This is a preliminary version of an article published by
S. Wilk, S. Kopf, W. Effelsberg
Collaborative Annotation of Videos Relying on Weak Consistency
ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 2016

Collaborative Annotation of Videos Relying on Weak Consistency

Stefan Wilk[1], Stephan Kopf[2], Wolfgang Effelsberg[2]
[1] Distributed Multimedia Systems, TU Darmstadt, Darmstadt, Germany
stefan.wilk@cs.tu-darmstadt.de
[2] Department of Computer Science IV, University of Mannheim, Mannheim, Germany
kopf@informatik.uni-mannheim.de, effelsberg@informatik.uni-mannheim.de

***Abstract*—This work discusses a distributed interactive video system that supports video annotation using simultaneous hyperlinking by multiple users. The users mark and annotate objects within the video with links to other media such as text, images, websites, or other videos. Annotations are visualized on the client user interface as an overlay close to the objects. Our system is intuitive to use; for example, it contains automatic object-tracking functionality that correctly positions the annotations, even when the form or location of an object changes. Thus, our first contribution discusses the adaptive object-tracking algorithm used for this repositioning. It shows improved precision and reliability in comparison to non-adaptive algorithms. A second key issue is to keep the system responsive when the number of concurrent annotators increases. Thus, we rely on the concept of eventual consistency between different network entities. While this weak form of consistency allows temporary inconsistencies, it ensures that a consistent state can be reached. Thus, the second contribution is the design and evaluation of our distributed interactive video system, which relies on the weak consistency paradigm.**

Keywords: semantic image search, iconic images

I. Introduction

For users of Video Sharing Sites (VSSs) like YouTube, annotations - more commonly known as comments or likes - are an important mean for expressing opinions and feelings. Yet, most annotation types are linked to an entire clip, which is more suitable for a still image than a video. This motivated us to rethink video annotation by developing a fine-grained interaction with objects that appear in a video. Therefore, we leverage the idea of a hypervideo, which links visual objects in a video to related media, e.g. text or images.

Our use case for such a system is the co-editing of video in an eLearning scenario. The video is perceived as an extensible document, such as a Wikipedia page, and a large number of users can add links to visual objects. The annotations linked to these video objects are accessible for users as an overlay on the user interface. We have shown in [1] that such a system improved the learning outcomes when hyperlinked videos were used. Co-editing video annotations in the eLearning scenario resulted in a need for a highly *scalable* system, as we aimed for hundreds of concurrent users, as well as an *intuitive-to-use* system, in order to enable arbitrary users to contribute annotations to the video. Regarding the *intuitive* usage of the system, users can access annotations by interacting with the objects in the video by pointing at an object (e.g. a mouse click). The annotations are visualized as an overlay to the digital video, as so-called interactive hotspots, and are stored independent of the digital video. In order to visualize interactive hotspots in relation to the video object, our system requires an annotation repositioning according to the movement and deformations of objects over the course of the video playback. To reduce the manual effort, an automatic object tracking algorithm is integrated into servers of our system. When an object within a video is marked by a user for the first time, automatic tracking ensures that annotations linked to that object are positioned and scaled according to the position and deformations of the object over the course of a video. This positioning affects the x- and y-coordinates of the associated interactive hotspot over different video frames. This approach does not require any manual intervention of a user. Yet, at any point in time users may adjust tracking results. Thus, our first contribution describes the automatic and adaptive tracking of objects in a video. Since requirements for tracking can usually not be fulfilled by a single algorithm with high reliability, we propose an adaptive selection of object-tracking algorithms. The result of this contribution is that our system simplifies the complex task of annotating a moving object in a video into a simple image-tagging task. As a result, automatic object tracking reduces the time required for annotating and allows users to focus on embedding new content.

Besides intuitive usage, *scalability* in our system is a challenge due to massive interactions with video objects by a large number of concurrent users. The system also needs to remain responsive to user interactions. A single server would not be capable to process video object tracking upon the requests of hundreds or thousands of concurrent users. Furthermore, as the system allows the parallel editing of interactive videos, the synchronization of those edits may result in classical, strong consistency approaches, i.e., locking objects for an exclusive access. We decided to use cloud capabilities to adapt the number of servers executing object tracking and managing users in relation to the number of active users in the system. The server functionality is distributed to many server instances, and each server manages a subset of the users. Yet, despite its distributed design, the system reacts to user interactions as if a local application is used, and a user is able to edit any video object at any time. As both concurrently created annotations as well as object tracking

descriptors are represented using the MPEG-7 standard, large media files and huge processing capabilities are required. A strong form of consistency in combination with a massively distributed server setup, in combination with the need for a responsive system without data locks, is thus not possible with commonly available cloud instances or retail server hardware. Thus, our second innovation is a solution to the problem of consistency of the replicated data in a scalable manner. The data copies in our system, i.e., user annotations and video object tracking features, are changed very often - but not immediately synchronized. Yet, all copies in the system are guaranteed to converge to the same values after updating ceases, which leads to eventual consistency. In this work, while we demonstrate and evaluate system use by several hundred users, these concepts can be mapped to thousands of concurrent users.

II. Background and Related Work

A. Interactive Video

In today's VSSs, interaction with videos is mostly limited to annotating an entire video with a rating or a comment. Guimaraes et al. [2] proposed to extend this functionality by describing how to personalize comments on user-generated video. In another work of Guimaraes et al. [3], the idea of annotating videos from heterogeneous sources by captions is described. Thus, Guimaraes et al. describe creating dynamic annotations in videos and using them to link videos to each other. In contrast to their work, we explain how objects in a video can be made accessible, and how related media can be linked to those objects. Thus, annotations must adapt to object positions in different video frames. In addition, we investigate weak consistency to ensure scalability and the system responsiveness to users' interactions.

Cesar et al. [4] developed a video system that focuses on social interaction during media production. The authors designed, developed, and evaluated this media sharing system that allows video annotation and sending recommendations to users. The proposed work does not focus on improving usability by applying an automatic object tracking algorithm or investigating scalability issues. The *Raconteur* system [5] combines social media and chatting, thus allowing users to interact with each other. Videos can be integrated at any time during the chat. General requirements for systems that combine digital video with social interactions on mobile devices are described by Juhlin et al. [6]. All of these approaches show communication, e.g., via chat, and how such communication should be designed. These findings are used and extended in our system, as we allow annotating video objects with chat topics.

As a basis for integrating interactivity with video objects, we leverage the concept of hypervideo. A hypervideo is interpreted in *HyperCafe* [7] as an opportunity for users to decide which direction a storyline should take. It implements a fixed, authored setting in a coffee store and allows users to listen to different conversations. The concept of detail-on-demand (e.g., in the Hyper-Hitchcock system *Hyper-Hitchcock* system [8]) tries to reduce the negative effect of disorientation a user might experience. It restricts the amount of information by allowing only one link for a video at a time. Bulterman's annotation tool [9] describes pen-based video editing. His aim is to construct a testbed for exploring the possibilities of user-generated annotations. If the underlying object of an annotation is moving, users can manually embed an animation path. The SIVA Producer/Suite [10], [11] offers a rich set for producing interactive, non-linear, and hyperlinked videos. It focuses on the user interface design, and proposes opportunities for intuitively authoring videos.

The *Interactive Shared Educational Environment* [12] combines active reading with video and takes the first step towards collaborative video consumption, allowing users to interact with each other. *ToolClips* [13] implements and validates the concept of videos supporting the learning processes better than text and that they help users understand program functionalities much better. StoriSphere [14] is building a user-centric video-sharing platform, which allows one to combine professional broadcasts with user-generated video. StoriSphere includes automatic content analysis methods for shot detection as well as title and description generation.

These existing systems lack at least one of the following features, not allowing users to:

- annotate objects in a video and add their own ideas,
- reduce the manual effort of users by an automatic tracking of objects,
- scale to large user communities by implementing a weak consistency level for copies of the videos and their annotations.

To the best of our knowledge, this is the first successful attempt to combine those features in one system.

B. Automatic Object Tracking

A major advancement of our system is the integration of an adaptive automatic object tracking method. To understand design decisions for our object tracking algorithm, we follow the classification of object tracking steps by Yilmaz et al. and Li et al. [15], [16]. They classify steps in an object tracking algorithm into: object representation selection, feature selection, an initial object detection step, and tracking an object throughout all frames of a video sequence. Designers of object tracking algorithms have to decide on a representation of a trackable object in one of three ways: in a shape model (e.g., point clouds, primitive geometric shapes, skeletal models), the object's contour, or the appearance of an object within a frame, which includes templates or probability models of an object appearance [15]. From the object representation, features for the tracking process can be derived, such as colors, object's edges, optical flow, or texture [15], [16]. In our system, the *initial object marking* is performed manually by the user. The *tracking step* includes point-based, as well as kernel-based, tracking methods. Point-based tracking methods leverage specific information on pixel positions and movements between adjacent frames solely on pixel attributes. Kernel-

based tracking leverages the object shape and appearance, e.g., the rectangular shape of the template and a feature histogram.

Object tracking algorithms that facilitate manual template selection was proposed by Weng et al. [17]. Similar to our algorithm, they have to cope with uncertainty as to whether the template selected by a user includes a suitable object representation. It is an adaptive approach, feeding results from already analyzed frames into their feature model. Our approach leverages similar ideas - such as using color as a feature to track moving objects within one video shot - however, we use different features to compensate for massive deformations and illumination changes.

