Cineast: A Multi-Feature Sketch-Based Video Retrieval Engine

Luca Rossetto Ivan Giangreco Heiko Schuldt
Department of Mathematics and Computer Science
University of Basel, Switzerland
{*firstname.lastname*}*@unibas.ch*

***Abstract*—Despite the tremendous importance and availability of large video collections, support for video retrieval is still rather limited and is mostly tailored to very concrete use cases and collections. In image retrieval, for instance, standard keyword search on the basis of manual annotations and content-based image retrieval, based on the similarity to query image(s), are well established search paradigms, both in academic prototypes and in commercial search engines. Recently, with the proliferation of sketch-enabled devices, also sketch-based retrieval has received considerable attention. The latter two approaches are based on intrinsic image features and rely on the representation of the objects of a collection in the feature space. In this paper, we present *Cineast*, a multi-feature sketch-based video retrieval engine. The main objective of Cineast is to enable a smooth transition from content-based image retrieval to content-based video retrieval and to support powerful search paradigms in large video collections on the basis of user-provided sketches as query input. Cineast is capable of retrieving video sequences based on edge or color sketches as query input and even supports one or multiple exemplary video sequences as query input. Moreover, Cineast also supports a novel approach to sketch-based motion queries by allowing a user to specify the motion of objects within a video sequence by means of (partial) flow fields, also specified via sketches. Using an emergent combination of multiple different features, Cineast is able to universally retrieve video (sequences) without the need for prior knowledge or semantic understanding. The evaluation with a general purpose video collection has shown the effectiveness and the efficiency of the Cineast approach.**

***Keywords*-Video Retrieval, Content-based Information Retrieval, Motion-based Video Retrieval.**

I. Introduction

From all commonly used types of media, video offers the largest variety and degree of expressiveness. This is reflected in the growth of video collections (both in terms of numbers and sheer collection sizes), the number of users, and the diversity of use cases in which video is becoming more and more important. However, despite of this development, support for retrieving videos and/or video sequences from large collections is still largely underdeveloped. Finding a specific video in this data deluge usually relies on manual annotations. However, it is not realistic to assume that every video creator is willing to or even capable of adding the required annotations during the upload process since this would be a quite time-consuming and tedious task, and since all possible future usages and searches would have to be anticipated already at upload time – which is practically infeasible. Automating the annotation process would, in turn, require a program which is able to understand the meaning of an arbitrary video sequence. While this can be achieved to some extent in narrowly defined use cases or concretely specified semantics (e.g., [1]), it is something that cannot be applied to general purpose collections. In the context of image retrieval, content-based searches that use one or several images as query object(s) have been established, in addition to standard keyword search, both in academic prototypes and in commercial search engines. This approach is known as query by example (QbE). Essentially, it completely abstracts from the images' semantics and considers low level visual features. Recently, with the proliferation of sketch-enabled devices, also sketch-based retrieval (a.k.a. query by sketch, QbS) has received considerable attention, especially for known item search (i.e., the user has already seen the object(s) s/he is looking for and is able to sketch the most important characteristics). Both QbE and QbS are based on intrinsic image features and rely on the representation of the objects of a collection in the feature space.

In this paper, we present *Cineast*, a multi-feature sketch-based video retrieval engine. Cineast allows to retrieve videos solely on the basis of their content without the need for annotations and/or prior knowledge. Hence, the main objective of Cineast is to enable a smooth transition from content-based image retrieval to content-based video retrieval. It aims at supporting powerful search paradigms in large video collections using hand-drawn edge or color sketches or sample video sequences as query input. For this, Cineast relies on a large variety of different intrinsic features, extracted from the collection as a whole, from video sequences, and from individual video frames. Moreover, Cineast also supports a novel approach to sketch-based motion queries by allowing a user to specify the motion of objects within a video sequence by means of graphical gestures, also specified via sketches. Using an holistic combination of all these different features, Cineast is able to retrieve general purpose video, mainly in a known item search context, without the need for prior knowledge or semantic understanding.

