Robust Tracking for Interactive Social Video

Stefan Wilk, Stephan Kopf, Wolfgang Effelsberg
Department of Computer Science IV
University of Mannheim, Mannheim, Germany
{wilk | kopf | effelsberg}@informatik.uni-mannheim.de

Abstract

We propose a novel approach for tracking video objects in interactive multimedia systems. Instead of designing a single tracking algorithm that works well with some templates (video objects to be tracked) but fails with others, our adaptive algorithm uses three distinct tracking techniques with different strengths and weaknesses. The tracking technique that is selected for a given template depends on its visual content and the content of the video. Our approach is robust and allows tracking requests of users to be computed despite an imprecise selection of a template. In addition, our approach is designed to comply with high precision, speed, and scalability to support many users simultaneously. As an exemplary testing environment for this tracking system, an interactive hypervideo system was chosen. The system was integrated into the social network Facebook and used by more than 12,000 users.

1. Introduction

Current social network sites increasingly use multimedia and make it possible for users to customize their profiles. Platforms such as *YouTube*[1] or *Yahoo! Video*[2] define the current state of the art in interactive social media for videos and allow users to share, rate, and comment videos. In analogy to the term *social media*, we define *social video* as the use of web-based technologies to enable interaction with video content, to allow the creation of user-generated content, and to support social interaction with other users while watching or annotating videos. Our interactive social video system merges social media with a special form of interactive video called hypervideos. *Hypervideos* map the idea of hypertext (using hyperlinks to reference external websites) to videos by allowing users to interact with video objects.

[1]http://www.youtube.com
[2]http://video.yahoo.com

We define the following terminology: An *object* is a distinct element in a video clip. Its position typically changes due to camera or object motion. The task of a tracking algorithm is to locate the position and size of objects. A *template* is a rectangular patch that specifies the object region in one frame and is manually defined by a user. The template is used as input for the object tracking algorithm. The result of the object tracking is called *hotspot* which defines an interactive region in a video. It is used by the client to determine which pixels in a frame are associated with an annotated object.

A major requirement of object tracking is to guarantee a **high precision** in determining the position of interactive hotspots. Due to the integration into social media, several users can annotate arbitrary parts of a frame at the same time, and tracking has to be completed within a short time frame. Only algorithms that guarantee **high speed** and work approximately in real time are able to fulfill this requirement. The tracking module should be able to satisfy both speed and precision in a **rich set of videos**, assuming that new videos are uploaded by users and that arbitrary objects in the videos are searched. Multiple users can access the web-based system at the same time. Therefore, the tracking algorithm should **scale** according to the number of requests for tracking. Compared to previous approaches, the novel contributions of our system are:

- The proposed object tracker considers the most relevant requirements of collaborative hypervideo systems namely precision, speed, handling of incorrect object positions, and scalability.
- Considering three object tracking methods, we propose an automatic method to select the most suitable one based on the visual content of the template and the video.
- We analyze precision and latency of the adaptive object tracking algorithm in detail and present an evaluation with 225 users.

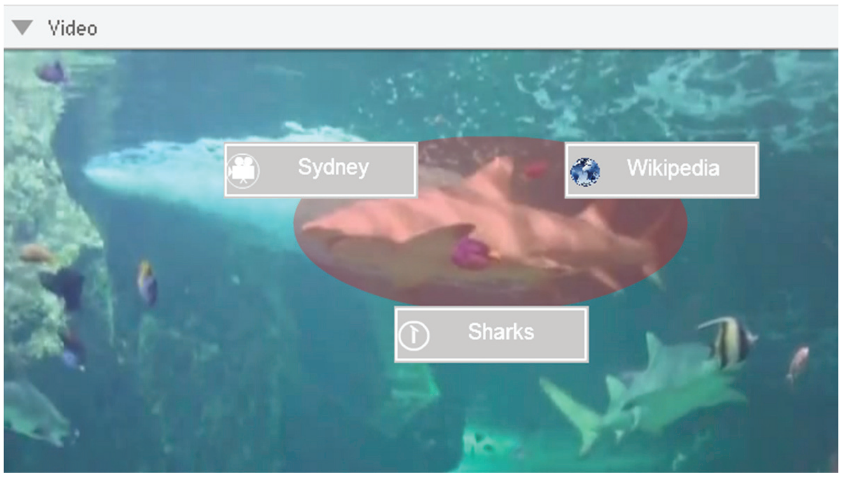

Figure 1. Using a hotspot to access information in a video

The remainder of the paper is structured as follows. Section 2 describes related work. The overall system and the details of the tracking algorithm are presented in Sections 3 and 4. The results of our evaluation are discussed in Section 5 followed by a conclusion and an outlook in Section 6.