Similar to our goal of designing a reliable tracking approach for hyperlinked videos, Goldman et al. [18] designed a particle-tracking approach similar to that proposed by Sand et al. [19]. Particle tracking is designed to generate long-range tracking results like other feature tracking algorithms – and offer the possibility of being spatially dense, to be capable of tracking small objects. We investigate an adaptive selection of object features and object tracking algorithms, depending on the investigated video sequence and the selected video object. Features include a template's color features, optical flow features, or feature descriptors that leverage illumination differences. The resulting tracking algorithms use the MeanShift algorithm [20], template-based matching, and SURF features [21] combined with the Kanade-Lucas optical flow tracker [22]. These algorithms were chosen due to their ability to perform well in distributed environments with limited resources. The idea of *MeanShift* color-based clustering [20] is to identify areas in a frame with characteristic colors. It is a kernel-based tracking algorithm that leverages color features. Depending on the color distribution of the searched object, the algorithm sets hypothesized clusters in the frame. The cluster centers are iteratively shifted to the mean of the data in a cluster until no more changes are detected.

Our *template-based tracking* leverages both brute-force color comparisons and pixel intensity values in the annotated *object region* with the pixel values in the video frame. For tracking purposes, this is repeated for consecutive video frames. *Template-based matching* is a brute-force approach that compares color or intensity values between a template and a series of consecutive frames. *Speeded Up Robust Feature* (SURF) [21] is a fast method for detecting and extracting feature descriptors that are robust to scaling and rotation. Additionally, as described by Li et al. [16], it is robust against small occlusions, illumination changes, and shape deformations. Feature correspondences are defined as the nearest neighbors between the features of a template and an arbitrary frame. The *Kanade-Lucas tracker* [22] calculates pixel displacements of consecutive frames on the basis of motion vector fields, which compensates for feature loss.

C. Consistency

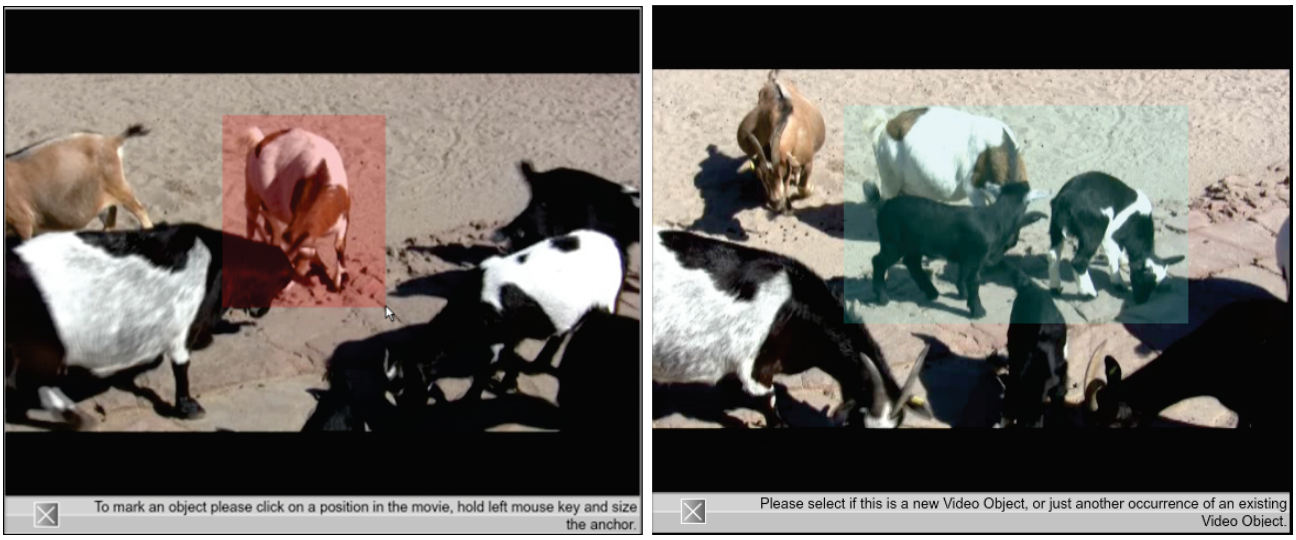
For large-scale distributed systems – such as our distributed, interactive video system – ensuring the CAP (Consistency, Availability, and Partition Tolerance) theorem may limit system performance. Thus, only two of the three desired properties can be achieved at the same time [23].

With *eventual consistency*, data is not necessarily saved consistently on any network entity at any given time [24]. It is a special form of weak consistency, which guarantees that all changes will be synchronized to the same value across all copies if no updates occur from that point forward [24]. Such an eventually consistent system allows inconsistencies at any point in time.

The idea of eventual consistency has evolved into various forms and levels of weak consistency. An overview on different consistency levels is given by Saito et al. [25]. We focus on *read-your-writes-consistency*. Without occurrence of an error, the network entity performing the change will never retrieve an outdated value from its own site, but the data is not necessarily shared instantaneously between the remaining network entities. Ballis et al. [26] describes that, in most situations, eventual consistent systems behave quite similar to strong consistent systems. Only massively parallel access to those systems reduces the consistency in practice. Different database systems relying on the eventual consistency paradigm have been introduced including Cassandra and DeeDS [27]. The downside of the system architecture and the scale of distribution, an inconsistency usually requires between 200 milliseconds (Cassandra) [28] and 13 seconds to be resolved (Amazon S3) [29].

The *DIMA* (Distributed Interactive Multimedia Applications) system proposes consistency analysis for distributed multimedia applications [30]. Bouajjani et al. [31] present a similar formal verification scheme for different consistency levels including weak forms. Thus, both Bouillot et al. and Bouajjani et al. developed a generic algorithmic verification method for different application schemes. A methodology for validating the temporal consistency of pre-designed, interactive multimedia flows was given by Mirbel et al. [32], who used a description language to illustrate inconsistencies. However, they do not offer a solution for implementing weak or strong consistency. The necessity of integrating management of weak consistency into programming languages is discussed by Burckhardt et al. [33]. They introduce new data types for cloud programming that enable eventual consistency-secure addressing of data items. In contrast to the work of Bouajjani et al. Burckhardt et al., Buillot et al., and Mirbel et al., we present an application of the eventual consistency paradigm to a collaborative video system. Thus, our application benefits from good practices that have been recommended regarding the programing eventual consistent applications

The aim of our system is to enable many concurrent users to collaboratively enrich videos with annotations that link to related media. In comparison to many previous hypervideo systems, the proposed system allows users to interact with every object visible in a video. For our system, we assume that consistency is continuously broken and eventually reestablished, as classical consistency paradigms cannot ensure the desired system response times.



(a) (b)

Fig. 1. Overview of the interactive video system's user interface. A user can create an annotation by drawing a rectangular, interactive hotspot on (a) one video object (in this case, a goat) and (b) multiple video objects (two goats) when the user wants to annotate a group of objects.

III. System Overview

A. Requirements and Design Overview

Our aim is to design a distributed, interactive video system that scales well with the number of concurrent users by leveraging the concept of eventual consistency. The resulting requirements were then tested in an eLearning scenario. Our system is used to annotate videos by linking objects to additional media that ease the viewer's understanding of concepts discussed in the video. Users can thus easily enrich the video, but they can also navigate using links in the video if they are specifically interested in understanding more about the discussed topics. This results in a constantly growing video document; a close parallel is an online encyclopedia such as Wikipedia, where each user can contribute to an existing knowledge base.

To achieve our goal, we address five requirements that our system should fulfill:

1. Interactive video: Users should be able to interact with distinct video objects and access available annotations, thus personalize their video consumption and learning experience. To show users which objects contain annotations, the concept of an interactive hotspot is used. An interactive hotspot is a visual overlay to an annotated video object. A transparent, rectangular area is visualized to the user (see Figure 1) which positions the area according to the video object's movements. Furthermore, it is sensitive to interactions, e.g., a mouse click.

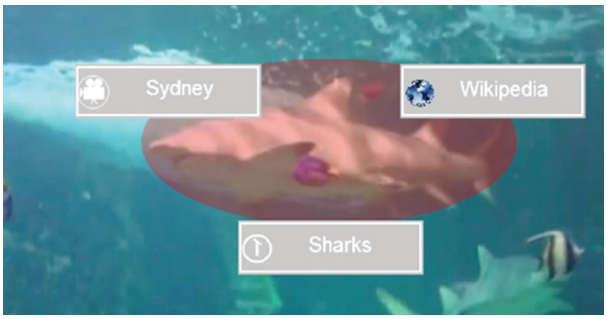
2. Collaborative annotation: Each user shall be able to easily annotate video objects with other media. Those annotations are propagated to all users to allow them to add their own media links as soon as possible. Due to the concept of eventual consistency, no precise deadline can be set when annotations are available for all users. Thus, no locks are set on the annotated video objects; these annotations can be modified by any user, at any time.

3. Automatic object tracking: An interactive hotspot, once created, is positioned and scaled according to the underlying video object. Automatic tracking compensates for the video object's movement, scaling, deformation, or occlusion. A measure of this requirement's success is the correct localization of objects in different video frames (see Section VI-B).

