Journal of Visual Communication and Image Representation **11**, 154–182 (2000)
doi:10.1006/1999.0432, available online at http://www.idealibrary.com on IDEAL®

VORTEX: Video Retrieval and Tracking from Compressed Multimedia Databases—Multiple Object Tracking from MPEG-2 Bit Stream

Dan Schonfeld and Dan Lelescu

Signal and Image Research Laboratory, EECS Department (m/c 154), University of Illinois at Chicago, 851 South Morgan Street-1120 SEO, Chicago, Illinois 60607-7053

Received February 19, 1999, accepted June 22, 1999

Multimedia data are generally stored in compressed form in order to efficiently utilize the available storage facilities. Access to multimedia archives is thus dependent on our ability to browse compressed information. In this paper, a novel approach to multiple object tracking from compressed multimedia databases is presented. This approach is intended to operate in a distributed environment, where users initiate video searches and retrieve relevant video information simultaneously from multiple compressed video archives. The system operates on the compressed video to find and track objects of interest and determine their positions in the image. This enables more complex query formulations in terms of the relative positions of the target objects in the image. The filtering and analysis of motion information (motion vectors) is used to track objects in the video bit stream. Once the search has terminated, the system may decompress and display the query-relevant video sequences upon request. © 2000 Academic Press

1. INTRODUCTION

In the context of rapid expansion of multimedia databases all over the world, the need for efficient selective retrieval of video data is essential. The accelerated accumulation of multimedia information is due to the contribution of traditional video data sources such as the movie and television industry as well as the proliferation of commercially available video cameras. Additionally, satellite information has become an increasingly important source of video data for various environmental and military applications.

The volume of data stored in multimedia databases is so vast and expanding so rapidly that the only way to accommodate the data in existing facilities is to store it in compressed form. This is particularly important for video, which accounts for a large portion of existing data due to the large memory required to store a single image and the number of images contained in a single video. Existing video compression standards have become very commonly used in all video archives, and access to video information in multimedia databases is essentially restricted to compressed video information. The video collections are generally unindexed and they are normally accessible by decompression and sequential scanning. A direct approach to the development of visual search engines based on decompression of multimedia databases, followed by exhaustive matching of the templates provided, with each and every image in the video archive, is impractical due to the large volume of video data and the high cost of decompression. Therefore, efficient selective retrieval of video data must be based on methods that operate on compressed data.

In this paper, we present a novel approach to video tracking for retrieval purposes from compressed multimedia databases (VORTEX). The goal of VORTEX is the implementation of a visual search engine that operates in a distributed environment, where users initiate video searches and retrieve relevant video information simultaneously from multiple video archives. A user searching for relevant video information in a video database may formulate a video query in terms of sample or template images of objects of interest, attributes, or symbolic descriptions. Addition of logical conditions such as spatial positional information of objects or relative positions of multiple objects in the image is also possible.

Since video collections are often independent and located at multiple sites, VORTEX is intended to operate in a distributed environment. Systems based on distributed architectures use resources more efficiently and allow parallel processing. The system presented in Fig. 1 is a variation of the client–server paradigm [1], illustrating the operation of VORTEX in a distributed video database environment.

In this configuration, *clients* (users) are connected to *video archive servers* that run the visual search (VORTEX), through an intermediate *connection management server*. This server is responsible for the management of user requests and the return of query results to the appropriate clients. Operating on the compressed video data stored on a particular video server, VORTEX retrieves and tracks the objects of interest, thereby determining

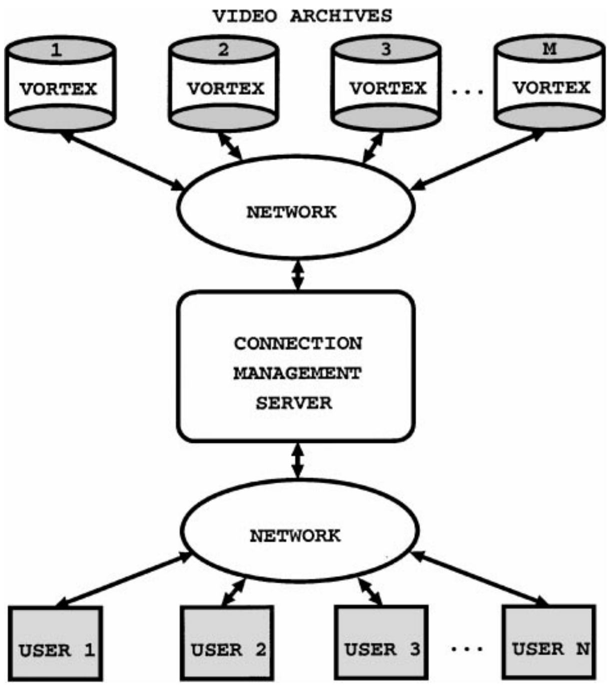

FIG. 1. VORTEX in a distributed environment.

the positions of the starting frames for the query-relevant video sequences present on that server.

