51] presents a video retrieval system for complex search tasks, like the query to retrieve shots where a person is walking or riding a bicycle. In [52] social video annotation and retrieval systems are proposed that operate on scene level while [46] exploits the social knowledge that is embedded in Wikipedia, Flickr, and YouTube to generate such annotations without user interaction. Further scene-based and shot-based annotations originate from video annotation methods that operate on objects. A recent study of these methods is given in [10] together with a taxonomy to characterize object level retrieval systems. In this context, we have to distinguish between manual, interactive, and automatic object annotations. A manual system is given in [14] where users can make comments to spatio-temporal regions (bounding boxes) of online videos. Objects, object relations, and object related events are then automatically extracted from these user comments to enable object-based and event-based scene retrieval. Similar object annotation systems are presented in [40,53]. An interactive video segmentation was presented in [54] to extract objects-of-interest from the background in videos. The segmented objects can be further used for video annotation.

Sivic and Zisserman [55] proposed one of the first automatic attempts to enable object-based video retrieval that was inspired by Google web searches. [56] automatically labels the characters in professional videos while video objects are captured in [34] by static cameras. [57] presents a new window-based technique to identify which regions of an image are likely to contain a foreground object by a combination of several image cues. Moreover, saliency detection methods are used for automatic object identification in image sequences [58–60] while [61,62] try to detect frequently occurring regions and characteristic video patterns.

Only a few evaluation sets exist that are specifically designed for video analysis. The TREC Video Retrieval Evaluation (TRECVID) that is operated by the National Institute of Standards and Technology (NIST) is the leading initiative for video annotation and retrieval evaluations amongst them [63]. This benchmark is annually performed with different retrieval tasks, such as content-based copy detection, instance search, known-item search, semantic indexing, and surveillance event detection. A collaborative annotation tool that already integrates low-level feature analysis and reasoning techniques [64] is thereby used for ground-truth generation. Another evaluation challenge is MediaEval [65] that includes tasks for tagging and geotagging of videos. The Google Challenge [66] is a web video categorization challenge that uses text and social features to identify the category of a video. The PETS dataset presents an evaluation benchmark for surveillance related tasks, such as person counting, tracking, left luggage detection, and loitering [35]. We also use the TRECVID data for the evaluation of this work. Thereby, we stick to the data and evaluation procedures of the instance search tasks 2011 that fit perfectly to the aims of the proposed annotation system.

Similarly to the mentioned works, we perform video annotation on object level to enrich the metadata of these videos and to enable scene-based video retrieval. In contrast to most works, we propose the use of query-dependent keyframes to visualize the retrieved videos. The relevance of these keyframes is significantly higher compared to usual keyframes [15]. We further propose an annotation process where user interaction is only required to initiate new annotations. The described approaches that use external knowledge and identify objects-of-interest might be incorporated into the proposed framework in order to enable a fully automatic object annotation. One fact that makes the proposed annotation process unique compared to the related work is that no defined analysis techniques are used. Instead, we use a flexible framework that can be extended with all kinds of visual features, pre-processing and post-processing steps, matching and machine learning strategies.

2.2. Object Recognition

Recently, researchers have achieved quite good results for the recognition of individual objects [67,68] and object classes, for example, in the Pascal VOC challenge [69]. Best practice approaches are based on visual features that are generated from local region detector-descriptor chains [68,70] and from object models that are suited for recognition with feature matching or machine learning approaches [71]. In the context of object recognition, good visual features should generally compute the same values when they are applied to the same objects and distinct values for different objects. It is well established that feature types mainly differ by the trade-off that they achieve between their discriminative power and invariance. Furthermore, different recognition tasks require different trade-offs, and thus no single visual feature is optimal in all situations [72]. In addition to visual features, [73] has shown that all components of a

recognition approach can have strong influences on the achieved results. Popular approaches include interest point and region detectors [74] (Difference of Gaussian points, Harris–Laplace corners, MSER regions), segmentation-based approaches [75], dense sampled regions [76], and even global approaches can be very efficient for some tasks [77]. Texture-based features (SIFT [67], SURF [68], HoG [78]) are mostly used to describe these interest regions while color-based and shape-based features have been of minor interest in the last years. However, the success of combined features that incorporate color and texture [76] indicate the importance of all feature types. In a further processing step, visual features are then used to form object models such as the popular bag-of-features [79] and sparse representations [80].

The authors of [81] pointed out that the manual process of choosing appropriate algorithms and tuning them for a given task is more an art than a science. A couple of works investigated the automatic selection and customization of recognition approaches from different directions: Attempts to optimize the parameters of specific visual features, like SIFT or HOG, are given in [82–84]. Varma *et al*. [72] proposes a kernel-learning approach to select the best feature combination for a task using a SVM framework that works with all types of features. [85] uses a convolutional neural network to learn new features for each task instead of using manually designed, hand-crafted features while [81] proposed a trainable local feature matching approach that uses a boosting framework. In contrast to these works, we try to automate the simultaneous selection and customization of the entire recognition process with all kinds of visual features, matching strategies, and parameter settings. To the best knowledge of the author, no work exists that presents such a holistic approach for the selection of object recognition approaches in video.

Similar to the mentioned works, we select recognition approaches with the help of initially selected training data. Although it might be possible to tweak algorithms manually for specific tasks and objects, it is much easier and more intuitive to provide appropriate training data instead [81]. According to [82], these data-driven approaches further increase the probability to learn the invariance accurately for specific applications. A potential drawback of data-driven systems is the requirement of training data that is usually human labeled [86,87] or synthetically generated by artificial image transformations [85]. The manual annotation of training data is time consuming and synthetic approaches often do not capture all the invariance of real data. Thus, [74,82] used optical flow tracking as a data collection step to customize the recognition of moving objects without any human intervention. We propose an iterative data collection approach where only one keyframe that contains an object example has to be selected by the user in the first place. More instances are suggested by the system to support the interactive annotation process.

2.3. Recognition Infrastructures

Recently, [88] stated that many object recognition architectures exist in the literature but only a few of them present really generalized infrastructures. They usually propose one specific approach for the recognition of all kinds of objects. However, [89] pointed out that different object types should be treated differently to achieve the best recognition results and [90] stated almost 20 years ago that the time has come to stop searching for a single method that can solve object recognition and to combine different methods instead. We stick to these ideas and propose an infrastructure that integrates multiple

approaches and that can be configured for various recognition tasks individually without a pre-defined processing chain.

Although several computer vision and machine learning toolboxes [91–93] exist that significantly reduce the development effort compared to development from scratch, the needed effort and the required engineering skills are still considerable. A Matlab-like programming interface for content-based retrieval systems is presented in [94] that contains various features, distance measures, and classifiers. This RetrievalLab is user extendable and includes visualization tools. The work that is most similar to the infrastructure proposed in this work (CORI) is [88] which presents REIN: a REcognition INfrastructure for robot vision applications. In this infrastructure, the used algorithms and approaches can also be configured dynamically to adapt the system to a certain recognition task. However, every processing step and the connections between these steps have to be specified in a complex XML format. In contrast, CORI works with rough, human readable configurations and it determines the execution order of the processing steps and their data flow automatically. More degrees of freedom are further provided for the extension of CORI by the support of user-defined data types and the use of one single interface for new algorithms and approaches while three pre-defined interfaces constrain the processing chain of REIN. Furthermore, CORI was developed to run on general purpose computers (PCs on various platforms) while REIN runs on top of the robot operating system ROS. For this reason, CORI is the optimal foundation for the proposed object-of-interest annotation approach, as described in Section 4.

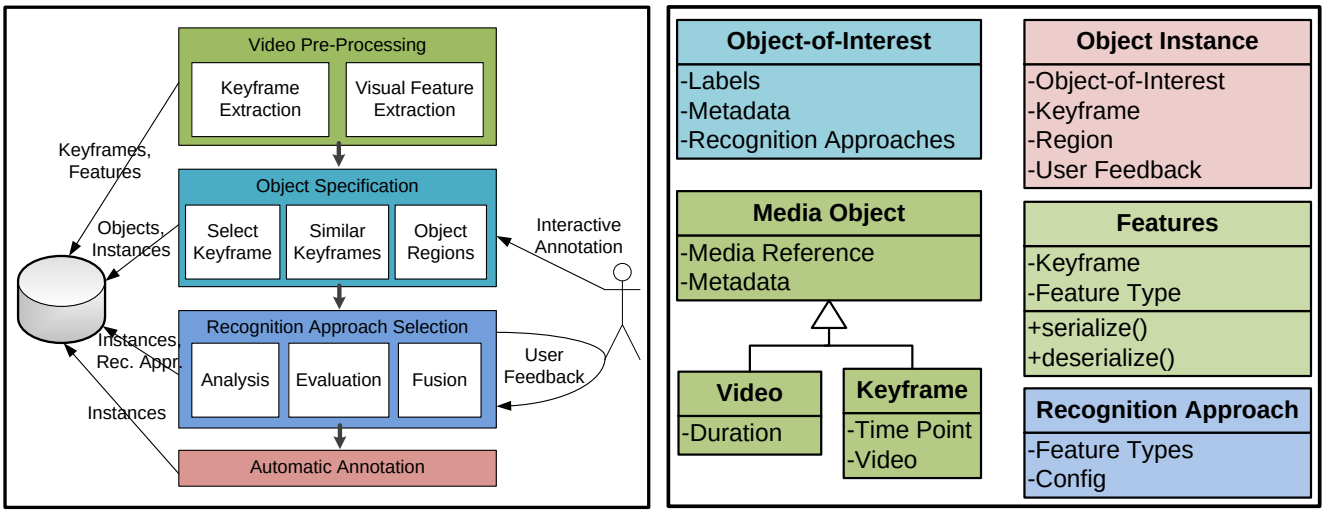
3. Object-of-Interest Annotation

This section presents an easy and user-friendly approach to annotate those scenes in a large video collection that contain a specified object-of-interest. These annotations are well suited to improve text-based video retrieval, especially when a user searches for specific objects or scenes. The left part of Figure 2 shows the corresponding annotation workflow. The entire video collection is first pre-processed to extract keyframes and visual features. Next, a few initial object instances are interactively collected starting from a single keyframe that shows this object. In this process, the region of each object instance is automatically selected. After this, appropriate recognition approaches are selected that fit to the visual attributes of the object in an integrated analysis-evaluation approach. Eventually, the object is automatically annotated in all videos of the online portal.

The right part of Figure 2 shows the database schema that is used to store the results of this object annotation. *Objects-of-interest* are given with one or more user-generated free text labels to enable retrieval tasks. These labels can be ambiguous as the same object might be annotated differently [28], but it is out of the scope of this work to dissolve such ambiguities. Furthermore, metadata (like a user identification, annotation date, and an interaction protocol) are stored together with the selected *recognition approaches* that are used for the automatic annotation of this object. *Object instances* capture the object-of-interest that is shown in a keyframe with an optional region (arbitrary shaped polygon) and a user feedback flag (correct or false annotation). *Videos* and *keyframes* are derived from the *media object* class and they contain a media reference and metadata. The keyframes are either stored in small resolution or extracted on the fly from the video at the specified time point. In the latter case, the media reference is left empty. On the fly generation is especially reasonable when this annotation system is integrated to a video portal that contains suitable video summarization and browsing capabilities. The

schema can be extended by further media object entities, such as shots and scenes, when necessary. Shot entities are, for example, used in the TRECVID 2011 experiments in this work. Furthermore, the proposed annotation schema contains *visual features* that are serializable to binary form which makes them suitable for storage in all kinds of databases and file systems. Further details about the storage of recognition approaches and visual features are given in Section 4.

Figure 2. Annotation workflow (left) with the associated database schema (right).



3.1. Video Pre-Processing

As first step, all videos are pre-processed in an off-line process to enable an efficient annotation process. Keyframes and visual features are thereby extracted from the videos without any user interaction. We expect that a large number of videos has to be processed in this process and that new videos are frequently added. Thus, pre-processing can be performed in a distributed manner on multi-core and multi-machine architectures. The generated data is stored in the database schema described above.