2. Related Work

Several applications have been developed which support users to annotate spatial regions in videos. The *DIVER project* [8] records panoramic videos in classrooms to analyze the human activity in teaching situations. The system allows to annotate a region in a video frame by adding text. The *HTIMEL project* [3] evaluates the hypervideo concept with a focus on distance learning. *Hyper-Hitchcock* [9] implements a specialized type of hypervideo that supports a single hyperlink in a frame. This link does not annotate an object but a certain time interval of the video. Similar to HTIMEL, the Hitchcock system lacks support for automatic tracking algorithms. *Advene* [1] supports multiple annotations at a time but it still annotates time periods and no distinct objects. In comparison to our application, none of these hypervideo systems provide a fast and robust object tracking.

We present an adaptive approach that analyzes the template and video content and selects the most suitable object tracker, based on the MeanShift algorithm [4], template-based matching, and SURF features [2] combined with the Kanade-Lucas optical flow tracker [6]. These algorithms were chosen due to their ability to perform well in distributed environments with limited resources and especially without the support of GPUs. The idea of *MeanShift* [4] is to identify regions with characteristic colors. Considering the color distribution of the template, the algorithm begins by setting 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. *Template-based matching* is a brute force approach that compares color or intensity values between a template and a series of consecutive frames. To reduce the run time of such an approach, template-based matching is usually implemented by only regarding the local neighborhood of the previously determined template position.

Speeded Up Robust Feature (SURF) [2] is a fast method for detecting and extracting feature descriptors that are robust to scaling and rotation. Feature correspondences are defined as nearest neighbors between the features of a template and an arbitrary frame. The computational effort to compare feature elements is very high. Algorithms like hierarchical k-mean trees or randomized k-d trees provided by the *Fast Library for Approximate Nearest Neighbors* (FLANN) [7] allow a more efficient comparison of features.

The idea of the *Kanade-Lucas tracker* (KLT) [6] is to calculate the pixel displacements of consecutive frames on the basis of motion vector fields. Especially in case of object occlusion, where SURF feature matching is no longer possible, we use KLT to estimate the object position. In earlier work, we tracked Harris feature points to estimate the camera motion and segment moving objects in video sequences [5]. The two-dimensional shapes of the segmented objects were used for objects classification. The computational effort of this approach is too high to be applicable to our hypervideo scenario.

3. System Overview

Our system implements hypervideos as a combination of video and additional information of any type. Users should be able to access information (navigate) within a hypervideo via an interactive hotspot (see Figure 1) and to add their own ideas to the hypervideo (annotate). Supporting users in both processes – the *navigation and annotation* of a video

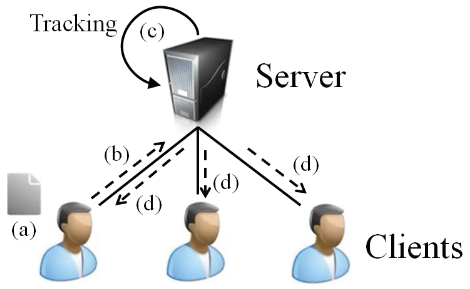

Figure 2. Overview of the tracking process: user defines template region (a), information about region is sent to server (b), adaptive object tracking (c), hotspot regions are sent back to all clients (d).

– is the major goal of our system. Reliable object tracking is one of the most important features in such a system. Due to the large number of frames in a video, a manual object tracking by users is inapplicable. The tracking should be done in a transparent way: users should not be aware of the underlying functionalities and required resources.

Figure 2 gives an overview of the tracking process. Tasks like the communication between user clients and the server, the simultaneous access of multiple users, and the automatic tracking functionalities lead to a system design based on a distributed web application. It is implemented to operate in a cloud-like environment and thus distributes the computational load of tracking to multiple instances of the server.

4. Object Tracking

The standard concept for representing navigation in hypervideos is the usage of *hotspots* that build an overlay on video objects. Users can simply mark a new object by clicking and dragging the mouse on the video screen. The selected region is the template that will be automatically searched in the hypervideo. It is essential that users only have to mark an object once and that the hotspots follow the movements of annotated objects throughout the video.