Challenges associated with the application of VORTEX to retrieval and tracking include operating on compressed data, object recognition and tracking, and the speed of execution. Our approach exploits the structure of video compression standards in order to perform visual object matching directly on the compressed video data stream. The estimation and interpolation of the desired method for object recognition is achieved by utilizing motion compensation—a critical prediction filter embedded in video compression standards. In order to guarantee that the tracking error remains small, we intermittently perform an exact match using the direct method for template matching on selected images. We also address the issue of robustness of the system in conditions of affine transformations and partial occlusion of objects of interest. The detection of changes in the object's appearance and the corresponding adjustments that have to be made for correct tracking are formulated in the compressed domain.

The ubiquitous MPEG-2 video compression standard is used as a case in study, as a representative example of motion compensated video encoding. This is motivated by the fact that codecs based on standard compression schemes, such as the MPEG suite and its video-conferencing counterparts such as H.261 and H.263, are widely used and strongly supported by the industry. However, our approach to video retrieval and tracking can be adapted for use in the context of other compression schemes that rely on motion compensation.

The paper is organized as follows: In Section 2, related work in the area of content-based video retrieval and tracking is presented. A basic overview of the MPEG-2 video compression standard is presented in Section 3. Topics related to objects' search and video tracking initialization are discussed in Section 4. In Section 5, an efficient visual object tracking from compressed video data is proposed. Simulation results are presented in Section 6. A brief discussion and concluding remarks are provided in Section 7.

2. RELATED WORK

A number of methods for video analysis and retrieval have been investigated. Video sequences can be textually annotated prior to insertion in a database in an attempt to capture information that is deemed relevant for later retrieval. Manual annotation requires a large amount of work and time especially for large databases. Annotation is inevitably incomplete in describing the complex scenes of natural video. Also, manual annotation will likely be subjective, reflecting the operator's decision about what is considered to be relevant in a video sequence. Annotation methods are presented in Ashley *et al.* [2] and Saur [3].

Video parsing or segmentation has been addressed by many researchers with the purpose of automatically uncovering structure in video data, which is subsequently used for further video analysis. Segmentation consists of dividing the video into shots or scenes by detecting scene changes (shot boundaries). Video shots may be associated with a "key" or "representative" frame that best represents the shot. These key frames can later be used for the actual retrieval process. The scene change detection can be done by using the decompressed video data [4–7] (e.g., color histogram-based methods). Decompression of video data is a costly operation. Thus, recently, the emphasis has been on video segmentation by operating on the compressed video, which has been actively pursued, with numerous papers appearing in the literature [8–12]. The issue of imprecise detection of shot boundaries often arises in video segmentation in the compressed domain, especially in conditions of gradual transitions and special effects. The correct detection of shot boundaries is essential for retrieval. It must also be noted that key frames have a limited capacity to describe the activity in a video shot.