Keyframe extraction: Keyframes are a very common way to present videos and they are often used as input for content-based video analysis. The keyframe extraction approach of [48] and others is similar to the one of this work. A standard shot detection [95] is applied before an activity-based keyframe selection algorithm is performed for each shot. Shots with little activity are summarized with only one or a little keyframes while up to 10 keyframes are extracted for high-activity shots. We measure the activity with global SIFT descriptors similar to the work of Oliva and Torralba [77] that used global features to capture the gist of a scene. As shown in Algorithm 1, the Euclidian distance is used to compare these features and an adaptive threshold that is decreased until no more than 10 keyframes are selected for a shot. During analysis of the first few video shots an appropriate value is automatically selected for this threshold, and thus its start value (default 0.5) only affects these first shorts. Due to the simplicity of these global features, keyframe selection performs faster than real time.

Feature extraction: The selected keyframes are then used to extract various visual features for object specification, recognition approach selection, and object annotation. Note that the proposed system is not restricted to the use of specific features and recognition approaches, but the following

features are currently integrated to the system: SIFT [67], SURF [68], Gabor Wavelets [96], MPEG-7 DominantColor, ColorLayout, ColorStructure, and EdgeHistogram [97] features. These features can be extracted globally from the entire keyframe or locally from dense sampled regions [73], Difference of Gaussian points [67], MSER regions [70], Viola–Jones faces [98], histogram of gradients body regions [78], and from image segments using a mean–shift segmentation [75]. Furthermore, a set of image pre-processing, region filtering approaches and object models are used. As feature extraction takes time and feature storage takes memory, it is reasonable to select only a subset of these features for analysis of specific video domains. The used recognition infrastructure (see Section 4) enables the selection of such sub-sets even on a running system without high development and redeployment efforts.

Algorithm 1 Keyframe Selection

Input:
vector<keyframe> $frames$ = every 10^{th} frame of the shot
float $threshold$ = threshold of last shot
Output:
vector<keyframe> $keyframes$

```
repeat
  keyframes.clear()
  for all f in frames do
    boolean lowActivity = false
    for all k in keyframes do
      if EuclidianDistance( globalSIFT(f), globalSIFT(k) ) < threshold then
        lowActivity = true
      end if
    end for
    if lowActivity == false then
      keyframes.add(f)
    end if
  end for
  if keyframes.size() <= 2 then
    threshold = thershold * 1.25
  else
    threshold = thershold * 0.95
  end if
until keyframes.size() <= 10
```

3.2. Object Specification

In the second annotation step, users can interactively select a few keyframes that contain the object-of-interest. These keyframes are used to select an appropriate recognition approach including region detector-descriptor chains, matching strategies and dissimilarity measures as well as the parameter settings of all components. Thus, it is important to collect examples of the object that represent the appearance of this object in all videos of the portal well with appropriate difficulties and levels of abstraction. If we want to annotate all cars independent of the used camera view, for example, then the

initial object-set should not only include blue SUVs shown from a frontal view. The object specification is composed of following three steps.

Select keyframe: First, a single keyframe of the object-of-interest has to be selected in one of the videos. This can be done when the user is browsing through the extracted keyframes, while she is watching a video, or in any other way that is provided by the surrounding video portal. The prototype that is presented in Section 5.1 allows keyframe selection in a simple interface where the thumbnails of an entire video are shown. The actual position of the object within a selected keyframe is automatically computed in a later processing step, described below. Generally, examples of an object might also be used from outside the video collection but this is not supported yet because we need to know from which video and keyframe the first object example stems.

Algorithm 2 Suggest similar keyframes

Input:
vector<features> $positives$ = features of the selected keyframe
vector<features> $negatives$ = features from keyframes without the object
integer $videoId$ = video id of the selected keyframe
integer $keyframeId$ = id of the selected keyframe
Output:
vector<keyframe> $suggested$

for $i = 1$ to numberOfKeyframes($videoId$) **do**
$features$ = loadFeatures($keyframeId + i$)
$distPos$ = nearestNeighbour($features$, $positive$)
$distNeg$ = nearestNeighbour($features$, $negative$)
if $distPos > thresholdA$ and $distPos/distNeg < thresholdB$ **then**
$suggested$.add($keyframeId + i$)
end if
$i = -i$
if $i < 0$ **then**
$i = i - 1$
end if
end for

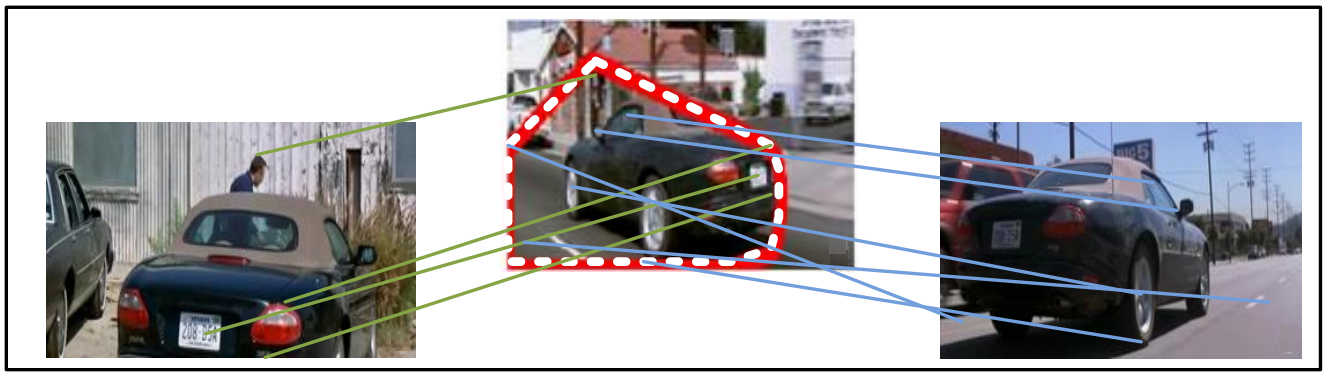
Similar keyframes: After the user has selected a keyframe that shows the object, we start to search for further examples of this object from nearby keyframes according to Algorithm 2. This search starts in the video of the selected keyframe but, if necessary, it can be expanded to related videos that stem from the same user or that have been labeled with similar tags. Due to the high run-time requirements of this process (the user is actively waiting for results to mark them as positive or negative examples) we use a fast bag-of-features matching (BoF) with dense sampled SIFT and ColorLayout features. Feature matching of an investigated keyframe *K* against all positive and all negative object instances *O* is thereby performed with a nearest neighbor strategy and the histogram intersection:

$$histogramIntersection = \sum_{i=0}^{i<cbSize} \max(K_i, O_i) - O_i \quad (1)$$

where *i* is the current codebook index of both BoFs. This histogram intersection decrease the importance of background objects that are shown in both images compared to traditional distance measures such as Minkowski distances and statistical measures [99]. Keyframes are suggested to the user if the distance of the current BoF to nearest positive neighbor is below a certain threshold and the ratio between the nearest positive neighbor and the nearest negative neighbor is above another threshold. The suggested keyframes are shown to the user as soon as they are available. Instantly after the user provides feedback (if a keyframe contains the object-of-interest or not), the visual features of this frame are also used for further keyframe suggestions. This process is terminated after a few object examples have been identified. Due to the needed user interaction, we designed the framework in a way that a small number of object examples (typically between 2 and 6) is sufficient.

Object regions: After keyframe selection, we try to compute the rough position of the object-of-interest in these keyframes automatically with a relaxed region matching. In this process, we use SIFT descriptors [67] extracted from up to 300 Difference-of-Gaussian (DoG) points using a high-contrast filter in order to match each keyframe of the set against each other. This matching is done with a nearest neighbor distance ratio strategy using Euclidian distance as proposed in the original SIFT approach but with lower threshold ratios. Thus, it is also possible to generate matches for less similar descriptors that stem from objects of the same class. After this, we compute the object region in all selected keyframes with the convex hull around all matching DoG points according to our prior work in [100]. The result of this region matching are shown for the top keyframe in Figure 3.

Figure 3. Automatic detection of object regions from a few keyframes.



3.3. Recognition Approach Selection

In the automatic approach selection, we compute the results for different features and recognition approaches, evaluate them, and combine the best ones. Optionally, we further present the results to the user to get a feedback. As shown in Figure 2, this process consists of the three steps: analysis, evaluation, and fusion. The initial object examples are used to compare different recognition approaches against each other and, if user feedback is given, these steps are iteratively executed to improve the achieved results.

As specified in Algorithm 3, appropriate approaches are selected after the recognition results of all approaches have been individually generated and evaluated. This algorithm uses the popular leave-one-out strategy [71] to match all object examples except one against the keyframes of all videos

and the left-out object in each iteration. The position of this left-out object within all results is then used for evaluation. The algorithm can be applied to analyze a few hundred hours of video at once. However, it is also reasonable to start the analysis on a single video. In each iteration, an ordered list with object instances is returned and one combined list per recognition approach is then generated using the same fusion that is later used to combine the selected recognition approaches for the investigated object. In the following, details about analysis, evaluation, and fusion are given.

Algorithm 3 Recognition Approach Selection

Input:
vector<recognition approach> $approaches$ = recognition approaches to investigate
vector<instance> $objects$ = the initially selected object instances
vector<keyframe> $collection$ = videos to annotate
Output:
vector<instance> $results$

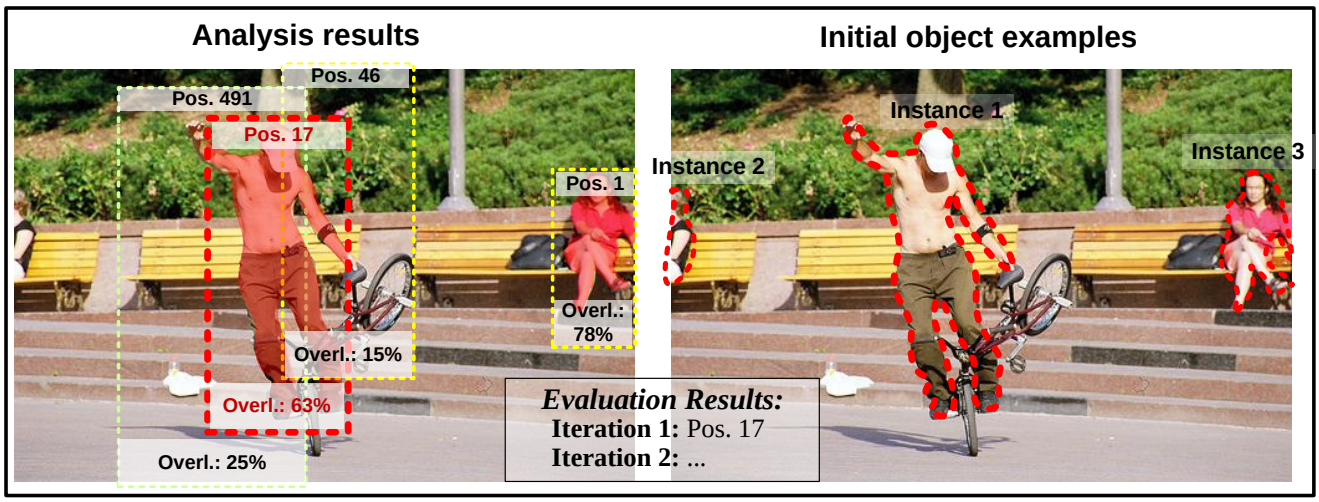
for all a in $approaches$ **do**
  vector<features> $resultsA$
  for all $leftOut$ in $objects$ **do**
    if $leftOut$.positiveExample() is false **then**
      continue
    end if
    $query$ = $objects$ - $leftOut$
    $testset$ = $collection$ + $leftOut$
    $orderedResults$ = analysis($query$, $dataset$)
    $position$ = evaluate($orderedResults$, $leftOut$)
    $eval$.add($position$)
    $resultsA$.add($orderedResults$)
  end for
  $result$.add(fusion($resultsA$))
end for
vector<setup> $bestApproaches$ = selectBest($eval$)
$results$ = fusion($results$, $bestApproaches$)

Analysis: In the analysis step, we try to recognize the selected object in all keyframes with a couple of different recognition approaches. Object instances are generated for each recognition approach and ordered according to their recognition probability. Instances with a high recognition probability are given on top of this list. As mentioned before, analysis is not restricted to specific recognition approaches, but the following approaches have been used in the experiments of this work with the features mentioned in Section 3.1. On the one hand, we have the matching strategies: nearest neighbor, k-nearest neighbor, nearest neighbor distance ratio [67], thresholding as well as their combinations. All of these strategies have been used with the dissimilarity measures Manhattan distance (L1), Euclidian distance (L2), Canberra metric, Jeffrey divergence, Psi Square, cosine function based similarity, and histogram intersection. Please refer to [99] for details about this dissimilarity measures. On the other hand, we used SVMs [71] with different kernels and applied a few post-processing steps, like geometric verification [67] and feature voting. Each approach further contains a number of parameters, such as the

threshold of simple feature matching, that can be tuned to improve the recognition of specific objects. Further details about the actual recognition approaches are given in the related works section.