4.1. Selection of a tracking algorithm

We known from experience that it is not possible to get robust results by using only one object tracking algorithm due to the different characteristics of the video objects but also due to changes in the background of the scene. Our system implements an adaptive rule-based approach. It is adaptive in respect to the templates that users can define. The algorithm decides dynamically which of the following three tracking algorithms is used:

- Speeded Up Robust Features (SURF) [2] combined with the Kanade-Lucas-Tracker (KLT) [6],
- MeanShift algorithm [4], and
- Template-based matching.

The general idea is to analyze the properties of the template and the underlying video to automatically select the algorithm that works best. SURF, for example, uses feature descriptors that allow a scale- and rotation-resistant localization of a template. However a template can only be tracked in case that it contains a sufficient number of SURF descriptors. In case of occlusion or object deformation, the number of corresponding SURF feature points drops, and KLT is used instead. KLT calculates the motion of arbitrary pixels but it is computationally expensive.

MeanShift, in contrast, works as a segmentation algorithm using colors to track templates. This approach is best used in situations where the color of the template is unique; this helps to reduce misclassification of objects and thus false hits. *Template-based matching* can always be used because it does not rely on distinct color distributions or on a minimum number of SURF descriptors. But template-based matching is very sensitive when objects are scaled, rotated, or in the case of perspective camera motion.

We have implemented an *adaptive* object tracking algorithm that switches the tracking according to the current properties of the template and the video. The number of SURF descriptors of the template defines whether to use SURF-based tracking or not. Features are accepted for the tracking if two requirements are fulfilled: the first requirement selects features that are not located at the borders of the template. Relevant features are classified by reducing the template size by 10 percent at each border. This reduces the impact of the inaccuracy of users that draw the rectangular region in the first phase of the annotation process. A second requirement restricts the chosen features by setting a minimum distance of 10 pixels between two features. Our algorithm uses SURF matching if at least ten valid features are located in a template.

If the number of feature correspondences drops after only a few frames, KLT is used to support the SURF feature matching. In frames where the corresponding features drop below the minimal threshold, arbitrary pixels within the last accepted object position are chosen to estimate the position based on KLT in the next frames. If the position is successfully estimated, new SURF descriptors are extracted and added to the template feature set.

A major problem in the tracking of objects is the occurrence of partial or full occlusion. Occlusion reduces the number of features for tracking and at the same time the occluding objects generate new features. The system should reliably track an object without accepting features from foreign objects. The occlusion management is initiated in situations where the found matches of SURF features significantly drop within one or more quadrants of the template. Features in these areas that cannot be mapped to the template are analyzed further: If the movement is linear uniform, it is assumed that all new features belong to a differ-

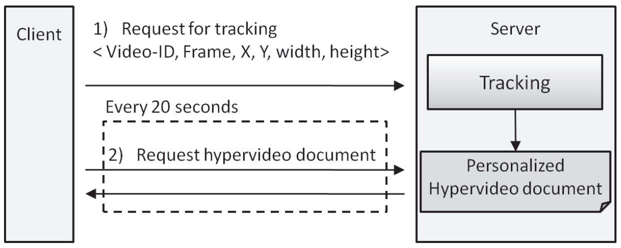

Figure 3. Communication while tracking

ent object and thus occlusion is present. We can linearly interpolate object movement in those situations.

If it is clear in advance that SURF feature matching cannot be initiated with enough features (less than 10) in the template, the MeanShift algorithm is selected for tracking. The average precision of MeanShift is significantly lower compared to SURF feature matching but it is still robust against scaling and rotation. The algorithm compares the color distributions of every frame of a clip with the searched template. MeanShift is selected for a shot when the color distribution of the template is different from the color distribution of most frames of this shot. Thus, the system does only apply MeanShift when the probability is very high that tracking works with high precision. The similarity measure is based on histograms with 180 bins in HSV color space. The system calculates the differences of the eight most significant bins and uses the MeanShift algorithm if the normalized distance exceeds a value of 0.6. This minimum distance is weighted according to the quota of template size and the video's frame size. This is necessary because the threshold should be smaller in cases where large objects are tracked. The assumption is that MeanShift can reliably track templates when the color distribution difference of the template and the current video shot is high. Shot detection was implemented but will not be described in detail here. In all other cases, in which neither SURF descriptors nor MeanShift is capable to track an object in a reliable way, template-based matching is used.

4.2. Support of real time tracking

Not only the precision is important for the practical usage of our hypervideo system but also the delay which a user has to wait until the automatically tracked objects are visualized. The system integrates several tricks to ensure high transparency for users, so that they are not aware of the underlying processes. A first trick assumes that users – after the annotation process is complete – will resume the current hypervideo. Tracking thus *starts from the current point in time* of the video and tracks the annotated object until the end of the video clip. Afterwards, the remaining part from the beginning of the video is computed.