Content-based video retrieval and video indexing [13–25] have the potential to overcome the shortcomings of annotation approaches and to automate the retrieval process. These methods are more complex and present challenges related to the capabilities of object recognition in natural video scenes and the automatic extraction of semantic information for retrieval. Also, many video databases have been, and are, created without the inclusion of additional information such as indexes and contain only a collection of raw video data. Thus, any preprocessing when a query is initiated must be performed in real-time. In these approaches, low to high level representations (e.g., symbolic description) of images or video objects are used in order to facilitate retrieval. In [13], Dimitrova and Golshani combine symbolic descriptions of objects, which enables query formulation, with object activity detection (e.g., trajectory), for the retrieval of query-relevant images. Trajectories can be used as an additional element to help in searching for relevant video objects. Using an algebraic framework for data modeling and formulation of a query language, queries can be formulated in terms of object descriptions and trajectories. A clustering process is used to produce candidate trajectories in the video. In video indexing, either major video objects can be indexed for later retrieval purposes or frame-related data such as DC coefficients form vectors that represent those frames. In an object indexing and retrieval context, Meng and Chang [14, 15] extract low-level visual features such as camera motion, shape, and trajectories of prominent moving objects. Moving objects are extracted using motion vectors by thresholding the magnitude of the local motion followed by morphological operations to delete small false objects. This information is used for key content browsing and search in the CVEPS system. This system performs an efficient video segmentation (operating on compressed MPEG video), where camera operation parameters are estimated in each shot and shape and trajectory features of moving objects are extracted. A content-based image query system, VisualSeek [16], is used to index and retrieve key frames or video objects based on their visual features. Kobla *et al.* [17] introduce VideoTrails for representing and visualizing structure in video data. They build on earlier work on extracting low-dimensional spatial features from the frames of an MPEG video. VideoTrails analyzes a video clip by generating a trial of points in a low-dimensional space where each point is derived from the physical features—macroblocks, DC coefficients, and motion vectors of a single frame in the video clip. Thus, the approach relies upon indexing global data in a video frame. This generates clusters of points whose corresponding frames are similar and correspond to portions of the video clip where there is little change in content. Transitions between clusters indicate physical activity taking place in the video clip and can be used to detect more difficult scene changes such as gradual transitions. FastMap [18] is used to perform dimensionality reduction to generate the low-dimensional vector for each frame. Kobla and Doermann [19] index the mapped low-dimensional representation of spatial and temporal features of key frames which were obtained as a result of a segmentation process. Spatial features are used as a primary indexing and querying element and the temporal features are used to rank the video clips and enhance robustness. Park *et al.* [20] introduce VideoRoadMap [21] as a content-based video indexing system for retrieving video clips and images from multimedia databases. Spatio-temporal features and information modeling are used for indexing. For increased flexibility, adaptive similarity measures based on the characteristics of media objects are employed to improve retrieval.

VORTEX emphasizes the concept of operating on compressed video data and exploiting the critical motion compensation information produced by the encoder's motion-based prediction for real-time object tracking and video retrieval [26–29]. The central idea is to intermittently initialize detection and tracking temporally by using a video parsing method and at the same time initialize spatially by using an available pattern recognition technique (template matching, affine invariant matching, use object indexing data or feature vector information). Then, track efficiently under general conditions (affine transformations), until the next reinitialization point. Only the objects of interest are detected and, through motion analysis applied to compressed video data, directly tracked in the bitstream. Given the general availability of video in unprocessed form, we do not necessarily assume the presence of annotated or feature-indexed video, neither do we generate a set of principal objects in the video. Thus, our approach relies on an object recognition—object tracking paradigm—with the first component having the potential to be continually improved in accuracy and speed, given the advances in computer vision research.

3. MPEG-2 OVERVIEW

The MPEG encoder uses block-based motion compensation to reduce the temporal redundancy present in the original video sequence. Consequently, three picture types are defined. I pictures (intracoded) are coded independently without reference to other pictures. P pictures (predictive-coded) are coded using motion compensated prediction from the preceding I or P picture and are used as references for further prediction. B pictures (bidirectionally predicted) require both past and/or future reference pictures for motion compensation. B pictures are not used as references for further prediction. If B pictures are produced by the encoder, the *display order*, which is the order in which the pictures are received at the input of the encoder as well as the order in which they are displayed at the output of the decoder, is different from the *coded order*, which is the order in which the pictures are received at the input of the decoder. The coded order is constructed such that, at any time, reference pictures that are needed to decode another picture are present in the bitstream prior to the arrival of the referring picture at the input of the decoder (see Fig. 2).

Motion compensation is implemented at the level of subunits of an image having size 16×16 pixels, called *macroblocks* [31]. Motion vectors represent the transition from a macroblock in the current picture to the best-match macroblock in the reference picture(s). The origins of the motion vectors are located on the macroblocks-determined, sampling grid in the current picture. The destination of the motion vectors can be specified from any position in the reference picture(s). There may be macroblocks where motion compensation is not possible and, as such, they are encoded as intramacroblocks. Once motion compensation has been completed, the encoder uses a 2D discrete cosine transform (DCT) for spatial redundancy reduction. In I pictures, the original image samples are directly encoded. However, in P and B pictures, only the residual error after the prediction of macroblocks is encoded.