Evaluation: If no region information is given in the initial object examples, a positive match is generated when the left-out keyframe is contained in the analysis results and we return the position of this match. This keyframe level comparison is possible because every object instance includes the information in which keyframe it was detected, as shown in the annotation schema of Figure 2. When region information is given, all object instances are individually compared against the left-out object by computation of the region overlap using a polygon intersection. Figure 4 shows this evaluation for a task where three object instances ("persons") have been selected (right image) with exact object boundaries. The suggested object instances from a recognition approach (left image) are given as bounding boxes. In this figure, the person shown in the middle of the keyframe was used as left-out object. Thus, the evaluation tries to find out if this object instance was returned and at which position in the ordered result list. During evaluation, every object instance that was generated from this keyframe is compared against the left-out instance using polygon intersection. Those instances that are above a certain intersection (default value is 60%) are true positives (bold bounding boxes) and all other are false positives (thin bounding boxes). The position of the best true positive is then returned, which is 17 in the shown example. Note that the proposed evaluation process does not guarantee that the best recognition approaches are selected. However, the experiments of Section 5 show that it works well if the initial object examples present the object in the videos well.

Figure 4. Evaluation for the first iteration of the one-left-out approach.



Fusion: Finally, the one-left-out results of each approach as well as the recognition approaches that achieved the best evaluation results are combined using following fusion strategy.

$$rank_i = best_i * \#lists * \frac{\#entries_i}{\#possibleEntries} - best_i \quad (2)$$

In this fusion, we compute the rank of each object instance *i* in the result *lists* based on its *best* position in all result lists and the number of *entries* of this instance in all lists in relation to the *possible* amount of entries. Depending on the recognition approach, multiple entries of the same instance can either be given

in one result list or not. After this, we order all instances according to this ranking value. The intuition behind this fusion is to order the instances similar to a zip fastener starting from their best entries. The order of entries with the same best value (originating from different result lists) is determined from the total number of entries. In an alternative fusion approach we additionally used the number of top x ranked entries (x $\in$ 50, 100, 500).

3.4. Automatic Annotation

The proposed recognition approach selection is only efficiently applicable for small video collections because the analysis of each approach takes some time, compare Section 5.2. Thus, we annotate only a couple of videos in this process and propose the annotation of further videos in an automated post-processing step. During this analysis, the selected annotation process is used to annotate an object only in the best matching videos. In contrast to the earlier steps of the workflow, we suppose that this annotation is performed without a human in the loop and that results of this annotation might only implicitly be inspected during video retrieval. The annotation of false object instances in this step will not significantly decrease the video retrieval performance, as user-generated annotations are always preferred to answer video queries. The automatic annotation process can be easily distributed over an arbitrary number of analysis machines, which is especially important for large-scale video portals that are incrementally growing.

4. Recognition Infrastructure

In this work, a flexible object-of-interest annotation framework is presented that employs specific recognition approaches for different video domains and objects. The configurable object recognition infrastructure CORI [101] provides the foundation of this flexibility. In this infrastructure, simple configurations define how keyframe extraction, the suggestion of similar keyframes, the computation of object regions, and the actual object annotation is performed, as shown in Figure 5. CORI consists of algorithms, descriptors, and tools in its core, and it provides the functionality to extract visual features for training and recognition. The annotation framework further operates on a few object examples and the videos of an online portal.

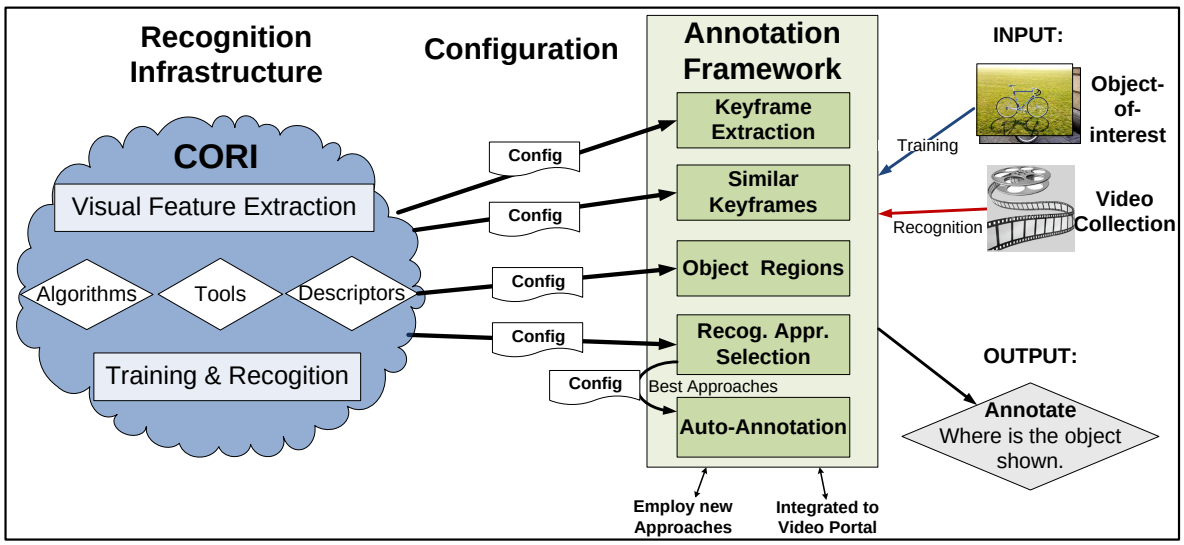
Generally, a set of different skills and expertise are required for the development of object recognition systems. For instance, algorithmic image processing skills are needed to develop feature extraction and matching approaches while application and database engineering skills are required to store these features efficiently in a database and to present the results adequately. CORI facilitates the separation of these heterogeneous parts and enables the integration of new algorithms independent of their later use. Changes of the used recognition approaches can be performed without recompilation and deployment of a running recognition system simply by changing its configurations. This is important because different video portals might investigate different recognition approaches for object annotation and because new recognition approaches can be employed without changing the entire annotation system. Furthermore, considerations about the performance have been taken into account by an intelligent combination of multiple recognition approaches during feature extraction and support of distributed computer architectures. CORI is written in C++ and it supports the use of existing computer vision

toolboxes, such as OpenCV [91], VLFeat [92], and [96]. In the following, we describe how CORI works and how it can be used to support different video portals appropriately.

Listing 1. Configuration of visual feature extraction (left) and recognition (right).

```
[Visual Features]                                  | [Recognition]
name     = Gist                                    | name      = KeyframeExtraction
output   = FloatVector                             | input     = Gist
module   = imagePreprocessing, size_32_32;         | strategy  = Nearestneighbor
           SIFT                                    | distance  = Euclidian
-------------------------------------------------- | ---------------------------------
[Visual Features]                                  | [Recognition]
name     = BagOfFeatures                           | name      = SimilarKeyframes
output   = IntVector                               | input     = BagOfFeatures
modules  = denseSampling, s_0.5_0.3_0.15;          | strategy  = K-Nearestneighbor
           SIFT; ColorLayout; BoFGenerator         | distance  = HistogramIntersection
-------------------------------------------------- |  --------------------------------
[Visual Features]                                  | [Recognition]
name     = localSIFT                               | name            = ObjectRegions
output   = FloatVector, Regions                    | input           = localSIFT
modules  = denseSampling; DoG;                     | descriptors     = FloatVector
           SIFT, highContrastFilter_300            | strategy        = NNDR, ratio_0.3
                                                   | distance        = Euclidian
                                                   | postprocessing = ConvexHull
-------------------------------------------------- | ---------------------------------
[Visual Features]                                  | [Recognition]
name     = Persons                                 | name      = ApproachSelection
output   = FloatVector                             | input     = {BoF,Gist,Persons}
modules  = FaceDetection; HoG;                     | strategy  = SVM, kernel_{RBF,X2}
           GaborWavelets                           | gamma_{1:+0.1:5}, c_{1:*2:8}
```

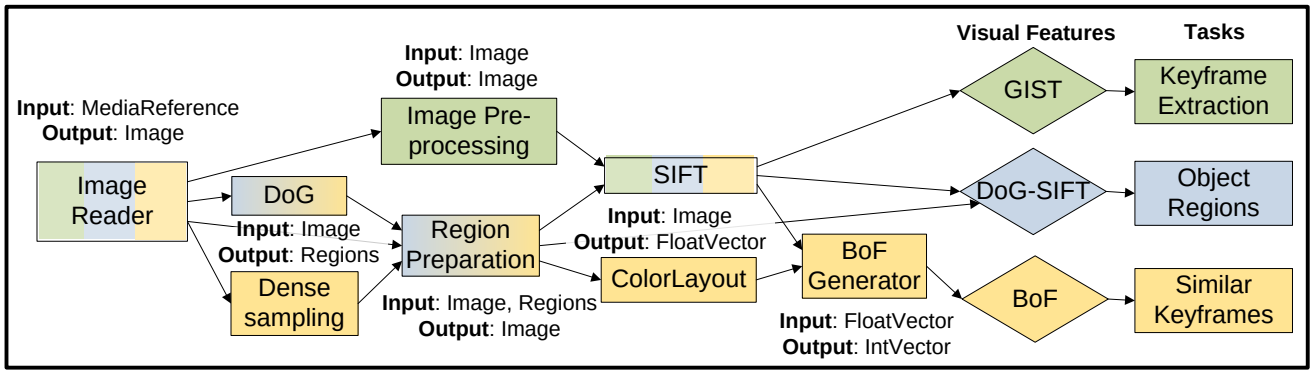
Figure 5. Overview of the recognition infrastructure in the annotation process.



4.1. Visual Feature Extraction

The extraction of visual features from video frames presents the starting point of the proposed object annotation. This extraction consists of several modules for image loading and pre-processing (e.g., to scale and normalize the keyframes), or for region-of-interest detection and description, and these modules have to be executed in an appropriate order. The actual modules differ from task to task and Figure 6 shows the used modules of the first three annotation tasks in a visual feature extraction graph. Global SIFT features (*keyframe extraction*), a bag-of-features approach with dense sampled SIFT and ColorLayout features (*similar keyframes*), and local SIFT features from Difference of Gaussian (*DoG*) points (*object regions*) are used for these tasks.

Figure 6. Visual feature extraction graph.



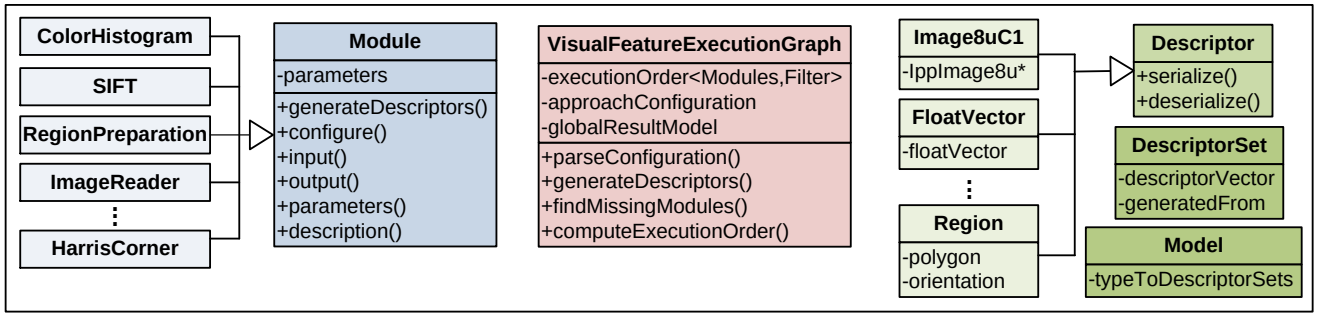
In this feature extraction graph, modules are shown as labeled rectangles and the arrows between them indicate the interconnections of their input and output. The output of each step (e.g., images from the image reader) can be used as input for several other modules and they can be combined to visual features as shown on the right side of the graph. Modules that are used for more than one task are shown in multiple colors in Figure 6. The repeated use of the same intermediate results accelerates the feature extraction significantly and reduces the memory requirements.