Tracking is *initiated immediately* when a user finishes the first step of the annotation process – the selection of the rectangular region of the video object. Users are then adding additional information nodes. That may take from a few seconds up to several minutes. During this time, the tracking module processes huge parts of the hypervideo. The current intermediate results of tracking are transmitted from the server to all clients every 20 seconds (see Figure 3). When a user aborts the annotation process before completion, tracking results are discarded.

Despite the proposed optimizations, tracking based on SURF features is computationally expensive; on the average the system classifies approx. 1,000 feature points per frame. We reduce the 64 elements of each descriptor to the most relevant 20 elements by using principal component analysis (PCA). A slight reduction of the precision is the negative effect of the speed improvements. To compensate these negative side effects the motion in two consecutive frames of the detected features is limited to 10% of the image size.

Nevertheless, keeping the descriptors for all video frames in main memory may overwhelm the capacity of a server. One technique our hypervideo system uses to reduce the load is to *distribute the task to different machines*. New support servers using *Amazon EC2 small-size instances* are automatically started when necessary.

The calculation of the SURF PCA feature descriptors and the histograms used for MeanShift is started whenever a new video is added to the system. A registration server stores the data and transfers it to support servers when necessary. Matching of feature points is the only major computational operation when users annotate new video objects at a later date. We additionally improve the feature matching step by using a *fast approximation of the nearest neighbors* on the basis of FLANN [7].

A final optimization considers the fact that the position of an object does not change significantly between two frames. Therefore, it is not necessary to estimate the object position in each frame. A few estimations per second are usually sufficient, and missing object positions can easily be *interpolated*. The combination of all our optimizations allows the tracking of video objects up to PAL resolution videos in real-time.

5. Evaluation

The main task of our adaptive object tracking is to correctly position the interactive hotspots. The following evaluation measures whether this adaptive approach based on different algorithms is able to handle complex situations in different video sequences. Experiments were done on three separate systems shown in Table 1.

Precision and latency of the tracking algorithm are the most relevant requirements for an efficient usage of the hypervideo system. **Precision** measures the quality of detecting the correct position of a video object. Deviations

Table 1. Configuration of the test systems

System	CPU	Memory
I	Intel Core I5 4 x 2.27 GHz	4 GB
II	AMD Opteron 2.33 GHz	1 GB
III	Intel Xeon 2 x 2.66 GHz	8 GB

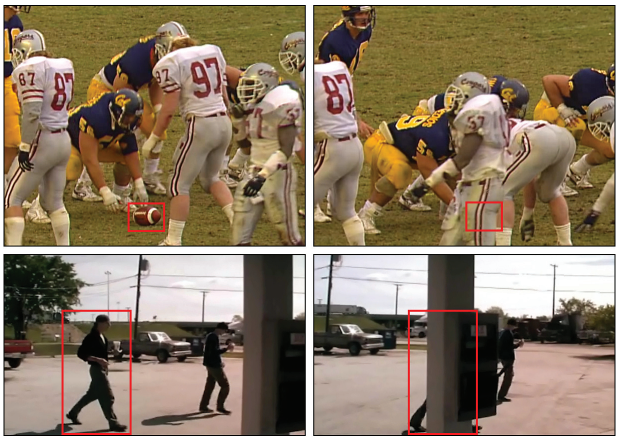
Figure 4. Object tracking with full occlusion.

Table 2. Object detection rates of the adaptive tracking algorithm

Video	Reference objects	SURF	Mean-Shift	Templ. based match.	Adapt. track-ing
Football	126	97.6%	80.2%	1.6%	99.2%
Cars	42	100%	0%	0%	100%
Street	40	100%	0%	0%	100%
Univ. I	170	98.2%	17.1%	25.9%	98.8%
Univ. II	81	98.8%	0%	0%	98.8%
Occlusion	147	74.8%	0%	7.5%	73.5%
Sports	225	88.9%	6.2%	0%	90.2%
Windmill	440	86.8%	86.6%	23.2%	99.1%
Sintel I	285	91.6%	0%	100%	100%
Sintel II	213	73.7%	89.2%	48.8%	97.7%
Sintel III	305	48.5%	0%	5.3%	49.5%
Rabbits	120	63.3%	0%	59.2%	70.8%
Flamingos	45	31.1%	100%	17.8%	100%
Highway	48	93.8%	0%	0%	93.8%
Lions	143	81.8%	0%	13.3%	89.5%
Prairie	96	77.1%	0%	30.2%	81.3%
Overall	2,526	80.1%	30.1%	27.4%	88.2%

may lead to misinterpretation or confusion as the user is no longer able to identify what the interactive hotspot annotates. **Latency** determines the period of time the algorithms need to calculate the position of a hotspot in the entire video sequence. High latencies may lead to the same negative effect as low precision. When the user resumes a video after completing the annotation, he expects the new interactive hotspot to move according to the object motion.