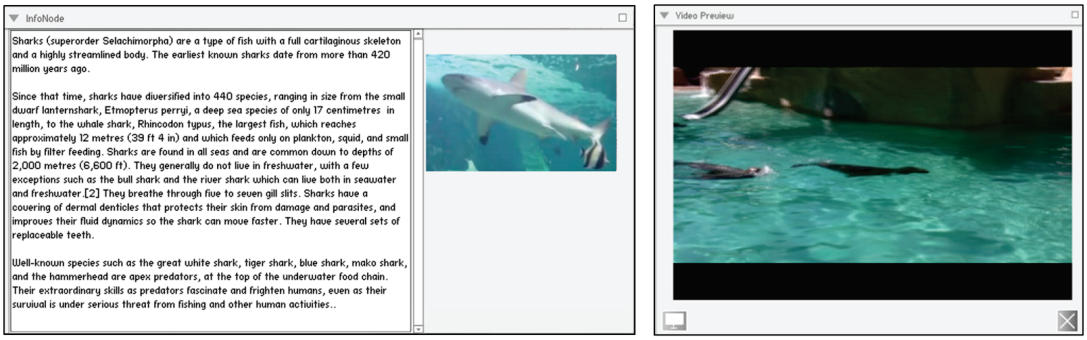
4. Scalability: The system supports sessions of a single user as well as many concurrent users. The concepts of this work shall be valid for thousands of concurrent users. For our prototypical evaluation in this work we aimed for two hundred users

5. Responsiveness: When a user interacts with the system's user interface, quick system feedback is desirable. Independent of the number of concurrent users, the system has to be responsive to each user interaction. Furthermore, the user should always be able to access any annotation in the system without any locks, even when they are currently edited by other users.

Requirements 1 and 2 are extensively discussed in our previous work [1], [34]. Requirements 3-5 include the high resource demands of an automatic object tracking requirement, the need for scalability, e.g., by using capabilities of a dynamically growing set of servers, and the necessity of having instant feedback for user interaction resulting in a distributed system design. In other words, we distribute the functionality over several network entities – namely clients, bootstrapping servers, and support servers. The *bootstrap server* offers to the user information for running the client side of the system, and assigns the user to a *support server*, where automatic object tracking is executed and sessions are managed. This



(a)



(b)

Fig. 2. In (a), additional information linked to the video object is shown. Two different information node types (b) are then shown: the hypertext and the video node.

session management includes the administration and conflict resolution of annotations for each of the assigned users. *Clients* implement the user interface and decode the video elements for presentation. In our system, co-editing video documents between clients requires that annotations are synchronized across network entities. Video co-creation or co-editing occurs when users of our eLearning platform want to enrich the video by annotations, e.g., if several users during a broadcast of a lecture were discussing topics in a chat or adding notes to video objects. The automatic object tracking approach then reduces the effort of the annotation process by allowing a user to mark an object in a video only once. The system will then take care of the video object's scaling or movement during the course of the video. No additional user interaction is needed to position an interactive hotspot according to the movements and scaling of a video object. Direct user interactions and an automatic object-tracking approach result in a large amount of data to be synchronized. A design choice arises whether to ensure strong consistency or a responsive system. We decided to relax the consistency level to ensure a responsive system (see Section V).

B. Interactive Video Client

The interactive video system is Web-based, implying that communication between the clients and servers is realized using the hypertext transfer protocol (HTTP). This Web-based design allows easy integration into *Facebook*, our prototyping and evaluation environment. The system relies on *annotating* objects and *accessing annotations* from objects in a video. In contrast to static text or images, the dynamics of a video and its sequence of images is a challenge for accessing or enriching information. It is a special technical challenge to annotate and access *video objects* because they appear, change shape or size, move, or disappear during the playback of the video.

Interacting with Video Objects

As explained above, our system illustrates to users which video objects are available for interaction using the concept of *interactive hotspots*.

The client software uses interactive hotspots to determine which pixels in a frame are associated with an annotated object. Video objects can be linked with additional media elements called information nodes. Information nodes can be *images*, *additional text*, links to *Web pages*, *other videos*, and *communication topics*. The interactive video client is able to decode and display videos and their annotations in parallel. Users can access those annotations at any time while watching a video.

Interaction with the video is thus possible in two unique ways: annotating related media to a video, as commonly used in hypertext documents, and accessing the annotated media objects (navigation). Navigation is possible by clicking on an interactive hotspot that represents the video object (see Figure 2), resulting in linked information nodes becoming visible. Information nodes offer an option to access a single

media item related to the annotated object in the video. An interaction with an interactive hotspot displays a radial menu (see Figure 2 - a). By navigating from one video to other media elements, the user can retrieve hypertext documents (see Figure 2 - b), communication nodes (including chatting functionality), and other videos. Users can thus jump between links to other media elements easily.

Annotating a Video

As a user *annotates* an object in a video, the system tries to directly show this annotation to all other users and let them interact with it. Annotations, as well as the related tracking information, are stored in the MPEG-7 format. All annotations for a video are stored in one so-called *interactive video document*. It is read by the client software to visualize annotations. Besides this, each video contains a second MPEG-7 document that contains object tracking features, which are only required by support servers. As explained in Section V, a synchronization approach is applied across different network entities to allow different users to co-edit videos.

A user can select a video object and add information to it in a two-step approach. Selecting a video object is done with a mouse by clicking on the object, which pauses the video playback. The object selection is then marked by drawing a rectangular area around the object (see Figure 1 - a). This initial annotation of the video invokes the automatic object tracking on a support server. In the next step, the user creates links to additional media (information nodes) to the video object. Information nodes include Web nodes, image-text (hypertext), video, and communication nodes. *Web nodes* represent links to external Web pages. If the user clicks on a Web node annotation, the system opens a new browser tab and shows the Web page in the Web browser. *Image-text nodes (hypertext)* link text blocks and images to the video object, which are then shown within the system. The basis for those nodes is a rich text editor, which is implemented into the system along with a hypertext editor. *Video nodes* refer to other videos that users can upload to our system. In the system's current version, videos have to be uploaded to our servers. Chats between users can be initiated using *communication nodes*, which users can link to video objects. Each communication node represents a distinct channel in the integrated chat system. Within a channel, users can exchange questions or comments.

Any user can edit any annotation at any time after a video object is selected. Thus, multiple users may edit the same information nodes or video objects at the same time. A detailed explanation of the user interface can be found in our previous work [34], and a video demo of the system is available online[1].

C. Representation of Annotations

For representing annotations in the interactive video document, the MPEG-7 standard is used [35]. Even though this standard is rather verbose, we leverage its advantages of available libraries, persistence support, and readability. We have chosen MPEG-7 due to its rich support for both representing annotations in continuous media, and as it allows to represent common computer vision feature descriptors used (as described in Section IV).

Interactive video documents use the <semantic> element, which allows for storing both the location and annotated information such as text, images, links, or other videos. As mentioned, the annotated video object itself is represented in our system as an interactive hotspot. The *Spatio-Temporal-Locator* of the *Visual Part* is used to map the interactive hotspots to MPEG-7 descriptors. With the *Spatio-Temporal-Locator*, the frame number and the location of an annotated object in a video can be specified. This *Spatio-Temporal-Locator* allows the client software to visualize the interactive hotspot in the user interface and reposition it over time.

Information nodes are linked to video objects. They contain an annotation of one specific media type, i.e., a hyperlink, a video, a chat topic, hypertext, or an image. For the information node, the <SemanticBase> element of MPEG-7 is used. One semantic element is allowed to consist of 0 to n elements. The type-attribute is used to distinguish the media which is included in the information node. The <FreeTextAnnotation> is used to represent annotations containing plain text or hypertext. Other media, such as video or images, rely on the <MediaLocator> element. Hyperlinks rely on the <MediaUri> element.

Data which the support servers need for an automatic video object tracking the so-called features are extracted from all frames after a new video is uploaded. They are persisted in order to allow real-time tracking, as an on-the-fly extraction of features would result in a significant runtime increase. Those features are not only persisted as MPEG-7 documents, but also stored independent of the interactive video documents that clients use. Most of the leveraged features are multidimensional and of float precision.

We leverage MPEG-7 in its XML representation, and can thus easily detect differences between copies of an interactive video document. When an existing video object is modified, the video object's MPEG-7 description is exchanged with the associated bootstrap server and later propagated to all other network entities. The description of a whole video object is exchanged in order to detect changes and potential conflicts. Potential conflicts occur as different users might change the same video object before the changes are synchronized between the network entities. A detailed discussion on the occurrence of conflicts follows in Section V.

D. Backend Functionality and Maintaining Consistency

To achieve a responsive system and to ensure scalability, the backend is split into a *bootstrap server* and several *support servers*.

The *bootstrap server* allows clients to access the interactive video system. One of its tasks is to redirect new clients to a designated support server. The bootstrap server manages those

[1]https://www.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_DMS/Social_Video_System.mp4

support servers, starting and stopping them, and assigning them to joining clients. Before a support server is paused, the bootstrap server is also responsible for retrieving all changes made to the support server to be stopped. When a support server is no longer needed (when no clients are connected to it anymore), it is hibernated. Our support servers are linked to clients upon request by the bootstrap server. Distribution of clients is performed according to user groups and system functionality. As the object tracking functionality creates high computational complexity and is directly related to the user creating an annotation, server entities conducting the tracking can only support a limited number of clients. Although clients may be distributed over different servers, they may work collaboratively on the same video. Thus, we split clients based on a load-balancing approach, and split the functionality for coordination and bootstrapping from the functionality of supplying a client and performing its object-tracking requests to support servers.