CORI constructs such feature extraction graphs automatically from simple configurations to support interoperability without high engineering costs. Listing 1 shows the corresponding configurations where the features of each task are specified in one paragraph (left) and the used recognition approaches (right). In the feature configurations, the *name* is used to identify the visual features for training and recognition and it can be chosen arbitrarily. The comma separated *output* list indicates the visual feature types and the *modules* string specifies the used processing steps. Here, semicolons separate different modules, while commas separate a module from its parameters. The first part of a parameter (before the underline) presents its name and all following parts present its values. In order to specify the use of several values per parameter, intervals and lists of concrete values can be defined. In these cases more complex extraction graphs with many instances of the same module (with different inputs and parameter settings) are generated. This property is especially important for recognition approach selection, where many different recognition approaches and differently parametrized features are investigated for the annotation of each object, as shown in the last paragraph of the right listing side. The syntax of intervals is *"*{*start value : step size : end value* }*"* while lists are comma separated. In addition to the configuration of

modules, *filters* can be specified to use only inputs that fulfill certain criteria. For instance, it is possible to use only regions with a certain gray-level contrast in their neighborhood or to use the 300 regions with the highest contrast, as specified for DoG-SIFT features (Line 17).

Modules: As shown in Figure 7, each module provides methods to explicitly state which input descriptors it expects, which output descriptors it will generate, and which parameters it holds. The internal state of a module instance can be changed using the configure method. The actual algorithms and functionality of a module are hidden in the generate-descriptors method. Each module further contains a description method to explain its functionality. New recognition approaches and algorithms can be added to CORI either as heavy-weight modules (that perform many processing steps internally) or as set of light-weight modules that contain smaller processing steps separately. Light-weight modules facilitate the reuse of intermediate results for succeeding modules. This improves computation and memory efficiency.

Figure 7. Class diagram of modules, descriptors, and extraction graphs.



Descriptors: Descriptors present all data of the recognition infrastructure in a type-safe way. They are implemented as smart pointers (Boost library) to prevent unnecessary copy operations. Furthermore, they provide binary serialization and de-serialization capabilities, see Figure 7. Descriptors of the same type can be stored in descriptor-sets together with the information from which modules they stem. Several descriptor-sets can be grouped together in a model where they are stored in a map according to their type. Models are used as IO-arguments for the generate-descriptors method of modules and visual feature extraction graphs and they provide methods to add, remove, and access descriptor-sets and single descriptors. Most visual features are described by vectors of basic data types, such as float-vectors. More complex features can be stored in user-defined descriptors or as combinations of simple descriptors in a model. User-defined descriptors can be added to CORI by inheritance from the descriptor base class and by overwriting the serialization methods. In addition to vectors of the basic data types, images, parameters and recognition results are the most important descriptors. Recognition results include a region polygon that indicates where the object-of-interest lies in the media object (keyframe) together with a recognition probability, compare object instances in Figure 2.

Extraction graphs: The automatic graph construction starts with the initialization of a new *visual feature extraction graph* shown in Figure 7. These graphs then parse the configuration for each recognition approach separately and perform a validity check. Next, each new module is compared to already existing modules in the graph according to their type, input and parameter settings. Modules that did not overlap with existing modules are then added to the graph and the specified filters of each module

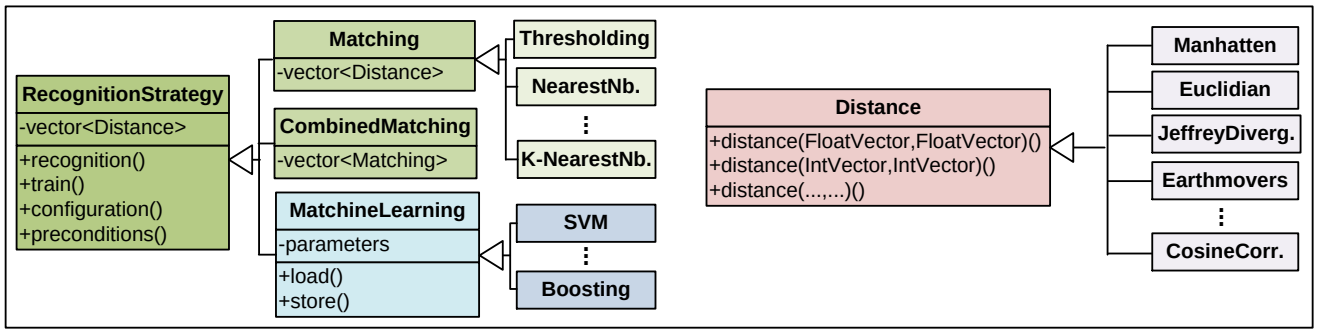
are set. If several filters exist for the same module, an aggregated filter is generated for this module and more specific filters are handed over to the succeeding modules in the execution graph if necessary.

4.2. Training and Recognition

Object recognition systems usually consist of a training stage and a recognition stage. Training is performed prior to recognition in order to learn appropriate object models. The right side of Listing 1 shows that each recognition setting is configured in its own paragraph. The *name*, the used *input features* and *descriptor types*, the *recognition strategy*, and the used *distance measures* are specified. Furthermore, it is possible to specify certain post-processing steps, as shown for object regions in Line 19 where the convex hull is used to cluster individual feature matches to a combined region, as described in Section 3.2.

During training and recognition, all modules of a visual feature extraction graph are executed in the calculated order. Thereby, the required inputs are selected for each module independently from the IO-set with respect to the specified input filters. After a module was executed with the generate-descriptors method, its output is again stored in this IO-set. Finally, the specified visual features of each recognition approach are selected from the IO-set and returned under the given name. These visual features can be further used to learn new object models, for example for machine learning strategies. As mentioned before, this model learning step is not necessary for feature matching strategies. Instead, post-processing modules might be used to transform the generated feature matches to an object model, for instance with a geometric verification as proposed in [67].

Figure 8. Class diagram of recognition strategies.



Recognition Strategies: CORI supports three different types of recognition strategies: feature matching, combined matching, and machine learning approaches. Simple feature matching strategies are nearest neighbor, k-nearest neighbor, or thresholding for one input feature type using one distance measure. The base class of all recognition strategies is shown in the left side of Figure 8. Matching strategies are implemented independently of the used distance measures and the same code is used to compute feature matches with different distance measures for different feature types. Furthermore, it is possible to extend CORI by both new strategies and distance measures, without the need to change existing code. The clue behind this functionality is the distance class that contains functions to compute the distance between two features of the same type. New distance measures have to be derived from this base class and it is necessary to overwrite the distance functions of those feature types that they

support. Matching strategies contain one or more distances that are used with the current descriptor type. Combined matching strategies contain several matching instances that are executed one after another in the recognition process and combined afterwards. In contrast to feature matching, machine learning approaches are similarly implemented as modules for visual feature extraction and, in fact, they are also derived from the module class of Figure 7.

4.3. Integration in Video Portals

There is not only one single way how the proposed framework can be integrated into a video portal, but several integration levels might be reasonable that fit to the available resources and software architectures of different video portals. In the following, we describe four different integration levels. The first, minimal integration only consists of a manual object annotation that uses the presented keyframe clustering algorithm and the database schema. Thereby, it is sufficient to store one keyframe (or its timepoint) for an annotated object to answer video queries that include the object label as keyword. The second integration level presents an interactive annotation where a video portal has to store all keyframes that a user has selected for an object with the help of the keyframe clustering and keyframe suggestion algorithms. The region detection approach of Section 3.2 can be optionally used to support the annotation process. In this integration level, video retrieval became possible on a scene level, which means that users can navigate between different video scenes of an annotated object. The third integration level additionally uses the recognition approach selection of Algorithm 3 to annotate the object-of-interest automatically in the first few keyframes that are returned by the selected approach. In this case, it is recommended to store the information if an object instance has been annotated by the user or not and to prefer user-approved instances to answer video queries. The fourth and last integration approach includes an user feedback step to classify the automatically annotated object instances as true or false annotations. The benefit of the later integration levels is a more complete instance-list of the object-of-interest and the possibility to (semi-)automatically annotate this object in newly uploaded videos.

Video portals have to store the annotated objects-of-interest and their instances (compare Figure 2) in order to improve the retrieval process. The computation of clustered keyframes and the feature extraction can be done at the client side (user of the video portal) before video upload or at the video portal severs after a video was uploaded. In the case of client side analysis, video portals do not have to bother about resources and scalability issues. However, we suppose that a stand-alone application is needed in this case instead of a single website. The storage of visual features at the server side is only required if object annotation should be performed in several videos at once.

5. Experiments

Ideally, the experiments of this work would be performed on an existing online video portal but this is out of the project's scope. Thus, we developed a prototype that integrates the proposed annotation framework and performed a case study with two video datasets on it. This prototype can be downloaded from [102] and the reader is encouraged to perform experiments with his own videos. Furthermore, we participated at the instance search task of TRECVID 2011 to compare the proposed framework against other systems.

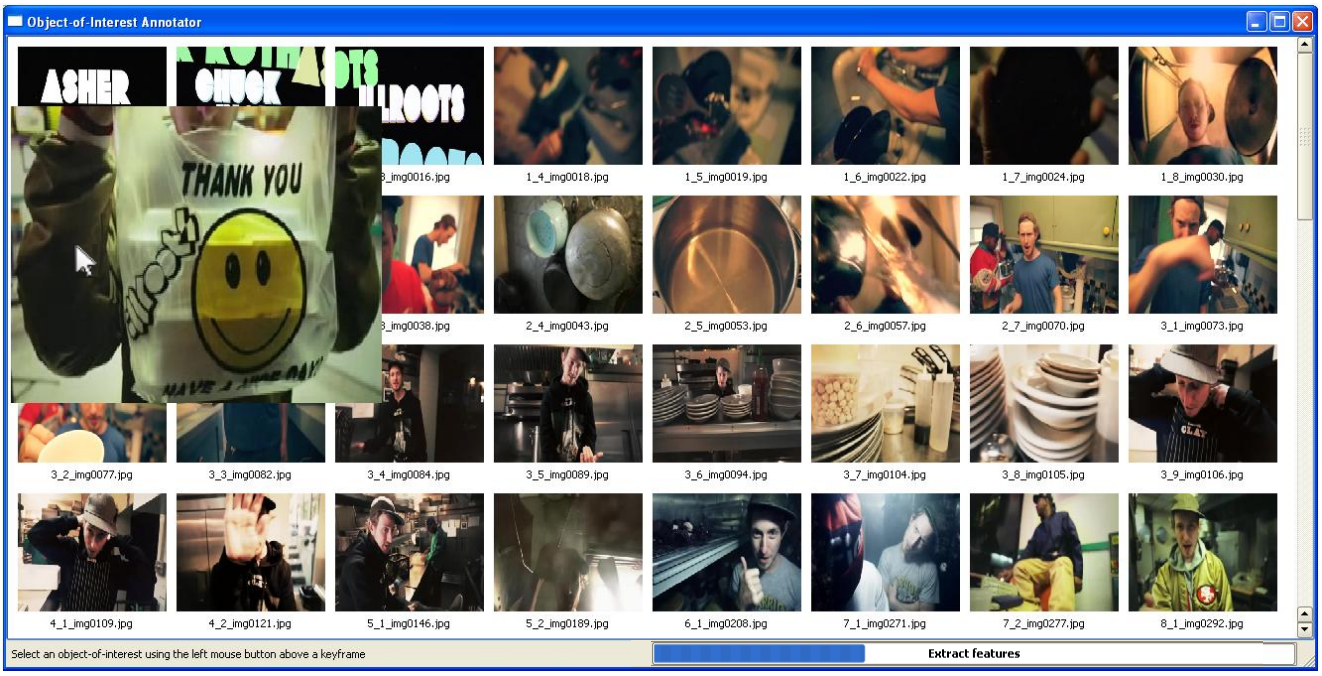
5.1. Annotation Prototype

The video annotation prototype is implemented as client application with a simple user interface. This prototype is free available as Windows distribution [102] and was developed in C++ with the Qt application framework [78]. The user interface of the prototype appears very sparsely to suggest the "touch and feel" impression of a web application that is shown in a usual browser. We suppose that this is the preferred way in which video portals might use the proposed framework. The only client application item in the prototype is a status bar that displays interaction hints and that contains a progress bar to visualize the current processing state. After a video was selected by the user for the first time, shot boundary detection and keyframe extraction is performed. In this process, the free Shotdetect software [103] and FFmpeg [104] are executed in parallel before keyframe clustering takes place. Next, the prototype extracts visual features from these keyframes and they are stored in a sub-directory of the video file. When a video is opened for the second time, the clustered keyframes and extracted features are loaded without further analysis.