For this, Cineast follows a general purpose approach and exploits features derived from the raw video information such as pixels within frames, samples within audio tracks,

and text from the subtitles as they are encoded, rather than relying on the semantic meaning instilled within a video as interpreted by a human being. Hence, the Cineast approach to deal with the *semantic gap* (i.e., the fact that objects can be represented in different ways and different media types even in the same context) neither attempts to bridge this gap, nor is it limited to only one representation of an object. Because video contains information in multiple channels of information with different modalities (visual, auditory, textual) which are semantically linked by their simultaneity, it is possible to cover a larger variety of possible input without having to translate between representations. This enables the use of different types of input within the same query by simply separating the input along its modalities, performing individual searches per channel and subsequently combining the results. This way, Cineast does not require any semantic interpretation or transformation in order to be present on all sides of the semantic gap.

For the evaluation of Cineast, we have compiled a general purpose video collection. On the basis of this collection, we have performed both qualitative user studies to verify whether the system actually meets the needs of users, and we have run quantitative experiments to assess the system's run-time performance and scalability. The results show that Cineast is able to provide responsive search times both in QbE and QbS settings and good retrieval quality.

The remainder of this paper is structured as follows: Section II discusses related work. In Section III, we introduce the Cineast approach and we present implementation details in Section IV. Section V summarizes the evaluation of Cineast and Section VI concludes.

II. Related Work

Research in the area of content-based multimedia retrieval has a long tradition. Early prototypes for enabling retrieval in videos include the QBIC system [2] and JACOB [3]. Retrieval systems from this early phase often use low-level features, e.g., color, texture, shape, etc. to query collections of video data. In more recent years, the focus in multimedia retrieval has shifted towards semantic video analysis and object recognition techniques using a Bag-of-Feature appraoch (e.g., [4], [5]). These include the use of support vector machines for event classification [6], or hidden markov models for audio-visual features [7]. These methods work reasonably well for very specific retrieval tasks, narrow ranges of content and/or appropriate training corpora, but they heavily narrow the use case they can be applied to. For more complex or more generic use cases, various research projects have turned to human-computation for the feature extraction, i.e., to using the wisdom of the crowd for event detection [8], [9].

Despite the fact that the more simplistic low-level features have been largely disregarded in recent years, the undoubted benefit of using simple features, apart from the lower computational effort at extraction time –and often also at query time–, is their large applicability, i.e., ranging from photographic stills to hand-drawn sketches (as in [10]), while at the same time neglecting any specific domain knowledge. To still be able to ensure qualitatively good results, various approaches have been used in recent years to combine multiple "weak" features to the end of a strong result. For instance, in [11] the authors use multiple features to detect video copies. Similarly, in [12], Multiple Feature Hashing (MFH) is used. MFH is a technique that applies learning to map multiple features to a single bit string with the objective of detecting near-duplicate videos both efficiently and effectively. Quick-Combine, an algorithm for combining multi-feature result lists from text and average color features of an image database, is proposed in [13]. In [14] multiple-features are used in a content-based image retrieval system for querying by example. The authors in [15] apply various visual and auditory features for content-based video retrieval.

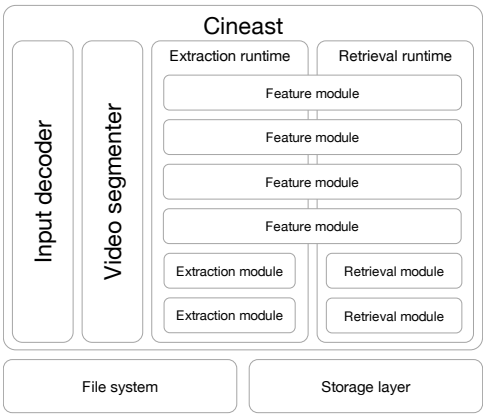

Figure 1. System Model of Cineast

III. Cineast

A. System Model

Cineast can conceptionally be divided into an *on-line* and an *off-line* part. The time-sensitive on-line part is concerned with the actual retrieval, while the off-line part handles feature extraction and indexing of the video data. Both parts are very similar from an architectural point of view as they primarily build on feature modules. The system model of Cineast is summarized in Figure 1.