Support servers have the objective of receiving and executing object-tracking requests when one of their assigned clients annotates a video. In addition, the support server stores annotations added by a client to the interactive video document. Once a support server is assigned to a client, requests and modifications made by this client to the interactive video documents are sent directly to the support server. Our interactive video system leverages *Amazon EC2 small-size instances*[2] as support servers. Each support server is based on a machine image that can be deployed at any time. It consists of the server software necessary to maintain interactive video documents and the object tracking algorithms. In our system, some support servers are kept as a backup for new clients, so that the support server's startup time has no influence on system performance. If a rapid increase in the user number (e.g., due to flash crowds) occurs, new support servers will be started, resulting in a significant delay. After a user specifies an interactive hotspot in a video frame, a tracking module on the support server automatically identifies the interactive hotspot in all subsequent frames. A *session management module* on all support servers coordinates modifications on interactive video documents. The next sections focus on the automatic object tracking in Section IV and the eventually consistent synchronization across distributed entities in Section V.

IV. Video Object Tracking

In the annotation phase, a user can mark a rectangular region, which is visualized as an interactive hotspot on the client's user interface. After it has been created by the user, the video object tracking uses the x- and y-coordinates of this rectangular region for tracking. We call this the *object region*, and it may move and change its size according to the position of the annotated video object. In the video frame in which the object region is created, features are extracted that are used for the automatic tracking. Features extracted include: Point descriptors using illumination, edge differences, optical flow, and color features. This variety of features helps our adaptive object tracking to leverage the fact that in different situations, tracking algorithms perform in different ways. Previous work has shown that an adaptive selection of appropriate tracking algorithms can improve the robustness of the tracking results in user-generated video [36] This robustness is hard to achieve, as user-generated videos suffer from degradations such as camera shakes and harmful occlusions; thus, recorded videos are often blurry or over/underexposed [37]. Therefore, our system analyzes properties of the object region to dynamically select the algorithm that works best for both the given video sequence and the object region.

We have implemented the following alternatives as described in Section II-B:

- the MeanShift color-based clustering algorithm [20],
- a template-based tracking approach,
- the Speeded-Up Robust Feature (SURF) extraction and tracking [21], in combination with the Kanade-Lucas-Tracker (KLT) [22].

The color feature-based *MeanShift* algorithm is used for tracking objects that have unique, and easily trackable, color distributions. Such video objects can be easily distinguished from other objects in the video frame. While *Speeded Up Robust Feature* (SURF) [21] is a reliable and quick feature extractor, but the extraction and comparison of features are still an order of magnitude slower in comparison to the MeanShift algorithm. Thus, to compare the descriptors of an *object region* with descriptors of a video frame, the hierarchical k-means and the randomized k-d tree are used; these are provided by the Fast Library for Approximate Nearest Neighbors (FLANN) [38]. In addition, most computational load caused by the extraction of SURF features can be performed in an offline pre-processing step. The idea of using the *Kanade-Lucas tracker* (KLT) [22]1981] is to compensate for the possibility of SURF-based tracking being unable to find a sufficient number of features for reliable tracking in time. This can be caused by object deformations or significant motion. In those cases, KLT is used to describe the object movement in subsequent video frames.

A. Adaptive Selection of an Algorithm

By leveraging different features and changing from one tracking algorithm to the other, our system achieves more robust tracking results. The selection process of the tracking algorithm is depicted in Figure 3. *SURF*-based tracking uses features that are scale- and rotation-invariant to the searched object region. Reliable results of this feature-based tracker require that a sufficient number of SURF descriptors can be extracted. The new object position is calculated when the number of matching descriptors exceeds a threshold of $N_{f,min} = 10$. This threshold has been empirically derived during the evaluation of our system, ensuring correct detection (true positive match) and risking tracked object loss. Two system design decisions were made in respect to *SURF*-based tracking. First, our approach attempts to find and track video objects over the course of a video sequence - not only within

[2]http://aws.amazon.com/ec2/

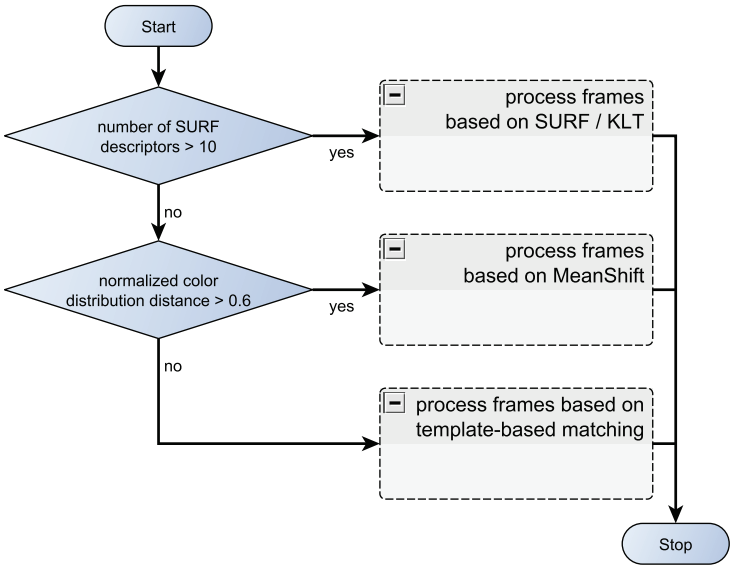


Fig. 3. The selection of the video object tracking algorithm.

the range of a video shot. Second, as moving objects can only cross a certain distance between two consecutive frames, we limit the distance between the descriptors. These descriptors are removed from further processing when the distance of such a match exceeds a defined threshold d_{max}.

MeanShift works as a segmentation algorithm, using colors to track the annotated *object region*. It is chosen when the object region's color distribution differs significantly from the color distribution of all frames in a video sequence (see Figure 3). The rationale behind this approach is that objects can be detected more reliably if their dominant colors are unique in comparison to the background. This works especially well for detecting the unique color of skin in face detection. The precision of using MeanShift for tracking is expected to be lower compared with the SURF-based tracking. However, the algorithm is, to a certain extent, robust with respect to scaling and rotation.

In the remaining cases, *template-based tracking* is chosen. It uses a brute-force comparison of color and intensity values in an object region versus the entire frame. In frames where the number of corresponding descriptors drops below the defined threshold, arbitrary features of the last detected object position are chosen to predict the object's position using the KLT in the next frame.

B. Potential for Reducing the Detection Time

Ideally, the tracking results of our system are required in near real-time to ensure that interactive hotspots are positioned according to movements of the annotated video objects. The following properties in our tracking algorithm have shown the highest impact on our tracking algorithm's runtime: 1) video resolution and frame rate, 2) the concurrent tracking requests on the support server that executes the algorithm, and 3) the selected tracking approach. Furthermore, capabilities of the server are important for ensuring a timely tracking completion.

To ensure a timely completion of automatic object tracking, some design decisions were made. For example, tracking is triggered instantly when an interactive video system user completes the first step of the annotation process. This step ends when a user has selected the object region. While the user is still adding information to the video object using information nodes, the adaptive object-tracking method is already processing the video. The intermediate results of the tracking are stored on the assigned support server and then - represented in the MPEG-7 notation - transferred first to the annotating client and next to other connected clients. This ensures that the client of the user creating a video object retrieves the tracking results first.

Additionally, the three algorithms pursue a coarse- to fine-grained tracking approach: When tracking is initiated, the algorithms are trying to find the video object's position in every 15th frame, i.e., every 0.5 seconds at a frame rate of 30 frames per second, or every 12th frame at a frame rate of 24 frames per second. Thus, the tracking algorithm is able to inform the client how to reposition the interactive hotspot as soon as the user continues video playback. Tracking information and object positions are transmitted - in MPEG-7 notation, as described in Section III-C - to the client. After coarse-grained tracking is complete, object positions are computed for all remaining frames. The interpolated object positions are then replaced by the exact tracking results. This last step is performed when the load of the server is low, e.g., only when some or no users are annotating video. Thus, after the user annotates a video and continues playing the clip, he will usually see the results of the coarse-grained tracking in our system setup.

In addition, we applied offline pre-processing steps: The SURF feature descriptors and color histograms of each video

frame are computed as soon as a video is available on one of our servers. In the case of the SURF descriptor-based tracking, this implies that only a comparison of the descriptors is necessary and not the feature extraction step. The only tracking algorithm that is computed immediately is the brute-force, template-based matching.

C. Robustness of the Tracking Approach

One issue that arises when tracking video objects is the occurrence of partial or full occlusions. Our system integrates an occlusion detection that is applicable if SURF descriptors are available. The occlusion detection is initiated when the number of matching SURF descriptors significantly drops in one or more quadrants of the object region. A first estimate of the position of the object region is calculated based on the remaining descriptors. To prevent occlusions from hindering the system in object tracking, a linear approximation of the feature movements is applied for the next three seconds to estimate if an object is only temporarily occluded. In those cases, the movement of the object during the occlusion phase is approximated, and the interactive hotspot will be positioned accordingly. More computational intensive algorithms with a higher reliability include methods relying on detecting edge changes [39] or object movements [17].