As shown in Figure 9, the clustered keyframes are shown in a compact matrix form in a scroll widget with 8 thumbnails per line. Typically, 5 to 7 lines fit to the screen which means that a user can see about 50 thumbnails at once. This representation makes it easy to navigate over a large number of keyframes. Below each keyframe a label is displayed that consists of the cluster id and a reference to the video timepoint. A detailed view of a keyframe appears when the mouse cursor is placed above its thumbnail and users can start to annotate objects by pressing the left mouse button above a thumbnail. The selected keyframe is then shown in the first line of the widget and the suggested keyframes that are supposed to contain the same object are shown in the lines below. Users can select further keyframes as positive or negative examples using the left and right mouse button above a keyframe. Negative examples are also shown in the first line of the widget but starting from the right side. As described in Algorithm 2, keyframe suggestion uses both positive and negative examples, and thus we recommend the annotation of negative examples especially in those cases where the object-of-interest is not contained in the first suggested keyframes. Moreover, negative examples are helpful if two or more objects are shown on several positive examples although only one of them is the object-of-interest. This happens frequently during the annotation of a person because different persons are often shown together and in front of the same location. As soon as two positive keyframes have been selected, the region detection algorithm of Section 3.2 takes place in the background to identify the object regions in all positive keyframes. These regions are then shown as red polygons in the prototype.

Users can finish the annotation of an object by pressing any button on the keyboard. After a label has been entered for this object, users can annotate further objects starting from the original keyframe representation. For each annotation, the prototype stores a few information (like the object name, the selected keyframes, and the needed annotation time) in simple text file. Moreover, it is possible to configure the used visual features and recognition approaches for the three algorithmic steps (a) keyframe clustering, (b) keyframe suggestion, and (c) region detection of the prototype according to Listing 1 in Section 4. New algorithms and recognition approaches can be integrated into the proposed recognition infrastructure and they can be used without recompilation of the prototype simply by replacing one dynamic linked library (AnnotationFW.dll).

Figure 9. Object-of-interest selection in the annotation prototype (under the mouse cursor).



5.2. Case Study

We evaluated the interactive annotation process and the proposed algorithms of Section 3 in a case study with the presented prototype using two video datasets. The first set consists of 20 short videos that have been randomly downloaded from YouTube. The second set contains 10 longer videos including feature films, documentations, and an animated video, as listed on top of Figure 10. Although the overall runtime of these videos is only about 15 h, we captured many characteristics of the annotation process including the number of average annotations per video, the needed user interaction time, and the storage requirements. We assume that these characteristics also apply to large video portals because each video is independently annotated while the automatic annotation of multiple videos (Section 3.4) is not considered in this case study.

Figure 10. Number of keyframes that contain the object-of-interest within 48 keyframes.

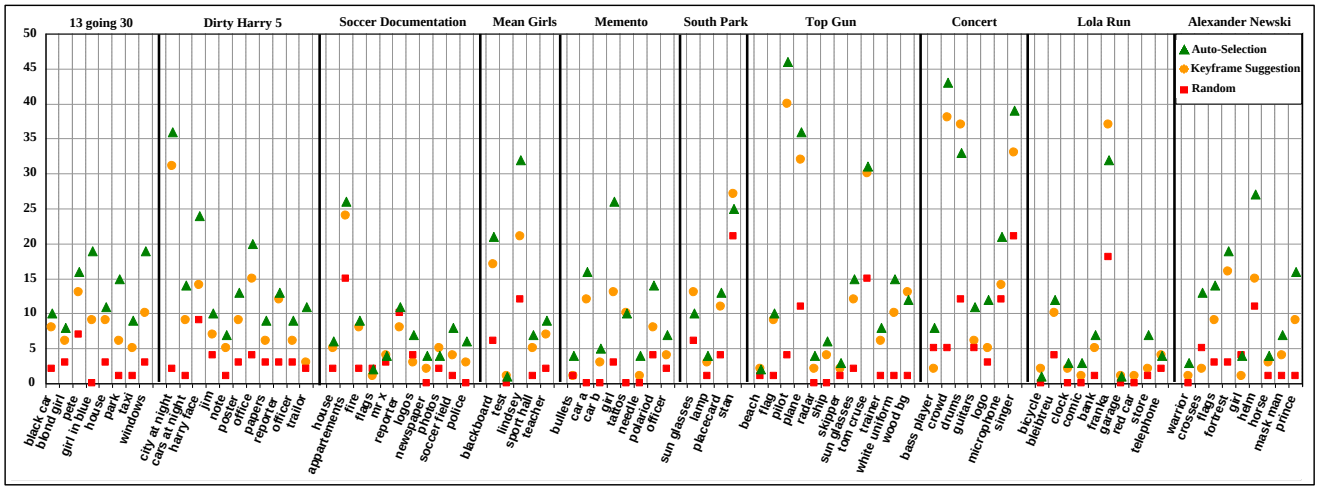


Table 1 gives an overview of the used video datasets. As shown, the average video duration of the short videos from YouTube was about 5 min with the shortest video of 31 s and the longest of 22 min. An average number of 68.5 keyframes are clustered from these videos, which means that on average one keyframe is used every 4.4 s while a larger interval of 7.6 s is given in the second dataset. The different cluster rates of the two dataset indicate a higher change frequency of the short videos. In each video, we tried to annotate a few objects that seem to be important for the entire video. Overall, we annotated 153 different objects in both datasets combined. The number of annotated objects per video highly depends on the video duration, and thus almost 10 objects are annotated for an average video of the second dataset while less than 3 objects are annotated in the short videos. In both datasets persons are the most frequently annotated objects followed by the category "location" that includes soccer fields, beaches, and specific buildings. The residual annotations either depict specific objects or object classes. Frequently annotated examples of these categories are specific cars (e.g., the car of a main actor) and a more general class of cars, such as taxis. The bottom of Figure 10 lists the labels of 88 annotated objects from the second dataset.

Table 1. Statistics about the used video datasets.

Dataset	#	Duration (min)			Clustered Keyframes			Number of Annotations				Categories				Annotation Time (secs)			Annotated Example	
		Avg	Min	Max	Avg	Min	Max	#	Avg	Min	Max	Persons	Locations	Specific Object	Object class	Avg	Min	Max	Positive	Negative
Short videos	20	5.02	0.52	22.00	68.5	13	177	55	2.75	1	6	15	10	17	13	32.56	9	60	5.02	1.8
Long videos	10	77.13	27.85	108.73	606.1	234	889	98	9.8	5	15	31	15	22	28	47.23	21	146	5.35	2.62

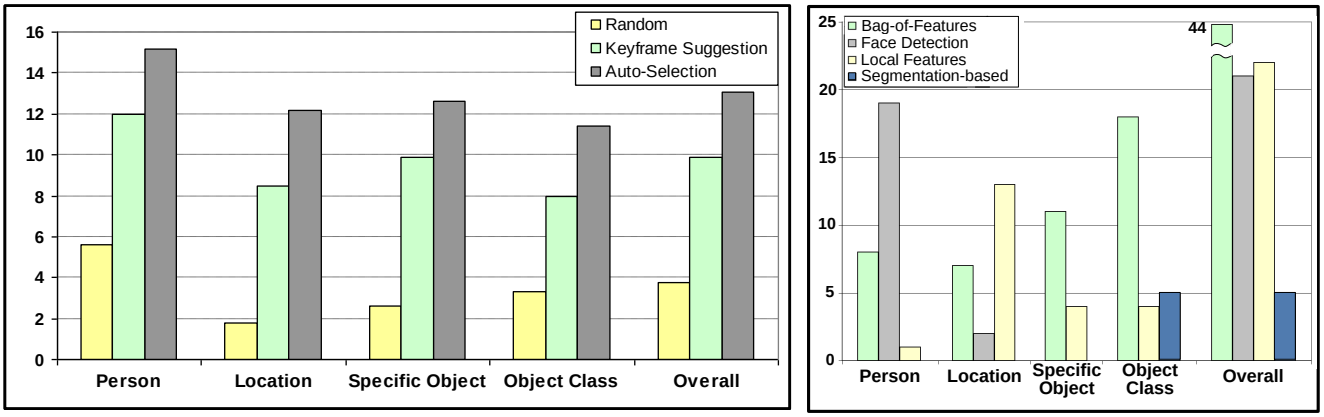
The variation of the used annotation time between the two datasets is rather small (33 to 47 s per object annotation) considering the large difference in size of clustered keyframes. The given annotation times include the selection of all positive and negative examples, and thus the time that a user needs to identify a new object that she wants to annotate. Obviously, it takes a longer time to select the first positive example when a larger number of keyframes are shown to the user and when more objects are annotated in one video. Considering the higher number of positive and negative examples that are selected for the second dataset, we conclude that the proposed interactive object annotation works efficiently even for entire feature films. Longer annotation times (up to 146 s) are often caused by the fact that no keyframe was suggested in the top ranks by Algorithm 2 that contains the initially selected object-of-interest. Furthermore, the average annotation times are slightly longer (55 s) for the gray-level movie "Alexander Newski" because object identification is more difficult when no color information is given.

After each selection of a positive or negative example, keyframe suggestion is performed on the residual keyframes to support the annotation of further object instances. As shown in Table 1, more than 5 positive examples were selected in an average annotation, although these statistics include some annotations where only 1 or 2 positive examples were selected. On average the suggested keyframe on position 6.1 was selected as positive example and in 37.5% it was ranked on the first position. It turned out that the selection of the second positive example is the most critical one, especially when a large difference between the initially selected object and all other instances exist. Sometimes, the second selection was placed above the 200th suggested keyframe. In these cases, the selection of a few negative examples usually improves the situation.

Figure 10 and the left diagram of Figure 11 show the results of an evaluation that we performed using those interactively annotated object instances of the second dataset that contain more than two

positive examples. In these experiments, we manually counted the number of keyframes that contain the object-of-interest (a) within the top 48 suggested keyframes of Algorithm 2, (b) within the top 48 keyframes that are returned by the automatically selected recognition approach of Algorithm 3, and (c) within a random selection of 48 keyframes from the video. We chose the number of 48 keyframes because this was the amount of visible thumbnails on one screen of the prototype. This experiment setup presents the typical interaction process for user feedback, as described in Section 3.2. Note that the randomly selected keyframes have no order and that the ordering (within the 48 keyframes) of the other runs is not important for user feedback as long as all keyframes are shown on one screen because users do not have to investigate the keyframes line by line. Thus, we omit to compute the average precision of correct keyframes and give the absolute numbers of correctly returned keyframes instead. These numbers can be interpreted as precision (using a division by 48) although the actual number of correct keyframes that exist for an object is unknown and only a small fraction of the objects are supposed to be shown in 48 keyframes or more.

Figure 11. Correct keyframes (left) and selected approaches (right) per object category.



The left diagram of Figure 11 shows that an average of 10 correct keyframes was returned from keyframe suggestion in the second dataset while almost 13 correct keyframes were detected from the automatically selected recognition approach. The number of correct keyframes from the random keyframe selection was slightly below 4 on average, though up to 21 for the annotation of some persons, as shown in Figure 10. However, random keyframe selection performed significantly worse than the object recognition approaches for all object categories and it achieved a slightly better result than keyframe suggestion for less than 6% of all objects. The automatically selected recognition approach outperformed keyframe suggestion in 89% of all annotations and a constant average improvement of about 30% was achieved for all object categories. The right diagram in Figure 11 shows the distribution of recognition approaches that make use of bag-of-features, face detection, local feature matching, and segmentation-based features for each object category. Bag-of-feature approaches were selected over all categories. For the annotation of persons, face detection approaches (using Gabor wavelets, SIFT, and color features) have been selected most often. Local feature matching approaches, such as the original SIFT matching [67], are the predominant approaches for the location category while segmentation-based approaches (color and texture features from mean-shift regions) were only selected to annotate a few general object classes. In this case study, object regions (Section 3.2) have not been used for the

recognition approach selection. However, the detected regions are shown in the positive examples of the prototype and they are useful as rough feedback for the quality of the selected examples. If the regions are too big, it is very likely that the object's background is not separately selected as negative example. In examples where the region of wrong objects is shown, negative examples of these objects might be useful. Figure 12 shows a few region detection examples.