In Cineast, a *feature module* can be viewed as an independent unit which deals with a specific feature representation, the extraction of the corresponding features, and their use for retrieving objects. In most cases, the modules use the same functionality for performing the feature extraction and the retrieval task. However, this is not an intrinsic requirement imposed by the system but rather a consequence of the structural similarity of feature representations.

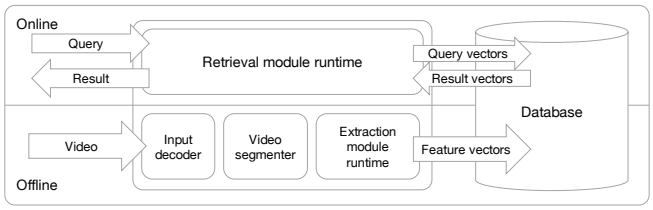

Figure 2. Workflows in Cineast

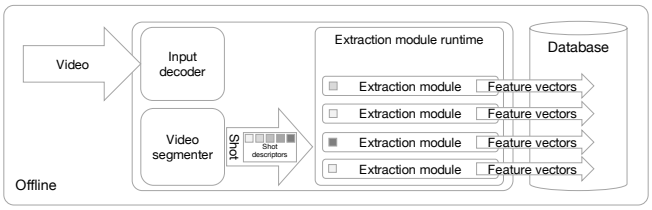

Figure 3. Off-line workflow

The on-line part of Cineast primarily consists of a *module execution run-time* which is in charge of managing the individual feature modules. Cineast's architecture allows the modules to work independently of each other without any interleave at retrieval time and thus supports multi-feature queries, i.e., queries that jointly consider several different features. The module execution runtime manages the feature modules, by providing functionality for the initialization and termination of the modules, by providing the modules with the input information they need, and finally also taking over and consolidating their outputs. Essentially, using a feature for retrieval means computing the representation of a query object in the feature space and searching for the nearest neighbours of this query object in the feature space among all objects in the collection.

For the off-line part of Cineast, units for the *decoding* and the *segmentation* of the video are provided. Input decoding logic provides the video segmenter with a continuous data stream which it segments into appropriate chunks. These chunks are then passed to the *feature extraction modules* which perform the extraction of the corresponding features. These are then handed-off to the storage layer.

B. Workflows

In what follows, we detail the on-line and the off-line workflows within Cineast which are depicted in Figure 2.

1) Off-line Workflow: The off-line workflow is used to process the videos and extract information and features necessary for retrieval. Figure 3 illustrates this workflow.

Video segmentation: The granularity at which Cineast performs the feature extraction and retrieval tasks is a *shot*. To perform the shot segmentation, Cineast uses a modified version of the fuzzy color histogram [16]. This method uses a fuzzy color segmentation to produce a histogram with 15 bins representing distinct colors. In Cineast, to improve the detection rate, the image is subdivided into 3×3 subimages for the histogram calculation. The histograms of the subimages are then concatenated to obtain a representation of the entire image. This is done for all frames. If the Euclidean distance between two consecutive normalized histogram representations exceeds a certain threshold, a shot boundary is detected. To improve the segmentation speed, the histogram representation in Cineast is calculated on a down-scaled version of the frame with a maximum of 200 pixels in every dimension. For practical reasons, we limit the length of a shot to 720 frames i.e., 30 seconds at 24 fps.

Shot descriptors: The shot descriptors (D) are an intermediate layer between the video and the feature extraction modules.[1] They introduce an abstraction of a certain aspect of a shot, e.g., pixel-wise average or median color of all frames within the shot, the sequence of frames of the shot, tracked motion paths and the like. A single shot descriptor does not contain all information of a shot but a compact representation of a subset of this information that can more easily be processed by the Cineast feature modules.

The following list provides an overview of the available descriptors.