Scaling of a tracked object region can be easily handled by our approach, as all three tracking methods (SURF/KLT, MeanShift, and template-based tracking) can detect scaling and rotation. While SURF descriptors are explicitly chosen to be invariant under rotation and scaling, and indeed achieve the highest robustness of the three approaches, MeanShift relies on the representation of color distributions in the form of histograms. The algorithm tries to find similar regions in the frames with similar color distributions, independent of the structures or proportions that those regions represent. Template-based tracking robustness is enhanced by applying a pyramid style matching approach. This approach scales the object region into five versions of different sizes (33%, 66%, 100%, 133% and 166% of the original dimensions), as well as three rotations to the left and to the right (0°, 10°and 25°), re-performing the template-based matching with different object region versions.

The system relies on the quality of user input during the annotation phase. An object region created by the user must cover important parts of the object to be tracked and simultaneously be precise enough to not include too much background. To exclude background features that should not be tracked, we crop a border area of the template (up to 15%). Figure 1 visualizes the effect of imprecise selections. A good template created by a user is annotated in Figure 1 - a. In addition to several background features, the template includes exactly one semantic object, a goat. Selecting an object region that covers exactly one object ensures a reliable and fast tracking process. Figure 1 - b, on the other hand, shows that multiple semantic objects (two goats) are selected in one template. Our object-tracking approach is not able to distinguish those objects, thus creating a single object region. If, for example, the goats move away from each other, this would result in an increasing interactive hotspot size. A solution implemented in our system is that a user can manually change an object region at any time. Based on this new object region, there will be a retracking of the video object.

V. Maintaining Consistency

As described above, our interactive video system uses a distributed architecture in which different network entity types are responsible for executing different tasks. The synchronization of interactive video documents across those entities can be challenging. An example of the assignment of clients and the setup of the system is shown in Figure 4 - a. The bootstrap server is used to assign joining clients to their support servers. Support servers maintain copies of the interactive video document for each client. This copy is always kept consistent with the annotations made by the associated client. Besides the communication necessary for maintaining the consistency across network entities (depicted in in Figure 4 - a), the connectivity between network entities is monitored. For managing interactive video documents and maintaining eventually consistent server states, HTTP GET and HTTP POST requests are used. For a specific client, HTTP GET requests the initial interactive video document and requests for receiving updates from other clients. If this specific client creates new annotations or changes existing ones, it informs the responsible support server using HTTP POST. ICMP (Internet Control Message Protocol) ping requests are sent from the central bootstrap server to the support servers in order to signal the status of the servers. Clients regularly report their availability using the HTTP POST requests. The Network Time Protocol (NTP) is used for the clock synchronization between network entities.

A. Occurrence of Inconsistencies

Inconsistencies can arise when several users, working on copies of the same interactive video document, edit the same video object when it is not yet consistently persisted on all network entities. This consistency is broken as soon as one user edits a video object, adds information nodes, or changes the interactive hotspot location manually (reinitiating object tracking). Until the user completes the annotation process and its changes are processed by all sites, any annotation by other users to the same video object implies an inconsistency that has to be resolved by our consolidation process (see Subsection V-B). Such inconsistencies can occur often in our system, as concurrent annotations of videos by users are an intended aim of our system.

In our system setup, we aimed for approximately 200 concurrent users. Our concepts are not limited to hundreds, but potentially thousands of concurrent clients and hundreds of concurrent support servers. Their concurrent annotations may generate multiple megabytes of conflicting updates until consistency can be reestablished, as annotations can have rather large data sizes. The data includes *annotations* created

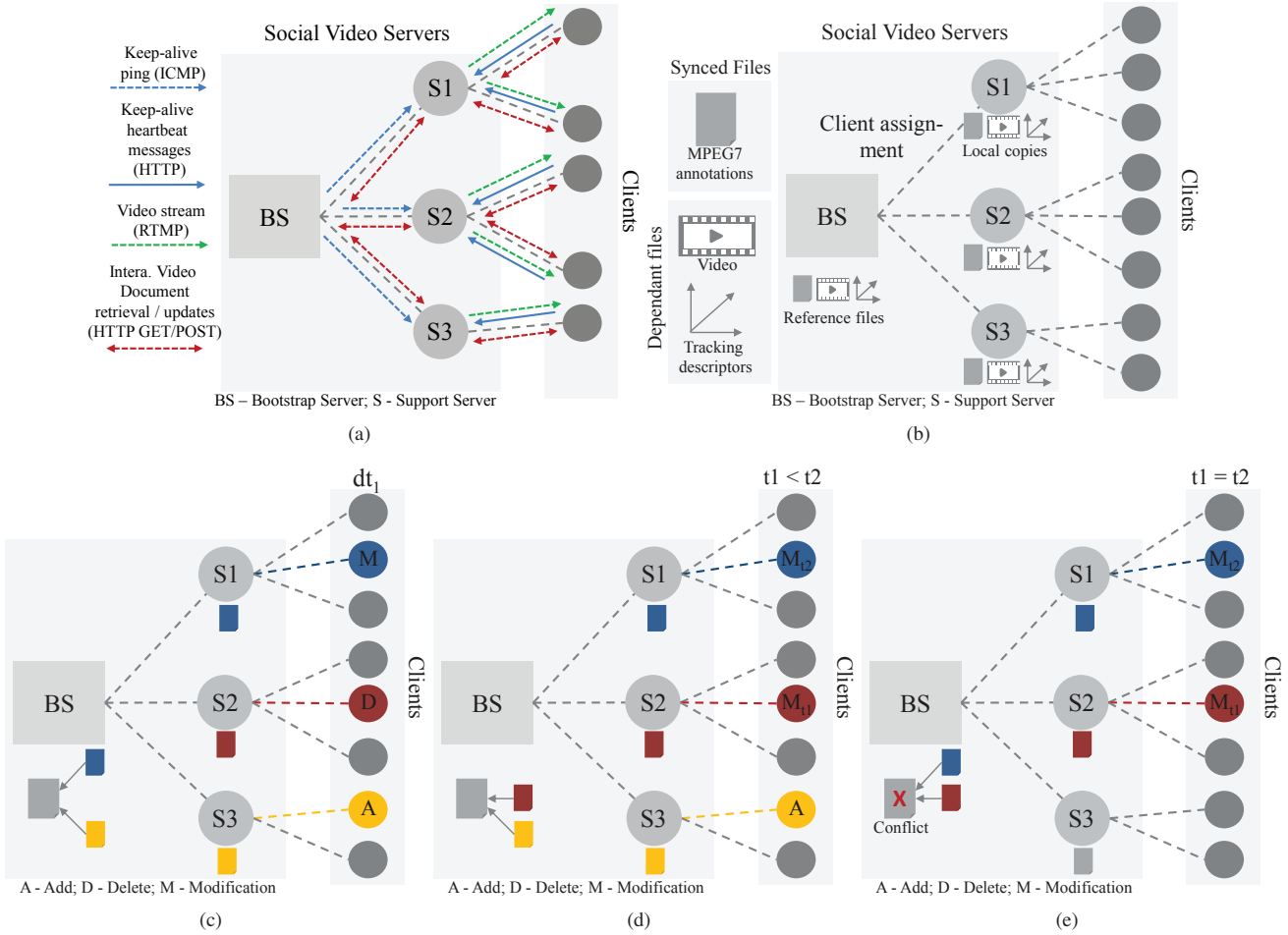


Fig. 4. Top: (a) Illustrating protocols used for video streaming, keeping messages alive, and managing interactive video documents (b) Overview of synchronization using one bootstrapping (BS) and three support servers (S). Bottom: Examples that (c) illustrate adding and deleting information, (d) show two modifications at different points in time that have to be merged, and (e) result in a conflict, as two clients modified the same annotation at the same time.

by the user as well as the *results of object tracking*, e.g., as a user initiated a tracking of an existing video object.

Every support server is responsible for managing the annotations of the assigned clients, and thus a copy of the interactive video document without consultation of any other server. Annotations can then be transferred to the bootstrapping server as soon as no assigned client is annotating the video anymore. Yet, in the meantime, another client assigned to another support server could have edited the same video object, which will result in an inconsistency – as remaining servers are not immediately informed if a client annotates a video. Thus, each support server is responsible for monitoring each of its assigned clients whether it is annotating a video by adding metadata to the interactive video document (see Figure 4 - b). Inconsistencies are then resolved first at the bootstrap server; then, the solution is propagated to the remaining support servers, and finally, the clients. By keeping changes locally on the support server, communication and synchronization overhead is kept low, and the system remains responsive, as objects are not locked for editing. Every annotation in the client's copy is identified by a managed identifier and a timestamp maintained on their support servers.

In addition, *results of the automatic object tracking* need to be synchronized. For example, SURF features consist of 64 dimensions of floating point numbers, where, depending on the resolution of a video frame, multiple thousands can be found in a single video frame. Even though clients do not require this data, multiple support servers exist that require establishing consistency across these features. Again, the descriptors and features needed for tracking are not persisted using MPEG-7 notation, but stored as independent files from the interactive video document that is used by the clients.