Figure 12. Region detection results for 4 objects.

Last but not least, we measured the processing time and memory consumption of the individual analysis steps on a desktop PC with a Intel quad-core CPU with 2.66 GHz and 8 GB of main memory. As shown in Table 2, the run times of keyframe clustering heavily depend on the input video size. Videos with a size of 320 × 240 pixels are clustered 14.8 times faster than real time, videos with 640 × 480 pixels 2.7 times faster than real time, and the clustering of a one hour video with 1280 × 720 pixels needs about 1h and 7 min. These run times include the frame extraction, shot detection, and the actual clustering algorithm. In the prototype, frame extraction with FFmpeg operates only on a single CPU core and takes between 80% and 90% of the overall clustering time. The clustered keyframes are stored with a resolution of 512 pixels in the longer dimension and consume about 19 MB storage for a one hour video. However, video portals do not have to store all clustered keyframes at their servers, see Section 4.3. The keyframe suggestion step needs about 31 s to extract features for 1 h of video while the suggestion of keyframes for a set of positive and negative examples takes only 250 ms. Region detection needs about 750 ms for each keyframe that was selected as positive example while the run time of the automatic recognition approach selection cannot be given that easily. As shown in Table 2, this time depends on the type and number of visual feature and recognition approaches of interest. In the case study, an overall number of 12 different feature types with 10 different matching approaches were used, which took between 2 and 3 h to analyze 1 h of video. The size of the metadata that is needed to enable object-based and scene-based video retrieval is below 1 KB for an entire feature film, compare the database schema of Figure 2.

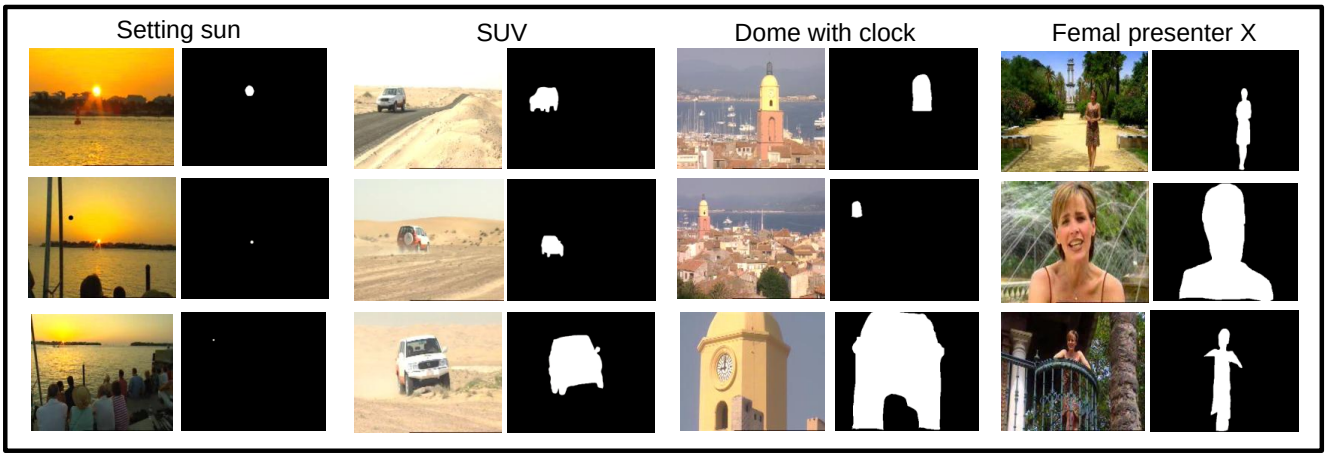
Table 2. The required run-times and storage for an average video of one hour.

Measurement Units	**Keyframe Clustering**			**Keyframe Suggestion**	**Region Detection**	**Auto-Selection**			
	320 × 240	**640 × 480**	**1280 × 720**			***BoF***	***Face Det.***	***Local Feature***	***Seg.-based***
Feature Extraction (seconds)	242	1,315	4,050	31	0.75 /kf	20	125	50	135
Matching (seconds)				0.25		0.2	8	87.5	0.15
Storage (KB)	19,000	19,000	19,000	620	40	445	3,500	15,000	185

5.3. TRECVID

The main purpose of the presented framework is the annotation of video objects in order to improve the text-based video retrieval of online portals. Unfortunately, it is very difficult to compare the results of different annotation systems against each other because the output of these systems vary in several matters, such as the representation of objects. Furthermore, users can annotate different objects in the same video, give different labels to the same objects, and operate with a different annotation speed and accuracy. However, these limiting factors do not exist for example-based video object retrieval, and thus initiatives like the TRECVID instance search tasks emerged lately to provide benchmarks for retrieval systems. We took the opportunity to participate at this pilot task that exists since 2010 with the goal to retrieve video shots of specific objects, persons, or locations. As shown in Figure 13, the used object queries are similar to the initially selected objects of the case study in Section 5.2, though a perfect segmentation is given as binary mask. Each query contains between 2 and 6 keyframes of the object-of-interest, extracted from nearby video sequences.

Figure 13. Object examples of the instance search task.



The National Institute of Standards and Technology (NIST) performs the ground-truth generation and the evaluation of this TRECVID task. In this ground-truth, 1830 video shots of 25 different objects (see the bottom of Figure 14) are annotated in about 136 h of video. The used BBC rushes video collection contains raw material from which programs and films are made in the editing room. These videos include material for several dramatic series as well as for travel programs. A shot detection was already performed by NIST to produce a collection of 20982 short clips and it was specified that these clips have to be treaded independently during analysis. We extracted about 70000 keyframes from these shots using Algorithm 1 for further analysis. The output file of each video object query contains up to 1000 ordered shots with those shots on top that are most likely to contain the object-of-interest. In the evaluation, NIST computes the average precision for each query together with a combined mean average precision for all queries [63]. These measures indicate how much of the annotated video sequences are recognized with a higher weighting of the topmost list entries.

Figure 14. Evaluation results for each query.

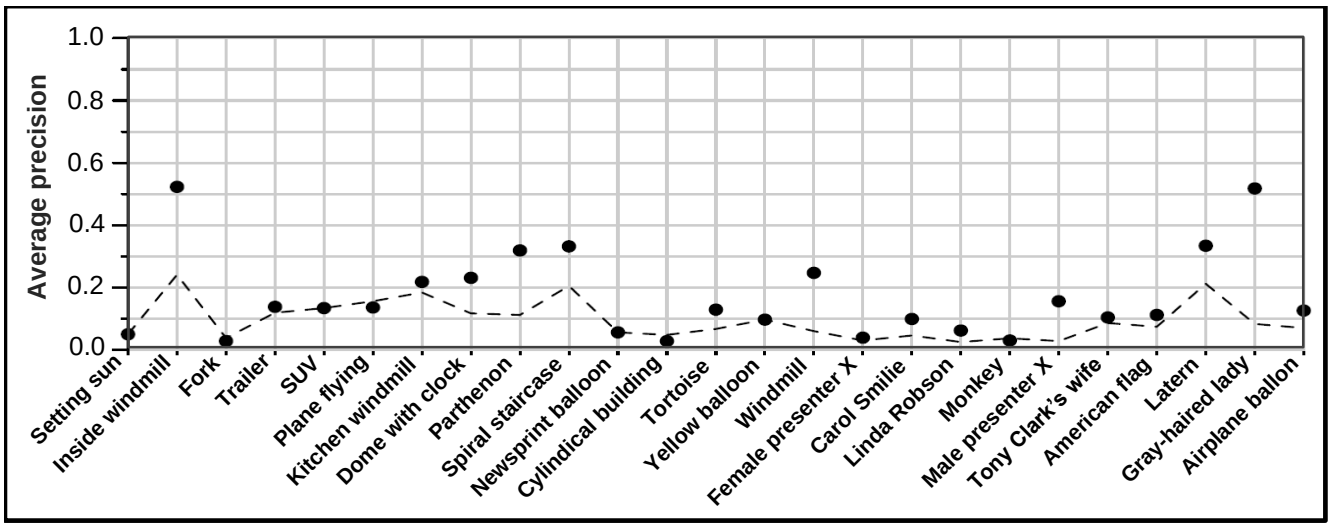
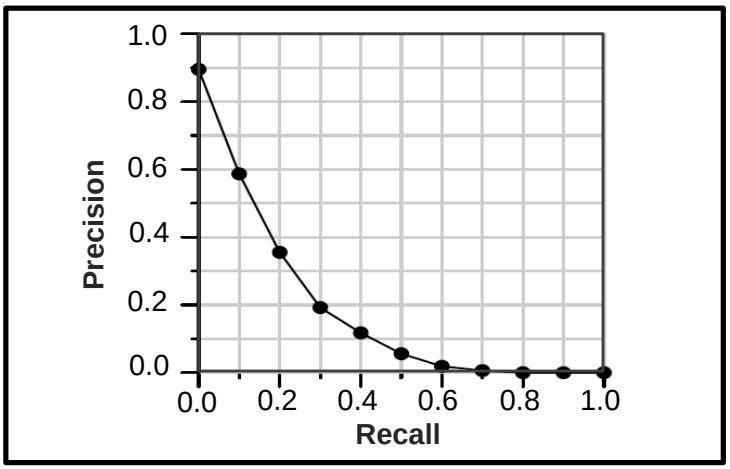


Figure 14 shows the achieved results for each object as black dots together with the median results of all 14 participating teams as dashed line. Our results are generally above this median except for a few objects where we achieved similar results. Overall a mean average precision of 0.170 was achieved and we retrieved up to 847 of the 1830 annotated ground-truth shots correctly. The left side of Figure 15 shows the interpolated recall-precision curve over all queries and, on the right side of this figure, the precision for the first n shots of the output list. Both diagrams indicate that the proposed framework works accurate to identify correct shots of a query object on top of the result list. About 15% of all objects instances are detected with a recall above 50%. These instances are good candidates for the automatic annotation of Section 3.4 where precision is more important than recall and where a few false annotations are acceptable. User feedback for the first 48 video suggestions (as done in the case study) would lead to an annotation of about 25% of all object instances in this task. On average this means that we would annotate about 12 instances per object query, which is almost the same result that we achieved in the case study of Section 5.2, compare Figure 11.

Figure 15. Interpolated recall-precision curve (left) and precision at n shots (right).



# Shots	Precision
5	0.6480
10	0.5840
15	0.5067
20	0.4540
30	0.3773
100	0.1708
200	0.0980
500	0.0444
1000	0.0245

5.4. Discussion

In the experiments section of this work, we wanted to demonstrate that the proposed framework facilitates the annotation and retrieval process of video portals significantly. Thus, we developed a freely available annotation prototype [102] and performed an initial case study on it. This case study shows that at least interactive object annotation can successfully be used to annotate a few objects in several scenes of a video in order to enable object-based and scene-based video retrieval. The annotation time that a user has to spend in this interactive process was measured with 1.5 min for short videos and with less than 8 min for entire feature films. We believe that many video portal users are willing to spend such an effort if their videos gain higher visibility as a revenue. Furthermore, we showed that the processing time and storage consumption of all processing steps is moderate and we depicted several opportunities for the framework's integration in video portals that allows analysis on the client as well as on the server.

The results of the TRECVID challenge support the framework's usability for video portals as a similar number of correct object instances were detected under the topmost shots although a much larger dataset was used. The challenge further shows that the proposed framework is capable of producing competitive video object recognition results. The achieved results are above the median of all 14 participating teams though they are not the best overall results. However, we do not intend to prove that the proposed framework would work better than all other systems because further recognition approaches (including the ones that performed best in the challenge) can easily be integrated in the framework. Moreover, example-based video object retrieval is not the focus of the proposed framework but only a part of it.

Another important contribution of this work is the automatic approach selection because, on the one hand, it is known that just one approach is not sufficient for the recognition of all objects. On the other hand, neither the video portals nor their users can be supposed to select an appropriate recognition approach for each object-of-interest. We have shown that the selected approaches significantly outperform a baseline bag-of-features approach for most examples and that different recognition approaches are selected for different object categories. For instance, face detection approaches are frequently selected for the annotation of persons. Furthermore, we want to point out that our evaluations (as well as the evaluations of related works) show that current recognition approaches cannot perform the annotation of any object like the human visual system. However, we believe that the proposed framework will still be useful, if not more effective, when computer vision techniques have improved further. In this context, automatic approach selection will be especially important to support novel recognition approaches that are tailor-made for specific object types.