- Metadata: unique ids to identify shot and video; start and end positions and length of the shot
- Frames: set of frames contained within the shot
- Average image: pixel-wise average per channel in RGB
- Median image: pixel-wise median per channel in RGB
- Most representative frame: frame with the smallest pixel-wise distance from the average image
- Motion paths
- Subtitle items: subtitle text which would be displayed during the shot

Feature extraction: The feature extraction is the primary part of the off-line computation process. In this process, feature extraction modules transform one or multiple shot descriptors (D) into a compact numerical vector representation (T) which is then handed off to the storage layer for persistent storage. Each module produces one or more tuples per shot, with a tuple containing multiple different vector representations. Tuples are associated to their corresponding shot using a unique shot id (I).

$$\mathcal{F} : \{I, D_1, \ldots, D_n\} \longrightarrow \{I, V_1, \ldots, V_m\} \quad | \quad V \in \mathbb{R}^d$$

2) On-line workflow: In the on-line workflow, illustrated in Figure 4, the actual retrieval is performed. The retrieval process can be broken down into distinct logical stages. At retrieval time, the module execution run-time receives a query object which it passes on to all retrieval modules. The retrieval modules use the storage layer to retrieve the most

[1]Note the difference to the definition of "descriptor" as e.g., used in the terminology of the MPEG-7 standard. While a MPEG-7 descriptor is a feature representation, we consider a shot descriptor only as an input for a following feature extraction step.

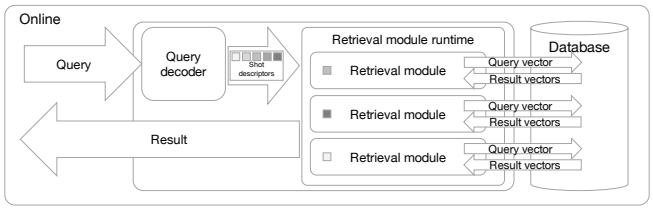

Figure 4. On-line workflow

similar elements in a similarity search. Note that the actual retrieval strategy is determined by the module; while some may perform a k-nearest neighbor search as part of a vector space retrieval, other modules may choose a different strategy. The result lists obtained from the similarity retrieval is returned by each feature module consist of tuples $\langle \text{id}, \text{score} \rangle$. Finally, the result lists are consolidated to a single, coherent result list.

Normalization and scoring: The feature modules within Cineast return a similarity score between 0 and 1. However, the similarity in vector space retrieval is often determined using a distance function $D : T \times T \to \mathbb{R}_0^+$ (in Cineast, this is mostly the squared Euclidean distance) between two feature representations (T) to obtain a distance d. Thus, we apply a correspondence function $K : \mathbb{R}_0^+ \to [0, 1]$ to transform the distance into a similarity measure with $K(0) = 1$ and $K(\infty) = 0$. The similarity score s is then calculated as $s = \kappa\left(1 - \frac{d}{d_{max}}\right)$ where κ denotes a clamping function necessary to avoid values outside the range $[0, 1]$. The value d_{max} is determined empirically.

Result consolidation: To combine the results of the different retrieval modules, we use a score-based late fusion approach. This allows all modules to work in parallel and independently of each other because no module requires data generated by a different module to perform its retrieval. The final similarity score S is a linear combination of the scores s_m given by every single module multiplied by an intra-category module weight w_m and an inter-category weight w_{c_m} for each category. The categories subsume various feature modules with similar foci, e.g., color, edge, etc. It is evident that an element appearing in only few result lists is prone to receive a rather low score S, as it does not receive any score from the modules that do not yield the element.

$$S = \sum_{m \in M} w_{c_m} \cdot w_m \cdot s_m \quad \text{with} \quad \sum_{c \in C} w_{c_m} \sum_{m \in M} w_m = 1$$

3) Retrieval modes: Cineast supports three different modes of retrieval. The first one is a direct user input mode in which the query object given to Cineast is a sketch (*query-by-sketch*). The second mode offers a *query-by-example* functionality. This approach can be used in cases where a user wants to find similar sequences to a previously retrieved one. The third mode, *motion-based retrieval* is a novel approach to search in videos by specifying the motion of objects across consecutive frames. For this, users are able to specify partial flow fields via sketches. This mode can be seamlessly combined with query-by-sketch. Finally, and complementary to the three main retrieval modes, Cineast supports *relevance feedback*. It enables the user to specify multiple relevant as well as non-relevant results from a previous search to refine a query.