We selected a set of 16 video sequences which include situations where the tracking has to handle changes in illumination, deformation of objects, and occlusion (see Figure 4). Furthermore, sports sequences like football include rapid movements, zoom, and rotations. Videos with many scene breaks and different resolutions down to 320 x 240 pixels complete the test set. Low resolution videos are especially challenging as details are lost and thus the precision of feature-based approaches drops. As ground-truth data and a basis for the evaluation of the adaptive tracking, 2,526 reference objects in randomly selected frames were marked individually in the video sequences. Thresholds and parameters are used as described in Section 4.

5.1. Precision

Only a correct positioning of interactive hotspots guarantees a high user acceptance of the system. Table 2 shows the results of the adaptive tracking algorithm and compares them to the performance of SURF feature matching, the MeanShift algorithm, and template-based matching. Tracked positions and object sizes are labeled as correct if the coordinates of the corners are located within a radius of 15 pixels of the reference corners.

The precision of the implemented *adaptive tracking* (88.2%) shows a superior performance with an improvement of 8.1 % compared to *SURF feature matching* alone (80.1%). The more simple algorithms like *MeanShift* (30.1%) and *template-based matching* (27.4%) perform significantly worse. The adaptive usage of different algorithms helps to compensate the weaknesses of the individual tracking techniques in most cases. Figure 5 shows sample frames of three video sequences of our test set. *Cars* is a typical sequence where SURF feature matching works very well. The video sequence *Flamingos* is an example where MeanShift performs best due to the unique color of flamingos. Deformations of the tracked objects during playtime cause the low detection rate when using SURF feature matching. Template-based tracking works very robustly in case of sequences like *Sintel I* because the deformation of the object to be tracked (foot of the person) is very low. The template is very small which causes some tracking errors in case of SURF feature matching. Object and background colors are very similar in this sequence which makes MeanShift inapplicable.

The percentage how often each algorithm is used by the adaptive tracking depends on the content of the video. In detail, SURF feature matching is used in 51.17% of all frames of the evaluated test set, the MeanShift algorithm in only 13.29%, and template-based tracking in the remaining 35.54%. MeanShift performs especially well in scenes where humans are tracked.

The **false hit rate** gives an overview on the percentage of frames in which a tracking algorithm calculated the wrong position in comparison to the overall number of frames. The adaptive tracking (7.0%) generates the lowest error

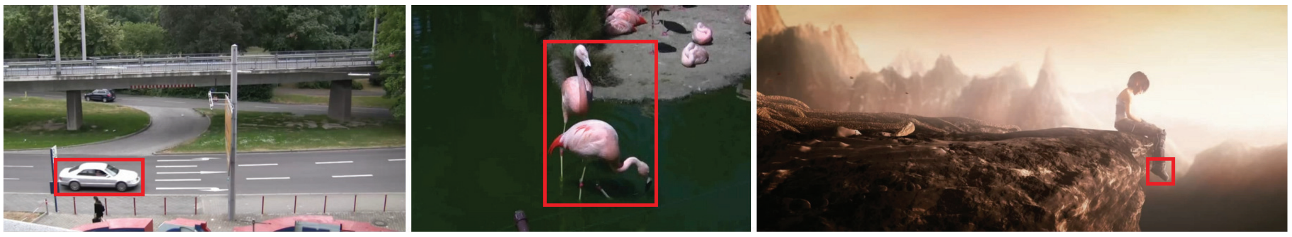
Figure 5. Sample frames and object templates of the sequences *Cars*, *Flamingos*, and *Sintel I*.

rate, followed by SURF (7.69%), template-based matching (20.37%), and MeanShift (26.06%). In case of occlusion (see Figure 4), our system compensates most of the partial occlusions (95.6%) and even handles full occlusions quite well (56.25%).