B. The Consolidation Process

We explain the consolidation process by the examples shown in Figure 4 c-e. This consolidation process is applied for all annotations and tracking results. The bootstrap server, using each annotation's timestamp, coordinates the consolidation. Consolidation in eventual consistency-reliant systems relies on an application-specific logic that the system designer has to de-

fine. The consolidation process specifies how inconsistencies for a specific application should be resolved.

Server consolidation is based on three simple rules:

1) New information is always added to the reference interactive video document;
2) a modification of an information node or a video object weighs more than its deletion;
3) for two modifications by two different users, the timestamp of an annotation determines which one is persisted; an earlier change survives.

These rules are applied individually for each information node. Information nodes have unique identifiers that allow easy retrieval and modification in even large interactive video documents. If conflicts occur between copies of an interactive video document, the corresponding information node identifier is retrieved, and rules are applied to this information node in the reference interactive video document.

The first rule implies a growing interactive video document. The second rule addresses inconsistent states between two entities of the same video object that are modified on two support server entities (S1, S3) and deleted on a third (S2). Consider the case as depicted in Figure 4 - b. The deletion of an annotation is reverted after all entities are synchronized and changes are persisted. Additionally, the first two rules mimic the behavior of other eventually consistent applications, e.g., see Vogel et al.'s work [24] – ensuring that users are always able to write their changes to the interactive video document. The third rule may result in conflicts, as modifications on the same video object that cannot be resolved by rule 1 or 2 are sorted by the timestamp in ascending order. If the changes cannot be merged automatically, e.g., a change on the same line of a hypertext node, the earlier change will be persisted. The user contributing the latter change will be informed that his changes have been discarded. This design decision is made in order to establish a growing video document, in which inconsistencies are resolved based on the idea that the first annotation always wins. This contradicts classical approaches, which take into account lost updates but only save the last change. In our eLearning scenario, this would mean that we want to motivate users to quickly enrich the video with new information. The changes made by a quick user should first be visualized to other users currently annotating the same information node, so that they can react to it. Consider a long text information node changed by user A: Even though the changes to a node of user B have been discarded, he can now (based on the updated text) decide whether to add information again or keep the updated version of user A. More formally, we assume that the latter modification is based on a version at t_2 of the element, which did not see the changes of t_1 (see Figure 4 - c). Thus, changes of t_1 should be synchronized first, and the network entity conducting changes at the same element at a time t_2 is informed.

C. Implications of the Design

We note that changes to the same element of a video object on two different server entities at exactly the same time are very unlikely. This unlikely case results in a conflict that has to be resolved manually. This conflict is resolved by creating two copies of the element and letting the users decide which elements are persisted (see Figure 4 - e). This is a disadvantage of relying on an eventually consistent paradigm.

When the load on a support server (the CPU utilization across all cores) decreases, e.g., when users are leaving or no changes affect the interactive video document, the support server transfers local modifications to the central bootstrapping server. This process is automatically aborted when a new client request is retrieved. Synchronization times can differ significantly. The bootstrap server distributes changes to individual support servers, incorporating changes into their copies of the interactive video document. Even under a high load, consistency is established at some point in time after updates cease. Due to our synchronization functionality, all users watching a video will operate on the same data after a while.

Because the system relies on cloud capabilities, it scales to multiple support servers and a large number of users. However, the bootstrap server is a single point of failure; when this entity fails, no new users can join the system. It should be mentioned that active users are not disconnected because every user client is assigned to one support server, and there is no need to directly communicate with the bootstrap server. When the bootstrap server is available again, every support server sends its reference document with the changes to it, which resolves inconsistencies. Note that in situations when a lost connection is reestablished, operations performed on the interactive video documents are idempotent.

VI. Evaluation

In this section, we discuss the performance of our distributed, interactive video system. Experiments were designed with three central questions in mind: First, how do users annotate objects in a video (Q1)? Second, how do users perceive the support provided by the automatic object tracking algorithm (Q2)? Third, what are the implications for users when a weak form of consistency is applied for synchronizing their annotations (Q3)?

Facebook was selected to host our interactive video system in order to easily reach potential users. Users recruited for our evaluation were mediated by running our system's client software as a Facebook application. An uncontrolled evaluation was conducted to assess the system performance under realistic conditions. Tutorials assisted users in system use. A snow-sampling approach was used to gather users by starting recommending the system to acquaintances on Facebook, which in turn motivated others to use our system. The interactive video system attracted more than 12,000 users for four months in 2011 (May-August). Around 45.16% of the users used the system longer than 240 seconds, showing a successful deployment.

We describe the results of our evaluation in three subsections. First, the annotation process and user interactions are evaluated (Q1). We integrated a *survey module* into our system

TABLE I
NUMBER OF USERS THAT RATE THE DIFFICULTY TO CREATE NEW INFORMATION NODES ON A MEAN OPINION SCORE (MOS) BETWEEN 1 (VERY COMPLICATED) AND 5 (VERY EASY) (SEE ONLINE APPENDIX: A12-A15).

Node \ Value	1	2	3	4	5	Average	Variance
Image/Text	9	6	65	77	68	3.84	1.03
Video	11	28	141	33	12	3.03	0.68
Chat	0	15	80	66	64	3.80	0.87
Web	6	18	72	61	68	3.74	1.12

to measure user satisfaction with specific system features. In total, the survey module asked up to 48 questions, which addressed areas including annotating videos using our system, errors in video object tracking that were obvious to users, and transparency of the eventual consistency paradigm (see Online Appendix). Each question was answered exactly once by each user. Only 225 of more than $12{,}000$ users were required to answer these survey questions. Other questions in our survey were designed to determine the performance of every system feature. After its first use, each system feature triggered a respective subset of survey questions. Features which were evaluated include, e.g., marking a video object with an interactive hotspot and adding information nodes – including all the steps discussed in Section III-B, which are required by the navigation and annotation process. Ratings were based on a five-point Likert-scale [40], which allowed us to determine a feature's effect on the overall user satisfaction with the interactive video system. The survey included the answers of 225 randomly selected users, who completely answered all questions[3]. *Mixing cocktails* was selected as the topic for the evaluation of our distributed, interactive video system. The system included short video sequences describing distinct steps in creating a cocktail. Users could annotate the interactive videos and add links to their own cocktail recipes. The basis for these recipe videos were 18 short videos, each of them showing a small step in the process of cocktail mixing, e.g. floating or shaking. One complete recipe video was provided to demonstrate the aim of our evaluation.

Second, a quantitative evaluation of the automatic video object tracking was performed (Q2). Besides annotations, automatic video object tracking also affects the user experience. Precision of position detection and processing time are the two major metrics. The *precision* describes how often the correct position of a video object was detected in comparison to a manually determined ground truth. If the object is not located correctly in a range of 15 pixels around the object contour, this may lead to misinterpretations, as users are no longer able to identify which object is meant. *Processing time* indicates if the video object-tracking algorithm is capable of repositioning the interactive hotspots in time.

Third, the synchronization process which applies eventual consistency has been evaluated (Q3). In order to better understand the eventual consistency paradigm and its impact on users, all sessions of our system were analyzed to determine the time period of inconsistent interactive video documents. As one central bootstrap server existed in our setup – operating both as a tracking and load balancing entity – we measured the delay until a consistent state on this node was reached.

A. Annotations in the Interactive Video System

Embedding the interactive video system into a social medium such as Facebook allowed a high number of users to annotate, upload their own videos as annotations, and share their annotations with others. 23 new video clips were created that can be classified as cocktail recipes; in total, 41 video clips were available at the end of our evaluation. Users created a rather high number of recipes (78). Thus, we conclude that the general aim of our system to motivate users to contribute their own annotations was achieved in our cocktail mixing scenario.

Table I shows the aggregated ratings given in the survey module (see Online Appendix: A12-A15). It illustrates that the creation of all types of information nodes was perceived as very easy or easy by a majority of the users. During this evaluation, each user created on average 5.1 (variance: 2.4) annotations to a video object. Each annotated video clip has an average of eleven newly created video objects. The video clip duration was between 65 to 175 seconds.

In total, $1{,}148$ annotations were added during the evaluation. Most annotations (49.39% of the information nodes) are links to external Web-pages. The number of embedded text-image nodes was the second highest at 29.36%. 12.89% of annotations were links to other interactive video documents. This low value is probably caused by the increased effort, because it is more complicated for users to find suitable videos that are related to the topic of cocktail mixing.

A total number of 136 changes to information nodes showed the collaborative use of the interactive video system. Most of these changes (107) are based on text-image nodes. These changes either complete text or fill gaps in the logical structure. It is obvious that users helped each other in creating a broad knowledge base. This observation is strongly supported by an average usability score of 3.87 (1 = does not support at all; 5 = very helpful) and a variance of 0.81, when users are asked how the system supports the annotation and co-editing of videos between users (see Online Appendix: A40). On average, a user interacted with 18.12 information nodes during a session. We interpret this as a considerably high rate of interactions with a video. Especially, the idea of leveraging links between videos

[3]The user dataset consisted of 68 female and 157 male users with an average age of 27.01 years.