Moreover, we made some additional observations in our experiments that might be interesting for video portals. Firstly, the objects-of-interest that are suitable for annotation are restricted by their minimum size and by the number of object instances. Only those objects that take at least one third of a keyframe were selected and it takes an exceptionally long time to annotate objects that are only shown in one video scene. Secondly, we found that it is difficult to annotate unknown videos in a way that the selected objects and their labels are not superficial. Thus, we recommend to annotate objects shortly before or after the video upload. Thirdly, we noted it as preferable to select keyframes that show the object-of-interest with the highest amount of variability (e.g., the same person with different clothing, haircuts, and from different viewpoints) instead of simply choosing the first correct suggestion if many

positive keyframes are suggested by the prototype. Note that the author of this work has performed all experiments and annotations in his own right. Further usability studies can be performed by interested video portals with the annotation prototype, see Section 5.1. This also applies to investigations on changes in the user's retrieval process caused by the existence of new object annotations, as it is out of the scope of this project to perform evaluations on existing video portals.

6. Conclusions and Future Work

Despite the fact that object recognition techniques have seen a significant progress during the last years, content-based video analysis is almost not present in current video portals. In this work, we gave a broad overview of the field and proposed an extensive framework to enable object-based text queries and video scene retrieval in various portals. The presented annotation process starts by the selection of a single keyframe in which the object-of-interest is shown. Positive and negative examples of this object are then selected by interactive annotation with the help of a fast keyframe suggestion. In a further step, an integrated analysis and evaluation takes place to select an appropriate recognition approach for each object automatically. Eventually, an object is automatically annotated in large-scale video collections and the generated metadata is stored in a compact, binary form. Video portals can either integrate the entire framework or selected parts of it in order to improve video retrieval on different levels. An advantage of the proposed framework is the systematic support of improved computer vision techniques that will be proposed in the future. This benefit is inherited from the configurable object recognition infrastructure CORI that can be extended with all kinds of visual features, training and recognition strategies. High processing performance is assured by the support of distributed computer architectures and an intelligent reuse of intermediate results during feature extraction.

As a proof-of-concept, we developed a simple annotation prototype and performed a case study on two video datasets. In this process, we demonstrated that the framework can be used to efficiently annotate objects in short videos and full-length feature films. The results of automatically selected recognition approaches significantly outperformed the baseline results and further improvements are expected by the planned integration of general and object specific recognition approaches. Furthermore, the presented framework achieved competitive results for example-based video object retrieval on the instance search dataset of TRECVID 2011. As future work, we plan to automatically detect objects as pre-processing step for the interactive object annotation. Moreover, we will improve the region detection approach by the use of image segmentation to capture object boarders more accurately. Finally, we intend to integrate and test the framework in existing video portals.

Acknowledgments

The author would like to thank Horst Eidenberger for his feedback and support. The research leading to this article has received funding from the Austrian FIT-IT project "IV-ART - Intelligent Video Annotation and Retrieval Techniques".

References

1. Cheng, X.; Lai, K.; Wang, D.; Liu, J. UGC video sharing: Measurement and analysis. *Intell. Multimed. Commmun. Tech. Appl.* **2010**, *280/2010*, 367–402.
2. YouTube. Available online: www.youtube.com (accessed on 21 February 2012).
3. YouKo. Available online: www.youku.com (accessed on 21 February 2012).
4. Vimeo. Available online: www.vimeo.com (accessed on 21 February 2012).
5. Hulu. Available online: www.hulu.com (accessed on 21 February 2012).
6. Metacafe. Available online: www.metacafe.com (accessed on 21 February 2012).
7. Alexia. Available online: http://www.alexa.com (accessed on 21 February 2012).
8. Snoek, C.; Worring, M. A State-of-the-Art Review on Multimodal Video Indexing. In *Proceedings of the Conference of the Advanced School for Computing and Imaging*, Heijen, The Netherlands, 30 May–1 June 2001; Volume 24.
9. Rowe, L.; Boreczky, J.; Eads, C. Indexes for user access to large video databases. *Storage Retr. Image Video Database II* **1994**, *2185*, 150–161.
10. Weber, J.; Lefevre, S.; Gancarski, P. Video Object Mining: Issues and Perspectives. In *Proceedings of the International Conference on Semantic Computing*, Pittsburgh, PA, USA, 22–24 September 2010; pp. 85–90.
11. Bolle, R.; Yeo, B.; Yeung, M. Video query: Research directions. *IBM J. Res. Dev.* **1998**, *42*, 233–252.
12. Netflix. Available online: www.netflix.com (accessed on 21 February 2012).
13. Yamamoto, D.; Masuda, T.; Ohira, S.; Nagao, K. Collaborative Video Scene Annotation Based on Tag Cloud. In *Proceedings of the Advances in Multimedia Information Processing (PCM '08)*, Tainan, Taiwan, December 2008; pp. 397–406.
14. Wakamiya, S.; Kitayama, D.; Sumiya, K. Scene extraction system for video clips using attached comment interval and pointing region. *Multimed. Tools Appl.* **2011**, *54*, 7–25.
15. Ulges, A.; Schulze, C.; Keysers, D.; Breuel, T. Identifying Relevant Frames in Weakly Labeled Videos for Training Concept Detectors. In *Proceedings of the International Conference on Content-Based Image and Video Retrieval*, Niagara Falls, Canada, 7–9 July 2008; pp. 9–16.
16. Davidson, J.; Liebald, B.; Liu, J.; Nandy, P.; Van Vleet, T.; Gargi, U.; Gupta, S.; He, Y.; Lambert, M.; Livingston, B. The YouTube Video Recommendation System. In *Proceedings of the Conference on Recommender Systems*, Barcelona, Spain, 26–30 September 2010; pp. 293–296.
17. Li, Z.; Gu, R.; Xie, G. Measuring and Enhancing the Social Connectivity of UGC Video Systems: A Case Study of YouKu. In *Proceedings of the 19th International Workshop on Quality of Service*, San Jose, CA, USA, 6–7 June 2011; pp. 1–9.
18. Cheng, X.; Dale, C.; Liu, J. Statistics and Social Network of Youtube Videos. In *Proceedings of the 16th International Workshop on Quality of Service (IWQoS '08)*, Enskede, The Netherlands, 2–4 June 2008; pp. 229–238.

19. Lai, K.; Wang, D. Towards Understanding the External Links of Video Sharing Sites: Measurement and Analysis. In *Proceedings of the 20th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Amsterdam, The Netherlands, 2–4 June 2010; pp. 69–74.
20. Hu, W.; Xie, N.; Li, L.; Zeng, X.; Maybank, S. A survey on visual content-based video indexing and retrieval. *Trans. Syst. Man Cybern. Part C Appl. Rev.* **2011**, *41*, 797–819.
21. Gupta, S.; Gupta, N.; Sahu, S. Object based video retrieval using SIFT. *Int. J. Electron. Commun. Comput. Eng.* **2011**, *1*, 1–4.
22. Zavřel, V.; Batko, M.; Zezula, P. Visual Video Retrieval System Using MPEG-7 Descriptors. In *Proceedings of the International Conference on SImilarity Search and Applications*, Istanbul, Turkey, 18–19 September 2010; pp. 125–126.
23. Sekura, A.; Toda, M. Video Retrieval System Using Handwriting Sketch. In *Proceedings of the Conference on Machine Vision Applications*, Yokohama, Japan, 20–22 May 2009.
24. Zha, Z.; Yang, L.; Mei, T.; Wang, M.; Wang, Z. Visual Query Suggestion. In *Proceedings of the ACM Multimedia*, Vancouver, Canada, 19–24 October 2009; pp. 15–24.
25. Datta, R.; Joshi, D.; Li, J.; Wang, J. Image retrieval: Ideas, influences, and trends of the New Age. *ACM Comput. Surv.* **2008**, *40*, 1–60.
26. Geisler, G.; Burns, S. Tagging video: Conventions and strategies of the YouTube community. In *Proceedings of the Conference on Digital Libraries*, Vancouver, Canada, 18–23 June 2007; pp. 480–480.
27. Ames, M.; Naaman, M. Why We Tag: Motivations for Annotation in Mobile and Online Media. In *Proceedings of the Conference on Human Factors in Computing Systems*, San Jose, CA, USA, 28 April–3 May 2007; pp. 971–980.
28. Marlow, C.; Naaman, M.; Boyd, D.; Davis, M. HT06, Tagging Paper, Taxonomy, Flickr, Academic Article, to Read. In *Proceedings of the Conference on Hypertext and Hypermedia*, Odense, Denmark, 22–25 August 2006; pp. 31–40.
29. Flickr. Availalble online: www.flickr.com (accessed on 21 February 2012).
30. Delicious. Availalble online: delicious.com (accessed on 21 February 2012).
31. Thaler, S.; Siorpaes, K.; Simperl, E.; Hofer, C. A Survey on Games for Knowledge Acquisition. Available online: http://www.sti-innsbruck.at/fileadmin/documents/technical_report/A-survey-on-games-for-knowledge-acquisition.pdf (accessed on 21 February 2012).
32. Eidenberger, H. The Common Methods of Audio Retrieval, Biosignal processing, Content-Based Image Retrieval, Face recognition, Genome Analysis, Music Genre Classification, Speech Recognition, Technical Stock Analysis, Text Retrieval and Video Surveillance. In *Fundamental Media Understanding*; atpress: Vienna, VA, USA, 2011.
33. Flickner, M.; Sawhney, H.; Niblack, W.; Ashley, J.; Huang, Q.; Dom, B.; Gorkani, M.; Hafner, J.; Lee, D.; Petkovic, D.; Steele, D.; Yanker, P. Query by image and video content: The QBIC system. *IEEE Comput.* **1995**, *28*, 23–32.
34. Dyana, A.; Das, S. MST-CSS (Multi-Spectro-Temporal Curvature Scale Space), a novel spatio-temporal representation for content-based video retrieval. *Trans. Circuits Syst. Video Technol.* **2010**, *20*, 1080–1094.

35. Li, Y.; Zhang, Y.; Lu, J.; Lim, R.; Wang, J. Video Analysis and Trajectory Based Video Annotation System. In *Proceedings of the 2010 Asia-Pacific Conference on Wearable Computing Systems*, Shenzhen, China, 17–18 April 2010; pp. 307–310.
36. Assfalg, J.; Bertini, M.; Colombo, C.; Bimbo, A. Semantic annotation of sports videos. *IEEE Multimed.* **2002**, *9*, 52–60.
37. Hauptmann, A.; Yan, R.; Lin, W.; Christel, M.; Wactlar, H. Can high-level concepts fill the semantic gap in video retrieval? A case study with broadcast news. *Trans. Multimed.* **2007**, *9*, 958–966.
38. Wyl, M.; Mohamed, H.; Bruno, E.; Marchand-Maillet, S. A Parallel Cross-Modal Search Engine over Large-Scale Multimedia Collections with Interactive Relevance Feedback. In *Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR '11)*, Trento, Italy, 17–20 April 2011.
39. Paredes, R.; Ulges, A.; Breuel, T. Fast Discriminative Linear Models for Scalable Video Tagging. In *Proceedings of the International Conference on Machine Learning and Applications*, Miami Beach, FL, USA, 13–15 December 2009; pp. 571–576.
40. Morsillo, N.; Mann, G.; Pal, C. YouTube Scale, Large Vocabulary Video Annotation. In *Video Search and Mining*; Springer: Heidelberg, Germany, 2010; pp. 357–386.
41. Brezeale, D.; Cook, D. Automatic video classification: A survey of the literature. *Trans. Syst. Man Cybern. Part C Appl. Rev.* **2008**, *38*, 416–430.
42. Wang, M.; Hua, X.; Tang, J.; Hong, R. Beyond distance measurement: Constructing neighborhood similarity for video annotation. *IEEE Trans. Multimed.* **2009**, *11*, 465–476.
43. Naphade, M.; Smith, J. On the Detection of Semantic Concepts at TRECVID. In *Proceedings of the 12th annual ACM international conference on Multimedia*, New York, NY, USA, 10–16 October 2004; pp. 660–667.
44. Lampert, C.; Nickisch, H.; Harmeling, S. Learning to Detect Unseen Object Classes by Between-Class Attribute Transfer. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '09)*, Miami, FL, USA, 20–26 June 2009; pp. 951–958.
45. Wang, X.; Zhang, L.; Jing, F.; Ma, W. Annosearch: Image auto-annotation by search. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, New York, NY, USA, 17–22 June 2006.
46. Ballan, L.; Bertini, M.; Del Bimbo, A.; Serra, G. Enriching and Localizing Semantic Tags in Internet Videos. In *Proceedings of the ACM Multimedia*, Scottsdale, AZ, USA, 28 November 2011.
47. Kennedy, L.; Chang, S.; Kozintsev, I. To Search or to Label? Predicting the Performance of Search-Based Automatic Image Classifiers. In *Proceedings of the International Workshop on Multimedia Information Retrieval*, Santa Barbara, CA, USA, 26–27 October 2006; pp. 249–258.
48. Ulges, A.; Schulze, C.; Koch, M.; Breuel, T. Learning automatic concept detectors from online video. *Comput. Vis. Image Underst.* **2010**, *114*, 429–438.