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	I	B	B	P	B	B	P	B	B	P	B	B	P	B	B	I	B	B	P
b	I	P	B	B	P	B	B	P	B	B	P	B	B	I	B	B	P	B	B
	1	4	2	3	7	5	6	10	8	9	13	11	12	16	14	15	19	17	18

FIG. 2. Picture correspondence in (a) display and (b) coded order.

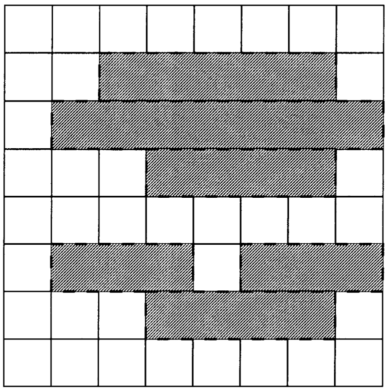

FIG. 3. Slices.

Following a quantization stage, a variable length (VL) entropy-coding—Huffman coding—is performed at the next stage in order to reduce the symbol redundancy. The decoder performs the inverse operations—VL decoding, inverse DCT, motion decompensation—in order to obtain a lossy reconstruction of the original video sequence.

Sequences of at least one coded macroblock that are restricted to a single row of macroblocks in a picture from *slices*. Figure 3 shows an example of slices' structure in a picture. Aside from predicted macroblocks, slices may also contain *skipped macroblocks*. For P pictures, skipped macroblocks can be seen as having zero motion vectors. For B pictures, skipped macroblocks have motion vectors that are similar in magnitude and direction to the motion vectors of the previous nonskipped macroblock in the slice. We may also have intramacroblocks in the slices of P or B pictures. Macroblocks that are not contained in slices are not encoded. The decoder simply replicates the corresponding macroblocks from the reference picture.

4. OBJECT SEARCH AND TRACKING INITIALIZATION

For the initialization of a tracking segment and for tracking error resetting, we perform object recognition only on selected pictures in the video sequence. We assume that the choice of the selected pictures in the video bit stream is generated based on the application, as well as on the availability of object recognition tools.

It is important to note that the focus of this paper is on object tracking for video retrieval and the proposed approach can be used in conjunction with any choice of video segmentation and object detection methods desired. We shall nonetheless provide a brief introduction to the topics of video segmentation and object detection since they must be used for tracking initialization.

Over the past few years video segmentation has been an active research area whose objective is the correct identification of the boundaries of visually coherent video segments. The result of a video segmentation procedure is a sequence of video shots, along with the positions of the starting frame of each video shot. VORTEX can use the structure assigned to the video data by video segmentation, i.e., the boundary images of the video shots and a degree of visual coherence of the shot. An object recognition method can be applied to the first I picture following the starting position for a shot, in order to detect the object(s) of interest. VORTEX can then be activated to track the target objects throughout the video shot in the compressed domain.

As we have stated earlier, any video segmentation method desired can be used for the initialization of the object tracking algorithm proposed in this presentation. Indeed, various methods for video parsing along with indexing and annotation of the video sequences may be used to greatly improve the efficiency of the tracking initialization component of the proposed algorithm. In circumstances when the creation of an off-line preprocessed video database is not feasible, however, an efficient method for real-time video segmentation must be used.

In the case where real-time video segmentation is required, the selected video segmentation method should be able to operate on the compressed bit stream for the purpose of efficiency and consistency with the operation of the VORTEX system. Ideally, the segmentation process should operate in parallel with VORTEX by sequentially detecting any scene changes so as to provide candidate locations of pictures where the object recognition method should be applied and tracking reinitialized. In general, however, video segmentation must be performed as a preliminary step before VORTEX is activated (i.e., a two-pass process).

As an alternative to video segmentation, a video sampling scheme can also be implemented. For this purpose we choose a variable number N_0 of frames (e.g., $N_0 = 1000$) that will determine how often we apply the object recognition technique to selected decoded I pictures. By reacquiring the object intermittently in the selected I pictures the tracking errors are reset and the tracking process reinitialized. An illustration of the operation of the sampling scheme is shown in Fig. 4.

The object search starts with the recognition method applied to the first decoded I picture, which also represents the start of the video sequence. If the object is not found, the algorithm moves ahead in the sequence by N_0 frames and the nearest I picture is decoded. If the object is detected in the decoded I picture, the tracking component of the algorithm is activated and it operates as presented in the following sections. The tracking continues until N_0 frames later, when the nearest I picture is decoded. Upon termination of the search process, the initial I frames corresponding to all relevant video segments in the video archive are available.