4) Features: The following list provides on overview of the variety of features currently used in Cineast.

- Global features
  - *Average / Median color*: aggregates over all pixels in all frames within a shot to produce a single color
  - *Dominant shot colors*: centre-points of the 3 largest color clusters of all pixels in all frames in a shot
  - *Chroma / Saturation*: average of all chroma / saturation values of a shot
  - *Color histogram*: relative occurrence of all colors of a shot quantized using a fuzzy color histogram
  - *Shot position*: relative position of a shot with respect to the entire video
- Regional color features
  - *Color moments*: channel-wise calculation of statistical moments over regional partitions (uniform grid, angular radial partitioning [17])
  - *Registered color grid*: grid of fuzzy quantized colors. At retrieval, a registration of the query-grid on to the target grid is performed
  - *Color layout descriptor* [18]
  - *Color element grids*: grids containing partial color information in various representation (average saturation, variance of hue, etc.)
  - *Subdivided color histogram*: fuzzy color histograms of image partitions
- Regional edge features
  - *Partitioned edge image*: ration of edge- and non-edge-pixels per regional partition
  - *Edge histogram descriptor* [19]
  - *Dominant edge grid*: edge directions are quantized into categories ($0°$, $45°$, $90°$ and $135°$)
- Motion features
  - *Directional motion histograms*: regional normalised histograms of quantized motion
  - *Regional motion sums*: regional sums of the lengths of all motion vectors.
- Text features: common text retrieval methods are applied to all subtitle elements of a shot.

IV. Implementation

Cineast is written in Java and provides a retrieval API for external applications. The current user interface is browser-based and a PHP backend mediates the communication between Browser and Cineast.

Since the feature extraction modules work independently of each other, they can be executed in parallel. Because the number of modules usually surpasses the number of available CPU cores, a fixed-size thread pool is used which executes a subset of the modules simultaneously. All modules operate in a single threaded manner to keep the thread-count constant. Round robin is used for module selection. The module order is randomized in the beginning to reduce the probability that multiple modules simultaneously request the same shot descriptors. Because the computation of a shot descriptor is deferred and blocking, it is beneficial to use different descriptors simultaneously to avoid a situation where multiple threads wait for the same computation to finish. Since blocking is unavoidable, this approach reduces the expected system runtime.

Cineast uses ADAM [20] as storage layer, an extension to PostgreSQL which provides nearest-neighbour search in high dimensional vector spaces together with Boolean retrieval. As every feature module has its own connection to ADAM, the primary bottleneck in our current implementation lies in the database as it can only handle a limited number of connections simultaneously.

V. Evaluation

A. Setup

To evaluate the Cineast system, a test set of 50 queries (25 screenshots, 25 hand-drawn sketches) was constructed. For each query, rank, score and average search time after 10 executions was recorded. The first set of settings *a* was chosen to guarantee a recall of 100% for the test set while the second *b* uses a more reasonable real-world configuration. The dataset used for the evaluation contains 200 videos from various genres with a runtime of nearly 20 hours.

The storage layer of Cineast runs inside an Ubuntu 12.04 virtual machine virtualised using Oracle VirtualBox 4.3 (4 vCores, 4096MB of RAM) while Cineast runs on the host system (Intel Core i7-4850HQ CPU @ 2.30GHz, 16384 MB RAM). In both settings we used 4 threads for the execution of the query. While both settings used all 38 feature modules, in setting *a*, each module returned 250 results and the whole system returns 1000 results, whereas in setting *b* the modules were set to only return 100 results and the whole system 200 results.

B. Results

Table I summarizes the results of the evaluation. It can be seen that the queries containing screenshots produced results with lower rank and higher score than those with freehand sketches. It can also be seen that the retrieval times for configuration *b* are much lower than those of configuration *a* while the recall only drops rather insignificantly.

Figure 5 illustrates the distribution of rank and score of the two query sets. It shows most of the queries with screenshots have a rank close to 1 while the freehand set has a much larger spread. Furthermore, the freehand queries generally have a lower score, but the scores have a similar spread.