49. Siersdorfer, S.; San Pedro, J.; Sanderson, M. Automatic Video Tagging Using Content Redundancy. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, Boston, MA, USA, 19–23 July 2009; pp. 395–402.
50. Wang, M., G. Li; Zheng, Y.T.; Chua, T.S. ShotTagger: Tag Location for Internet Videos. In *Proceedings of the the 1st International Conference on Multimedia Retrieval (ICMR '11)*, Trento, Italy, 18–20 April 2011.
51. Halvey, M.; Vallet, D.; Hannah, D.; Jose, J. ViGOR: A Grouping Oriented Interface for Search and Retrieval. In Video Libraries. In *Proceedings of the Conference on Digital Libraries*, Austin, TX, USA, June 2009; pp. 87–96.
52. Tahara, Y.; Tago, A.; Nakagawa, H.; Ohsuga, A. NicoScene: Video scene search by keywords based on social annotation. *Active Media Technol.* **2010**, *6335/2010*, 461–474.
53. Wattamwar, S.; Mishra, S.; Ghosh, H. Multimedia Explorer: Content Based Multimedia Exploration. In *Proceedings of the TENCON Region 10 Conference*, Hyderabad, India, 19–21 November 2008; pp. 1–6.
54. Weber, J.; Lefevre, S.; Gancarski, P. Interactive Video Segmentation Based on Quasi-Flat Zones. In *Proceedings of the International Symposium on Image and Signal Processing and Analysis*, Dubrovnik, Croatia, 4–6 September 2011; pp. 265–270.
55. Sivic, J.; Zisserman, A. Efficient visual search for objects in videos. *Proc. IEEE* **2008**, *96*, 548–566.
56. Everingham, M.; Sivic, J.; Zisserman, A. Hello! My Name is... Buffy—Automatic Naming of Characters in TV Video. In *Proceedings of the British Machine Vision Conference*, Edinburgh, UK, 4–7 September 2006; Volume 2.
57. Alexe, B.; Deselaers, T.; Ferrari, V. What is An Object? In *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR '10)*, San Francisco, CA, USA, 13–18 June 2010.
58. Hu, Y.; Rajan, D.; Chia, L. Attention-from-motion: A factorization approach for detecting attention objects in motion. *Comput. Vis. Image Underst.* **2009**, *113*, 319–331.
59. Liu, D.; Chen, T. Video retrieval based on object discovery. *Comput. Vis. Image Underst.* **2009**, *113*, 397–404.
60. Suna, S.; Wanga, Y.; Hunga, Y.; Changb, C.; Chenb, K.; Chenga, S.; Wanga, H.; Liaoa, H. Automatic Annotation of Web Videos. In *Proceedings of the International Conference on Multimedia and Expo (ICME '11)*, Barcelona, Spain, 11–15 July 2011.
61. Jain, M.; Jawahar, C. Characteristic Pattern Discovery in Videos. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Chennai, India, 12–15 December 2010; pp. 306–313.
62. Quack, T.; Ferrari, V.; Van Gool, L. Video Mining with Frequent Itemset Configurations. In *Proceedings of the International Conference on Image and Video Retrieval*, Tempe, AZ, USA, 13–15 July 2006; pp. 360–369.

63. Over, P.; Awad, G.; Fiscus, J.; Antonishek, B.; Michel, M.; Smeaton, A.; Kraaij, W.; Quénot, G. TRECVID 2010–An overview of the goals, tasks, data, evaluation mechanisms, and metrics. Available online: http://www-nlpir.nist.gov/projects/tvpubs/tv10.papers/tv10overview.pdf (accessed on 21 February 2012).
64. Volkmer, T.; Smith, J.; Natsev, A. A Web-Based System for Collaborative Annotation of Large Image and Video Collections: An Evaluation and User Study. In *Proceedings of the ACM Multimedia*, Singapore, Singapore, 6–11 November 2005; pp. 892–901.
65. Larson, M.; Soleymani, M.; Serdyukov, P.; Rudinac, S.; Wartena, C.; Murdock, V.; Friedland, G.; Ordelman, R.; Jones, G.J.F. Automatic Tagging and Geotagging in Video Collections And Communities. In *Proceedings of the International Conference on Multimedia Retrieval*, Trento, Italy, April 2011; Volume 51, pp. 1–8.
66. Wu, X.; Zhao, W.L.; Ngo, C.W. Towards Google Challenge: Combining Contextual and Social Information for Web Video Categorization. In *Proceedings of the ACM Multimedia*, Vancouver, Canada, 19–24 October 2009.
67. Lowe, D. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110.
68. Mikolajczyk, K.; Schmid, C. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1615–1630.
69. Everingham, M.; Van Gool, L.; Williams, C.; Winn, J.; Zisserman, A. The pascal Visual Object Classes (VOC) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338.
70. Tuytelaars, T.; Mikolajczyk, K. Local invariant feature detectors: A survey. *Found. Trends Comput. Graph. Vis.* **2008**, *3*, 177–280.
71. Hsu, C.; Chang, C.; Lin, D. A Practical Guide to Support Vector Classification. Available online: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf (accessed on 21 February 2012).
72. Varma, M.; Ray, D. Learning the Discriminative Power-Invariance Trade-Off. In *Proceedings of the 8th IEEE International Conference on Computer Vision (ICCV '07)*, Rio de Janeiro, Brazil, 14–20 October 2007.
73. Jiang, Y.; Ngo, C.; Yang, J. Towards Optimal Bag-of-Features for Object Categorization and Semantic Video Retrieval. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, Amsterdam, The Netherlands, 9–11 July 2007.
74. Mikolajczyk, K.; Tuytelaars, T.; Schmid, C.; Zisserman, A.; Matas, J.; Schaffalitzky, F.; Kadir, T.; Van Gool, L. A Comparison of Affine Region Detectors. In *Proceedings of the 10th IEEE International Conference on Computer Vision (IJCV '05)*, Beijing, China, 17–20 October 2005; pp. 43–72.
75. Carreira, J.; Sminchisescu, C. Constrained Parametric Min-Cuts for Automatic Object Segmentation. In *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR '10)*, San Francisco, CA, USA, 13–18 June 2010.
76. Van de Sande, K.E.A.; Gevers, T.; Snoek, C.G.M. Evaluating color descriptors for object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1582–1596.
77. Oliva, A.; Torralba, A. Building the gist of a scene: The role of global image features in recognition. *Prog. Brain Res.* **2006**, *155*, 23–36.

78. Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognitio (CVPR '05)*, San Diego, CA, USA, 20–26 June 2005; Volume 1, pp. 886–893.
79. Lazebnik, S.; Schmid, C.; Ponce, J. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, New York, NY, USA, 17–22 June 2006.
80. Mairal, J.; Bach, F.; Ponce, J.; Sapiro, G. Online learning for matrix factorization and sparse coding. *J. Mach. Learn. Res.* **2010**, *11*, 19–60.
81. Babenko, B.; Dollar, P.; Belongie, S. Task Specific Local Region Matching. In *Proceedings of the IEEE 11th International Conference on Computer Vision (ICCV '07)*, Rio de Janeiro, Brazil, 14–20, October 2007.
82. Stavens, D.; Thrun, S. Learning of Invariant Features using Video. In *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR '10)*, San Francisco, CA, USA, 13–18 June 2010.
83. Winder, S.; Hua, G.; M.B. Picking the Best DAISY. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '09)*, Miami, FL, USA, 20–26 June 2009.
84. Winder, S.; Brown, M. Learning Local Image Descriptors. In *Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07)*, Minneapolis, MI, USA, 18–23 June 2007.
85. Jahrer, M.; Grabner, M.; Bischof, H. Learned Local Descriptors for Recognition and Matching. In *Proceedings of the Computer Vision Winter Workshop*, Ljubljana, Slovenia, February 2008.
86. Torralba, A.; Russell, B.; Yeun, J. LabelMe: Online image annotation and applications. *Proc. IEEE* **2010**, *98*, 1467–1484.
87. Doermann, D.; Mihalcik, D. Tools and Techniques for Video Performance Evaluation. In *Proceedings of the 20th International Conference on Pattern Recognition*, Istanbul, Turkey, 23–26 August 2000; Volume 4, pp. 167–170.
88. Muja, M.; Rusu, R.; Bradski, G.; Lowe, D. REIN—A Fast, Robust, Scalable REcognition INfrastructure. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, 9–13 May 2011.
89. Forsyth, D.; Malik, J.; Fleck, M.; Greenspan, H.; Leung, T.; Belongie, S.; Carson, C.; Bregler, C. Finding Pictures of Objects in Large Collections of Images. In *Proceedings of the Object Representation in Computer Vision*, New York, NY, USA, April 1996.
90. Wong, G.; Frei, H. Object Recognition: The Utopian Method is Dead; the Time for Combining Simple Methods Has Come. In *Proceedings of the International Conference on Pattern Recognition*, The Hague, The Netherlands, August 30–September 3 1992.
91. Bradski, G.; Kaehler, A. Learning OpenCV: Computer vision with the OpenCV library. Available online: http://www.vision.ee.ethz.ch/~tquack/quack_fimi_videomining.pdf (accessed on 21 February 2012).

92. Vedaldi, A.; Fulkerson, B. VLfeat: An Open and Portable Library of Computer Vision Algorithms. In *Proceedings of the ACM Multimedia*, Florence, Italy, 25–29 October 2008.
93. Matlab: Computer Vision System Toolbox. 2012. Available online: www.mathworks.com/products/computer-vision (accessed on 15 February 2012).
94. Oerlemans, A.; Lew, M. RetreivalLab—A Programming Tool for Content-Based Retrieval. In *Proceedings of the International Conference on Multimedia Retrieval*, Trento, Italy, 17–20 April 2011.
95. Lienhart, R. Reliable transition detection in videos: A survey and practitioner's guide. *Int. J. Image Graph.* **2001**, *1*, 469–486.
96. Frigo, M.; Johnson, S. The design and implementation of FFTW3. *Proc. IEEE* **2005**, *93*, 216–231.
97. Manjunath, B.; Ohm, J.R.; Vasudevan, V.; Yamada, A. Color and texture descriptors. *Trans. Circuits Syst. Video Technol.* **2001**, *11*, 703–715.
98. Viola, P.; Jones, M. Robust Real-Time Face Detection. In *Proceedings of the 8th International Conference on Computer Vision (ICCV '01)*, Vancouver, Canada, 7–14 July 2001.
99. Liu, H.; Song, D.; Ruger, S.; Hu, R.; Uren, V. Comparing Dissimilarity Measures for Content-Based Image Retrieval. In *Proceedings of the 4th Aisa Information Retrieval Symposium (AIRS '08)*, Harbin, China, 15–18 January 2008; pp. 44–50.
100. Sorschag, R.; Morzinger, R.; Thallinger, G. Automatic Region of Interest Detection in Tagged Images. In *Proceedings of the International Conference on Multimedia and Expo (ICME '09)*, New York, NY, USA, 28 June–3 July 2009; pp. 1612–1615.
101. Sorschag, R. CORI: A Configurable Object Recognition Infrastructure. In *Proceedings of the International Conference on Signal and Image Processing Applications*, Kuala Lumpur, Malaysia, 16–18 November 2011.
102. Video Annotation Prototype. Available online: http://www.ims.tuwien.ac.at/sor/VAP.zip (accessed on 21 February 2012).
103. Shotdetect. Available online: http://shotdetect.nonutc.fr/ (accessed on 21 February 2012).
104. FFmpeg. Available online: http://ffmpeg.org (accessed on 21 February 2012).