The object recognition techniques that can be utilized by VORTEX range from direct template matching, affine invariant matching, to usage of higher-level object descriptions such as symbolic description or attribute vectors. The latter information can be present in an indexed object database environment as part of a preprocessed video database that contains indexed information along with the corresponding raw video data. Since video segmentation is usually a preliminary step to indexing, video shots are often described by key frames. Choosing the first frame of a video shot as a key frame is in general a sufficient and a natural choice [30]. This is based on the observation that the shot is usually characterized by the first few frames, before the camera operator begins to track or zoom to a close-up. The information in the key frame is further indexed (e.g., main objects in the scene). VORTEX can exploit the option of having indexed key frame information, while the limitations of having an entire video shot described by a key frame are eliminated and replaced by explicit

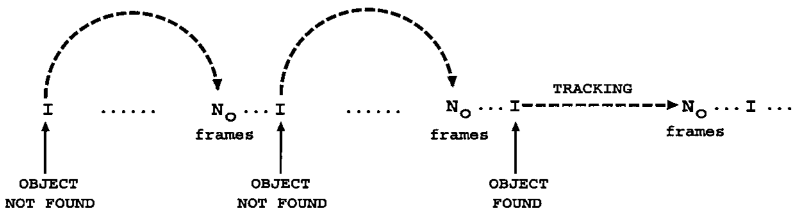

FIG. 4. Object search and tracking.

object tracking. Hence, pattern recognition based on the type of information that is available for different video databases can be utilized by VORTEX to acquire the query object(s) in selected pictures and thus initialize tracking.

Hereafter, we shall denote by *selected I picture*, the first I picture in an identified video shot, if usage of video segmentation is assumed, or the first I picture that marks the beginning of a sampling interval, if the video sampling scheme is utilized.

Once again, we emphasize that the tracking initialization may rely on any choice of video segmentation and object detection methods used. A thorough discussion of the topics of video segmentation and object detection is beyond the scope of this presentation.

5. TRACKING FROM COMPRESSED VIDEO DATA

VORTEX can work with various picture patterns in the encoded sequence (e.g., IPBBPBBPBB or IPBBBPBBBPBBB) and can also operate on video streams lacking B pictures (such as MPEG-1). As mentioned above, during tracking, we intermittently perform object recognition on selected decoded I pictures. This is done automatically if the video sampling scheme is used. In a video segmentation context, the boundaries of the video shots can be used to initialize tracking. We assume that tracking initialization by means of intermittent object recognition is available, using an external recognition module. Currently, for simplicity, template matching is used to acquire the object of interest in selected I pictures. Hence, it is assumed that, when the object initially appears in the video segment, it is similar in appearance to the image template—some variation is tolerated, to the degree where the correlation response surpasses the desired threshold. If direct template matching is used, in order to reacquire the object after a tracking segment, the algorithm can use the original template or the latest tracked version of that template. The system was designed so that any object recognition technique can be utilized and attached as an external module. Thus, an affine invariant object recognition method may eliminate the need for the above assumption. Figure 5 illustrates the exhaustive template matching and VORTEX's approaches to object retrieval and tracking from compressed video databases. The *interpolator* implements the

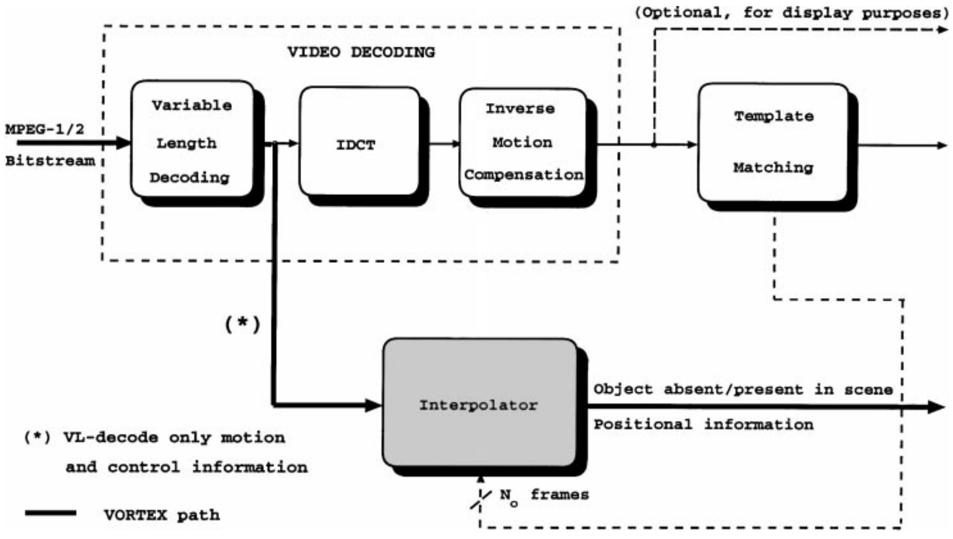
(Optional, for display purposes)
VIDEO DECODING
MPEG-1/2 Bitstream
Variable Length Decoding
IDCT
Inverse Motion Compensation
Template Matching
(*)
Interpolator
Object absent/present in scene
Positional information
N_o frames
(*) VL-decode only motion and control information
VORTEX path