Table I
Result overview

		Screenshot		Freehand	
Evaluation setting		a	b	a	b
Score	min	0.298	0.250	0.083	0.105
	avg	0.698	0.672	0.305	0.250
	max	0.852	0.852	0.593	0.500
Rank	min	1	1	1	1
	avg	3.48	3.6	108.8	53.9
	max	36	56	466	175
Time (in ms)	min	2047	1002	2146	1180
	avg	2804.1	1415.8	2731.1	1428.3
	max	3905	2023	3359	1802
Recall		100%	100%	100%	92%

The results presented in Figure 5 show a near perfect retrieval behavior for queries containing a screenshot. The few cases with a score below 0.5 are probably targeted towards longer shots with a higher amount of visual change or shots with soft transitions. The queries containing freehand sketches show a larger range of ranks. This is due to the varying accuracy in the sketches caused by differences in artistic skills and accuracy of memory of the users.

The overall lower scores for setting *b* can be explained by the hard limit of results per module. The more results an individual module returns the higher is the probability that the correct result is within the result set. This is especially true when a feature is not very discriminant w.r.t. a particular query so that the scores of the returned results are close to each other and the limit becomes more or less arbitrary.

The time differences between the screenshot and the freehand test set seem at least partially to be an artefact of the logging system which stores every incoming query image using the PNG image format. Compressing and storing generates more effort for complex images like screenshots than for freehand sketches.

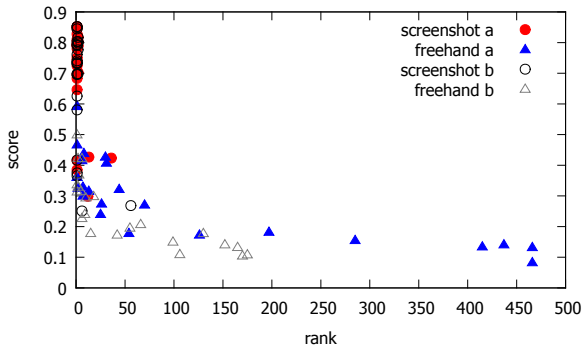

Figure 5. Rank and score of expected element per query

VI. Conclusion and Future Work

In this paper, we have presented Cineast, a content-based video retrieval engine. We were able to show that it is possible to achieve a good or at least a reasonably good retrieval quality (depending on thy type of query input) with responsive search times on a general purpose data set through the parallel use of multiple features. We are able to retrieve video sequences using queries containing screenshots or freehand sketches while not requiring any semantic content understanding.

In our future work, we plan to extend Cineast in two directions: First, in the current implementation, weights for feature modules are statically fixed beforehand (i.e., before the actual retrieval process starts and thus before the query objects are known). This, however, makes the system inflexible and unable to react to queries for which certain features offer poor selectivity. We plan to overcome this issue by automatically weighting potentially discriminant features on a per-query basis which should increase the overall result quality. Second, even tough the Cineast approach offers a high degree of possible distribution, the current system uses a single database instance. Future versions will make use of multiple databases simultaneously, by partitioning the collection or the features. This is supposed to lead to a better efficiency at retrieval time without diminishing the quality.

Acknowledgments

This work was partly supported by the Swiss National Science Foundation, project IMOTION (20CH21_151571).

References

[1] P. Over, G. Awad, J. G. Fiscus, B. Antonishek, M. Michel, W. Kraaij, A. F. Smeaton, and G. Quénot, "Trecvid 2010 - an overview of the goals, tasks, data, evaluation mechanisms and metrics," in *TRECVID 2010 workshop participants notebook papers*, Gaithersburg, MD, USA, Nov. 2010.

[2] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic *et al.*, "Query by image and video content: The QBIC system," *Computer*, vol. 28, no. 9, pp. 23–32, 1995.

[3] M. La Cascia and E. Ardizzone, "JACOB: Just a content-based query system for video databases," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 1996)*. Atlanta, USA: IEEE, 1996, pp. 1216–1219.