[4]Video Source: https://durian.blender.org/

TABLE II
DETECTION RATES OF OUR ADAPTIVE OBJECT TRACKING BASED ON SURF, MEANSHIFT, AND TEMPLATE-BASED TRACKING.

Video	No. reference positions	SURF/KLT	MeanShift	Template-based	Adaptive tracking
Football	126	97.62%	80.16%	1.59%	99.21%
Cars	42	100%	0%	0%	100%
Street	40	100%	0%	0%	100%
University I	170	98.24%	17.06%	25.88%	98.82%
University II	81	98.77%	0%	0%	98.77%
Occlusion	147	74.83%	0%	7.48%	73.47%
Sports	225	88.89%	6.22%	0%	90.22%
Windmill	440	86.82%	86.59%	23.18%	99.09%
Sintel I[4]	285	91.58%	0%	100%	100%
Sintel II[4]	213	73.71%	89.20%	48.83%	97.65%
Sintel III[4]	305	48.52%	0%	5.25%	49.51%
Rabbits	120	63.33%	0%	59.17%	70.83%
Flamingos	45	31.11%	100%	17.78%	100%
Highway	48	93.75%	0%	0%	93.75%
Lions	143	81.82%	0%	13.29%	89.51%
Prairie dogs	96	77.08%	0%	30.21%	81.25%
Overall	2,526	80.60%	30.09%	27.36%	88.16%

has been used by each user – on average in 77.6% of the available videos. This indicates that each user accessed more than three quarters of the available video nodes. With this, we can answer our first evaluation question (Q1) and conclude that by an easy authoring tool as implemented in our system, users tend to often annotate videos – and this ease of annotation is key to user satisfaction.

B. Automatic Video Object Tracking

Table II shows our results of the adaptive tracking algorithm in comparison to SURF-only feature matching, the MeanShift-only algorithm, and a template-based matching.

The *precision* of our algorithm (88.16%) shows a superior performance in comparison to SURF/KLT matching (7.56% improvement), which is the second best algorithm. Both MeanShift (30.09%) tracking and template-based matching (27.36%) performed significantly worse. The content of a video, its colorfulness, and the number of available SURF features in an object region determine how often each algorithm was initiated. In total, SURF/KLT tracking was used in 51.17% of all frames evaluated, the MeanShift algorithm in only 13.29%, and template-based tracking in the remaining 35.54%. The *false hit rate* gives the percentage of frames in a video in which our adaptive tracking algorithm calculated the wrong object region position. The adaptive tracking resulted in the lowest false hit rate (7.0%), whereas SURF/KLT-only caused 7.69% false classifications, the template-based matching generates around 20.37% false hits, and MeanShift-only caused more than 26.06% false classifications.

Latency is the second important metric for the interactive video system. While watching a video, a precise tracking approach has to ensure that interactive hotspots are positioned correctly. At Full-HD resolution and 30 frames per second, the system required between 2.66 and 3.48 seconds per video second, depending on the test system[5] used. Consequently, we reduced the analysis interval to every sixth frame of a video. These conditions are harder than in the deployed system, in which only every 12th frame (or 15th) was used for the coarse-grained tracking. Thus, the computation time could be reduced significantly, and the interactive hotspots were visualized correctly. The computation time (between 0.51 and 0.87 seconds) allows an object tracking in real-time. During the evaluation described above, users were asked to rate the quality and effectiveness of automatic tracking. In total, users initiated 383 tracking requests. Only 16 users (4.18% of the tracking requests) found incorrect object positions (see Online Appendix: B26). Such errors occur when occlusions, object deformations, or poorly selected object regions were present. None of the users of the evaluation reported a problem with slow tracking (Q2).

C. Maintaining consistency

For assessing eventual consistency mechanisms, we evaluated when the bootstrap server reached a consistent state after new annotations were added. From our system setup, synchronization of the support servers with the bootstrap server was done every time the support server retrieved no new annotations or requests by associated clients.

The cumulative distribution function (CDF) in Figure 5 - a shows the time it took until synchronization between the support server and the central registration server was achieved. The basis for this evaluation were 1,148 annotations that had to be synchronized. Within 4.1 seconds, 50% of all synchronizations were complete. Note that this duration was computed as the time interval between the object-tracking results (data saved to the MPEG-7 interactive video document on the support server) and the update on the central registration server, i.e., when document changes were integrated into the

[5]Test systems: Laptop: Intel Core i5 (4 x 2.27 GHz, 4 GB), Lab: AMD Opteron (2.33 GHz, 1 GB), Server: and Intel Xeon (2 x 2.66 GHz, 8 GB)

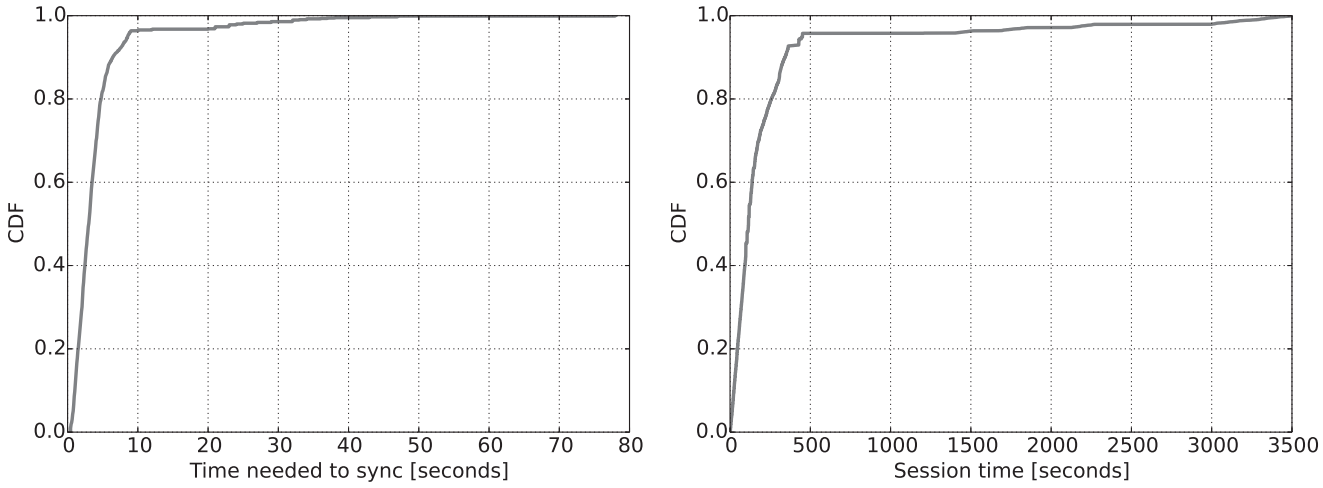


Fig. 5. An overview on the time the system requires to consolidate inconsistencies between the supporting servers and the central bootstrap server starting from the annotation of a user until the interactive video document is consistent on the bootstrap server in seconds (a) and the session lengths of users accessing the system within one month in seconds (b).

interactive video document. The CDF illustrates that at its peak, significantly larger times are required – up to 45 seconds. This results from concurrent modification, with a maximum of 216 parallel users. Yet, in more than 95% of all cases, the system reestablished consistency within 10 seconds. This is in line with the findings with the Amazon S3 system, which needs approximately 13 seconds for reestablishing consistency [29].

The collaborative annotation of video sequences is coordinated by one bootstrap server and three support servers. An interesting observation during the evaluation period was, that a constant load across all entities could be observed. For 34.7% of the active session length in peak usage times, the users observed inconsistent system states. This shows two things: First, eventual consistency in such a highly collaborative system can be applied without degrading the usability of the system; second, inconsistency arises for a large proportion of the times in our system. Approximately 82.9% of users recognized no synchronization errors at all (see Online Appendix: D44). 13.2% of the users recognized errors only when using the chat functionality. As chat is highly real-time dependent, all inconsistencies were obvious to users: elements such as backlogged topics or messages suddenly popped up in the user interface when consistency was restored. Thus for the third evaluation question (Q3), we conclude that a transparent usage of the eventual consistency paradigm is possible for an interactive video system. We can conclude that even weak consistency forms are typically imperceivable for most users and thus not reducing the perceived quality.

Figure 5 - b shows an overview of session times gathered from users who accessed the interactive video system within one month. Data from 9,393 sessions collected during this month was used to determine the average session length. The median lies at 115 seconds, with an upper quartile of 214 seconds. The median illustrates that despite a large percentage of very short sessions– where users decided to exit our system within a few seconds – half of the sessions lasted for almost two minutes, and 25% lasted for more than 3.5 minutes.

VII. Conclusion

This paper describes a system for collaboratively annotating video objects. As annotations are linked to objects in a video and visualized as an overlay close to the video object, they have to be continuously repositioned as the objects move. Our first innovation is an adaptive algorithm design for tracking annotated objects over all video frames. Our second innovation is the implementation of the weak consistency paradigm for multiple parallel annotations. Our system allows many users to annotate the video at the same time without waiting for a fully consistent state. While the world state might become inconsistent for some time, eventual consistency is guaranteed. To the best of our knowledge this is the first successful attempt to combine eventual consistency with a massively interactive and collaborative video annotation system.