FIG. 5. Exhaustive and VORTEX's approach to tracking for video retrieval.

estimation and interpolation of the desired method for object recognition by utilizing the motion compensation information. Thus, the costly decompression of the video data, for the purpose of retrieval and tracking of the objects of interest, is avoided.

From a procedural point of view, we have the flexibility of two modes of operation:

1. Object tracking: The algorithm operates on the compressed bit stream by only decoding the motion compensation information and occasional (selected) I pictures. Once the algorithm terminates, it provides the sequence position of the query-relevant video segments and the positions of the tracked objects in the relevant video segments. This information could then be used to display the relevant video sequences by using the corresponding initial I pictures as entry points in the video bit stream. In this case, significant computational savings are achieved since we decode only the minimum information needed for object tracking. This is VORTEX's normal mode of operation.
2. Object display: The second mode of operation assumes that the tracking and display of the video segments that contain the target are done simultaneously. This mode is used primarily for testing purposes. In this case, the decoding of the picture data corresponding to the relevant video segments is required. However, the algorithm used for tracking is identical to the previous mode.

The structure of the interpolator is illustrated in Fig. 6. For selected I pictures the algorithm performs the desired object recognition method for tracking initialization; whereas, for nonselected I pictures the algorithm proceeds directly with the motion analysis phase; and, finally, for predicted (P and B) pictures, the algorithm sequences through all stages of the tracking process: motion information acquisition, classification, filtering, and motion analysis.

5.1. Motion Information Acquisition

For details of the MPEG motion compensated prediction, the reader is referred to the video specification of the MPEG standard [31]. We shall describe the acquisition of motion information from the compressed bitstream as the preliminary stage of the tracking algorithm. Currently, VORTEX acquires and operates exclusively on the luminance video data, for matching and tracking of the target. For simplicity of presentation, we shall consider the case of tracking a single object. However, the algorithm has the capability to track multiple

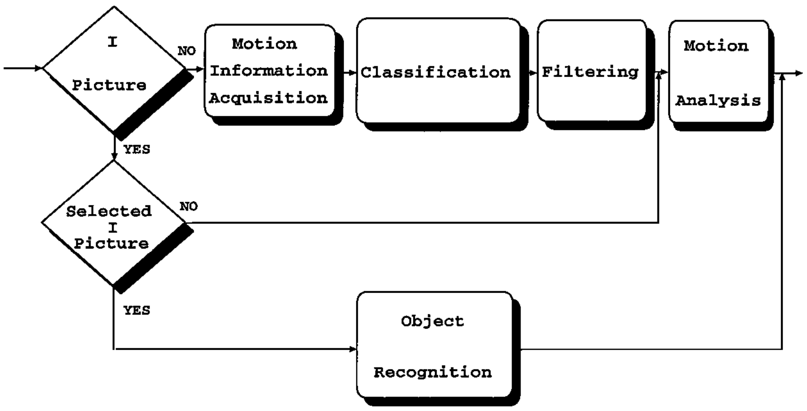

FIG. 6. Interpolator.