[4] Y.-G. Jiang, C.-W. Ngo, and J. Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," in *Proc. Int. Conf. on Image and Video Retrieval (CIVR 2007)*, Amsterdam, Netherlands.

[5] A. Bosch, A. Zisserman, and X. Muñoz, "Scene classification via pLSA," in *Proc. Europ. Conf. on Computer Vision (ECCV 2006)*, Graz, Austria, 2006, pp. 517–530.

[6] D. A. Sadlier and N. E. O'Connor, "Event detection in field sports video using audio-visual features and a support vector machine," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1225–1233, 2005.

[7] Y. L. Kang, J.-H. Lim, M. S. Kankanhalli, C.-S. Xu, and Q. Tian, "Goal detection in soccer video using audio/visual keywords," in *Proc. Int. Conf. on Image Processing (ICIP 2004)*, Singapore, 2004, pp. 1629–1632.

[8] F. Sulser, I. Giangreco, and H. Schuldt, "Crowd-based semantic event detection and video annotation for sports videos," in *Proc. Int. ACM W. on Crowdsourcing for Multimedia 2014 (CrowdMM 2014)*, Orlando, USA, 2014.

[9] L. von Ahn and L. Dabbish, "Labeling images with a computer game," in *Proc. Int. Conf. on Human Factors in Computing Systems (CHI 2004)*. Vienna, Austria: ACM, 2004, pp. 319–326.

[10] M. Springmann, D. Kopp, and H. Schuldt, "QbS: Searching for known images using user-drawn sketches," in *Proc. of the Int. Conf. on Multimedia Information Retrieval (MIR 2010)*. Philadelphia, USA: ACM, 2010, pp. 417–420.

[11] J. Law-To, O. Buisson, V. Gouet-Brunet, and N. Boujemaa, "Vicopt: A robust system for content-based video copy detection in large databases," *Multimedia systems*, vol. 15, no. 6, pp. 337–353, 2009.

[12] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong, "Multiple feature hashing for real-time large scale near-duplicate video retrieval," in *Proc. Int. Conf. on Multimedia (MM 2011)*. Scottsdale, USA: ACM, 2011, pp. 423–432.

[13] W.-T. Balke and W. Kießling, "Optimizing multi-feature queries for image databases," in *Proc. Int. Conf. on Very Large Databases (VLDB 2000)*, Cairo, Egypt, 2000.

[14] C. Liu and Z. Wei, "Multi-feature method: An integrated content based image retrieval system," in *Int. Symp. on Intelligence Information Processing and Trusted Computing (IPTC 2011)*. Wuhan, China: IEEE, 2011, pp. 43–46.

[15] M. Gabbouj, E. Guldogan, M. Partio, O. Guldogan, A. Iftikhar, S. Uhlmann, and S. Kiranyaz, "An extended framework structure in muvis for content-based multimedia indexing and retrieval," 2007.

[16] O. Küçüktunç, U. Güdükbay, and Ö. Ulusoy, "Fuzzy color histogram-based video segmentation," *Computer Vision and Image Understanding*, vol. 114, no. 1, pp. 125–134, 2010.

[17] A. Chalechale, A. Mertins, and G. Naghdy, "Edge image description using angular radial partitioning," *Vision, Image and Signal Processing*, vol. 151, no. 2, pp. 93–101, 2004.

[18] E. Kasutani and A. Yamada, "The MPEG-7 color layout descriptor: A compact image feature description for high-speed image/video segment retrieval," in *Proc. Int. Conf. on Image Processing (ICIP 2001)*, vol. 1, Thessaloniki, Greece, 2001, pp. 674–677.

[19] D. K. Park, Y. S. Jeon, and C. S. Won, "Efficient use of local edge histogram descriptor," in *Proc. of the ACM Workshops on Multimedia*, Los Angeles, USA, 2000, pp. 51–54.

[20] I. Giangreco, I. Al Kabary, and H. Schuldt, "ADAM - A Database and Information Retrieval System for Big Multimedia Collections," in *Proc. Int. Cong. on Big Data 2014 (BigData 2014)*, Anchorage, USA, 2014.