5.2. Latency

Latency is as important as precision for the hypervideo system. When users are watching a hypervideo, precise tracking is of no value if the system does not deliver the tracking results in time. When every single frame is analyzed, for every second in the video the system requires between 2.66 and 3.48 seconds depending on the test system used (see Table 1) to track an object. In case of SURF, between 59.6% and 81.2% of the calculation time is caused by the comparison of feature elements and the detection of the nearest neighbors. The computational effort would be much higher without using the fast approximations of nearest neighbors on the basis of FLANN [7].

By analyzing only every sixth frame of a video, the computation time is reduced, and the visualized hotspots appear even more stable due to the interpolation of the unknown object positions. This significantly reduces the computation time per video second (between 0.51 and 0.87 seconds) and allows object tracking in real-time.

5.3. User Feedback

The hypervideo client was developed as a Flash[3] application which allows a high accessibility due to the integration of the client into standard web browsers. Facebook was chosen as the environment for the user evaluation. This approach attracted more than 12,000 users who tested the system within four months. A survey tool was implemented and fully integrated into the hypervideo system. The evaluation with this survey tool was stopped after 225 fully completed evaluation data sets.

The evaluation tested the combination of annotation and navigation in the hypervideo system. The users started 383 tracking requests by defining new video objects with the mouse. 16 users (4.1% of the tracking requests) recognized some incorrect object positions. These errors typically occur in case of occlusion, object deformation, or when using small templates. In case of shot boundaries, the same object might be captured from a different camera perspective, and the object is lost due to sudden changes of object features (*e.g.*, an object switches from profile to frontal view). Tracking errors also occur when users define a template that includes a large ratio of background pixels.

Despite the incorrect object positions in some frames, the overall system was rated very good which could also be recognized in the large number of Facebook recommendations. Although a large number of users tested the system, speed limitations caused by the tracking algorithm have not been reported by any user.

[3]`http://www.adobe.com/products/flash.html`

6. Conclusions and Outlook

Our adaptive tracking approach allows automatic object tracking in an interactive hypervideo system. It guarantees a combination of high scalability with precision and speed. A suitable tracking technique is chosen by easily detectable features. Major requirements of our system are to handle common problems like occlusion of objects in videos and inaccurate selections of video objects by users, as well as ensuring a near real-time experience for a large number of users that collaboratively work with hypervideos. Due to the integration into the social network *Facebook*, we got much feedback from more than 12,000 users to improve our hypervideo system. The evaluation shows that the algorithm performs well in a rich set of sequences.

Still, there are several open issues for future research. Occlusion detection works in a majority of situations quite well, but an improvement could be to support the linear approximation of the object position with particle filters. Further possible enhancements include using the computational power of modern graphic cards. Although current instances of *Amazon EC2* do not support graphical computation on dedicated cards we expect that special services will offer this functionality in the future.

References

[1] O. Aubert and Y. Prié. Advene. In *Proc. of the 15th Int. Conf. on Multimedia*, pages 1005–1008, 2007.

[2] H. Bay and T. Tuytelaars. SURF: Speeded up robust features. In *Proc. of European Conference on Computer Vision*, volume 3951, pages 404–417, 2006.

[3] T. Chambel, C. Zahn, and M. Finke. Hypervideo design and support for contextualized learning. In *Proc. of the Int. Conf. on Advanced Learning Technologies*, volume 1, pages 345–349. IEEE Computer Society, 2004.

[4] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, May 2003.

[5] D. Farin, T. Haenselmann, S. Kopf, G. Kühne, and W. Effelsberg. Segmentation and classification of moving video objects. In B. Furht and O. Marques, editors, *Handbook of Video Databases: Design and Applications*, volume 8 of *Internet and Communications Series*, pages 561–591. CRC Press, Boca Raton, FL, USA, September 2003.

[6] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Conf. on Artificial Intelligence*, volume 3, pages 674–679, 1981.

[7] M. Muja and D. G. Lowe. Fast Appoximate Nearest Neighbours with automatic algorithm configuration. In *Int. Conf. on Computer Vision Theory and Application*, pages 331–340. Springer, 2009.

[8] R. Pea, M. Mills, J. Rosen, K. Dauber, W. Effelsberg, and E. Hoffert. The DIVER project: Interactive digital video repurposing. *Multimedia, IEEE*, 11(1):54 – 61, 2004.

[9] F. Shipman, A. Girgensohn, and L. Wilcox. Authoring, viewing, and generating hypervideo: An overview of Hyper-Hitchcock. *ACM Trans. Multimedia Comput. Commun. Appl.*, 5, 2008.