An evaluation of our system with real users of the online social network Facebook shows that reduced consistency is easily accepted, often even going unnoticed. The proposed system is a successful step towards combining the weak consistency paradigm with interactive videos and automatic object-tracking algorithms. We conclude that our system offers scalability, consistency, and an enhanced user experience. Scalability is also maintained even if concurrent access rates on the same videos increase spontaneously, e.g. in flash crowd scenarios [41]. In these situations, the illustrated synchronization times between support servers would increase, but individual user responsiveness would not decrease. Although we evaluated our system with hundreds of users only, we believe that our results are equally applicable to much larger user numbers.

For future work, we plan to concentrate on the support of video annotation systems by large-scale content distribution methods. Additionally, researchers like Bailis et al. [26] promise that the advantages of eventual consistent, yet scalable, systems can be achieved with stronger consistency paradigms such as CALM (consistency as logical monotonicity) [42]. Thus, there are other paradigms for consistency in distributed systems than the classical full consistency and the eventual consistency discussed in this work.

VIII. Acknowledgements

This work has been funded by the DFG as part of the CRC 1053 MAKI. We are grateful to Carolyn Gale for carefully proofreading the draft version of our paper.

References

[1] S. Wilk, S. Kopf, and W. Effelsberg, "Social Video: A Collaborative Video Annotation Environment to Support E-Learning ," in *AACE Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (EdMedia)*, 2013.

[2] R. Laiola Guimarães, P. Cesar, and D. C. Bulterman, ""Let me Comment on your Video"," in *ACM Brazilian Symposium on Multimedia and the Web*, 2012.

[3] R. Laiola Guimarães, P. Cesar, and D. Bulterman, "Creating and Sharing Personalized Time-based Annotations of Videos on the Web," in *ACM Symposium on Document Engineering*, Sep. 2010.

[4] P. Cesar, D. C. Bulterman, D. Geerts, J. Jansen, H. Knoche, and W. Seager, "Enhancing Social Sharing of Videos: Fragment, Annotate, Enrich, and Share," in *ACM International Conference on Multimedia*, 2008.

[5] P.-Y. Chi and H. Lieberman, "Raconteur: Integrating Authored and Real-time Social Media," in *ACM Conference on Human Factors in Computing Systems*, 2011.

[6] O. Juhlin, E. Reponen, F. Bentley, and D. Kirk, "Video Interaction - Making Broadcasting a Successful Social Media," in *ACM Conference on Human Factors in Computing Systems*, 2011.

[7] N. Sawhney, D. Balcom, and I. Smith, "HyperCafe: Narratic and Aesthetic Properties of Hypervideo," in *ACM International Conference on Hypertext*, 1996.

[8] A. Girgensohn, F. Shipman, and L. Wilcox, "Hyper-Hitchcock: Towards the Easy Authoring of Interactive Video," in *Human-Computer Interaction*, 2003.

[9] D. C. A. Bulterman, "Creating peer-level video annotations for web-based multimedia," in *Proceedings of the Seventh Eurographics Conference on Multimedia*, ser. EGMM'04. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004, pp. 49–57. [Online]. Available: http://dx.doi.org/10.2312/EGMM/MM04/049-057

[10] B. Meixner, K. Matusik, and H. Kosch, "Towards an Easy to Use Authoring Tool for Interactive Non-linear Video," *Multimedia Tools and Applications*, vol. 70, no. 2, pp. 1251–1276, 2012.

[11] B. Meixner, S. John, and C. Handschigl, "SIVA Suite: Framework for Hypervideo Creation, Playback and Management," in *ACM Conference on Multimedia*, 2015.

[12] X. Mu, G. Marchionini, and A. Pattee, "The Interactive Shared Educational Environment: User Interface, System Architecture and Field Study," in *IEEE Conference on Digital Libraries*, 2003.

[13] T. Grossman and G. Fitzmaurice, "ToolClips: Investigation of Contextual Video Assistance for Functionality Understanding," in *ACM Conference on Human factors in Computing Systems*, 2010.

[14] M. Mu, S. Simpson, C. Bojko, M. Broadbent, J. Brown, A. Mauthe, N. Race, and D. Hutchison, "Storisphere: from TV Watching to Community Story Telling," *IEEE Communications Magazine*, vol. 51, no. 8, pp. 112–119, Aug. 2013.

[15] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–43, Dec. 2006.

[16] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, "A Survey of Appearance Models in Visual Object Tracking," *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 4, pp. 1–48, Sep. 2013.

[17] S.-K. Weng, C.-M. Kuo, and S.-K. Tu, "Video Object Tracking using Adaptive Kalman Filter," *Journal of Visual Communication and Image Representation*, vol. 17, no. 6, pp. 1190–1208, Dec. 2006.

[18] D. Goldman, C. Gonterman, B. Curless, D. Salesin, and S. Seitz, "Video Object Annotation, Navigation, and Composition," in *ACM Symposium on UI Software and Technology*, 2008.

[19] P. Sand and S. Teller, "Particle Video: Long-Range Motion Estimation Using Point Trajectories," *International Journal of Computer Vision*, vol. 80, no. 1, pp. 72–91, May 2008.

[20] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based Object Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[21] H. Bay and T. Tuytelaars, "SURF: Speeded up Robust Features," in *European Conference on Computer Vision*, 2006.

[22] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *ACM International Joint Conference on Artificial Intelligence*, 1981.

[23] E. Brewer, "Towards Robust Distributed Systems," in *ACM Symp. on Principles of Distributed Comp.*, 2000.

[24] W. Vogels, "Eventually Consistent," *ACM Queue*, vol. 6, no. 6, p. 14, Oct. 2008.

[25] Y. Saito and M. Shapiro, "Optimistic Replication," *ACM Computing Surveys*, vol. 37, no. 1, pp. 42–81, 2005.

[26] P. Bailis and A. Ghodsi, "Eventual Consistency Today: Limitations, Extensions, and Beyond," *ACM Queue - Storage*, vol. 11, no. 3, Mar. 2013.

[27] S. Gustavsson and S. F. Andler, "Self-stabilization and Eventual Consistency in Replicated Real-time Databases," in *ACM Workshop on Self-healing systems*, 2002.

[28] M. R. Rahman, W. Golab, A. AuYoung, K. Keeton, and J. J. Wylie, "Toward a principled framework for benchmarking consistency," in *USENIX Workshop on Hot Topics in System Dependability*, 2012.

[29] D. Bermbach and S. Tai, "Eventual Consistency: How soon is Eventual? An Evaluation of Amazon S3's Consistency Behavior," in *ACM Workshop on Middleware for Service Oriented Comp.*, 2011.

[30] N. Bouillot and E. Gressier-Soudan, "Consistency Models for Distributed Interactive Multimedia Applications," *ACM SIGOPS Operating Systems Review*, vol. 38, Oct. 2004.

[31] A. Bouajjani, C. Enea, and J. Hamza, "Verifying Eventual Consistency of Optimistic Replication Systems," *ACM SIGPLAN Notices*, vol. 49, no. 1, Jan. 2014.

[32] I. Mirbel, B. Pernici, T. Sellis, S. Tserkezoglou, and M. Vazirgiannis, "Checking the Temporal Integrity of Interactive Multimedia Documents," *ACM International Journal on Very Large Data Bases*, vol. 9, no. 2, pp. 111–130, 2000.

[33] S. Burckhardt, M. Fähndrich, D. Leijen, and B. P. Wood, "Cloud types for eventual consistency," in *ECOOP 2012 Object-Oriented Programming*, ser. Lecture Notes in Computer Science, J. Noble, Ed. Springer Berlin Heidelberg, 2012, vol. 7313, pp. 283–307.

[34] S. Kopf, S. Wilk, and W. Effelsberg, "Bringing Videos to Social Media," in *IEEE International Conference on Multimedia and Expo*, 2012.

[35] J. M. Martinez, R. Koenen, and F. Pereira, "MPEG-7: The Generic Multimedia Content Description Standard," *IEEE Multimedia*, vol. 9, no. 2, pp. 78–87, 2002.

[36] S. Wilk, S. Kopf, and W. Effelsberg, "Robust Tracking for Interactive Social Video," in *IEEE Workshop on Applications of Computer Vision*, 2012, pp. 105–110.

[37] S. Wilk and W. Effelsberg, "the influence of camera shakes, harmful occlusions and camera misalignment on the perceived quality in user generated video," in *IEEE International Conference on Multimedia and Expo*, 2014, pp. 1–6.

[38] M. Muja and D. G. Lowe, "Fast Approximate Nearest Neighbours with Automatic Algorithm Configuration," in *International Conference on Computer Vision Theory and Application*, 2009.

[39] A. N. Stein and M. Hebert, "Local Detection of Occlusion Boundaries in Video," *Elsevier Journal on Image and Vision Computing*, vol. 27, no. 5, pp. 514–522, 2009.

[40] C. D. McDaniel and R. H. Gates, *Marketing Research Essentials*, 1st ed. Taylor & Francis, 1998.

[41] I. Ari, B. Hong, E. Miller, S. Brandt, and D. Long, "Managing flash crowds on the internet," in *IEEE Analysis and Simulation of Computer Telecommunications Systems*, 2003.

[42] J. M. H. Peter Alvaro, Neil Conway and W. R. Marczak, "Consistency Analysis in Bloom: a CALM and Collected Approach," in *Conference on Innovative Data Systems Research*, 2011.