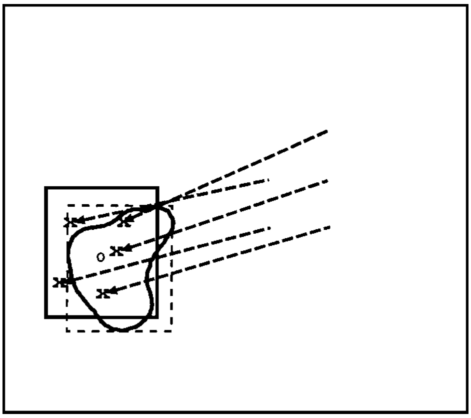

FIG. 7. Hits inside the decision area. ×, Valid hits; ○, center of ADA.

objects simultaneously (as will be demonstrated in the simulations section). The picture for which the object characteristics are determined is referred to as the *current picture*. The motion information is gathered with respect to the reference picture(s). The algorithm progresses sequentially through the pictures in the video sequence in coded order.

With each nonskipped macroblock in the current picture, motion vectors are provided that indicate the positions where the prediction samples should be read from the reference picture(s). We refer to the tips of the motion vectors originating in the current picture and pointing to the reference picture(s) as *hits*. Our interest is in those hits (if any) that will be used by the decoder to read samples from the reference picture(s) and that correspond to some region of the tracked object. An *adaptive decision area* (ADA) around the known object position in the reference picture is used to isolate the hits that are relevant to tracking the object of interest. Here it is assumed that the ADA has been computed and is available to facilitate the motion information acquisition. The hits that are located within the ADA around the target position are called *valid hits*. For now, let us assume that the ADA is rectangular, fixed in size and shape. An example of valid hits within the ADA is illustrated in Fig. 7. It should be noted that the ADA is not centered around the object's centroid due to the fact that the motion vectors have their origins at the upper left corners of macroblocks and not at their centers. Hence, the ADA is shifted up and to the left by eight pixels in each direction to account for this feature.

According to MPEG prediction methodology [31], nonslice macroblocks are not encoded. The decoder simply replicates the corresponding macroblocks from the reference picture. Figure 8 illustrates an example where the determined position of the object of interest in the

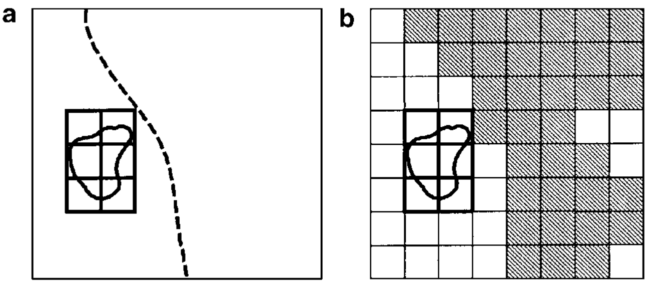

FIG. 8. Object outside slices. (a) Reference picture; (b) current picture.

reference picture is such that the object is located outside the slice structure in the current picture. Thus, the corresponding macroblocks in the current picture, being outside slices, would be copied from their counterpart macroblocks in the reference picture where the object is situated. Hence, if the above situation is detected, the motion filtering and analysis of the algorithm is not activated—the position of the object in the current picture should simply be copied from the reference picture.

The algorithm collects all the valid hits that are located inside the ADA in the reference picture(s). The motion vectors corresponding to the valid hits are acquired. The relevant macroblocks in the current picture may have various prediction types, depending on the current picture type (P or B). If the number of intramacroblocks detected in the ADA is small relative to the number of predicted macroblocks, these intramacroblocks are assigned zero motion vectors in both P and B pictures. This is done in order to allow the motion analysis stage to compute a conservative estimate of object motion, position, and ADA when intramacroblocks are involved. On the other hand, if the ratio of the numbers of intracoded macroblocks to motion compensated macroblocks detected in the ADA exceeds a specified threshold, this is an indication that the object area of the image has changed significantly to the degree where the object is lost or has severely changed its appearance. In this case, the algorithm returns to object acquisition mode (see Section 4) in order to provide for reacquisition of the target and thus tracking initialization.

5.2. *Classification of Motion Information*

Once the relevant motion information has been acquired, the occlusion detection stage of the algorithm is activated. We now assume the usage of a variable ADA as part of our tracking algorithm. The details of ADA determination are presented later in this section. Occlusion of the tracked object would disrupt the tracking process, since the system would not be able to discriminate between the motion information that belongs to the tracked object and that which belongs to the occluding object. This would introduce potentially erroneous information into the motion analysis module. Thus, we introduce a classification module to address this problem. This must be done prior to the filtering and motion analysis modules which must operate on the correct motion data for the tracked object.

The *classification module* is used for detecting the object motion cluster. Figure 9 illustrates the structure of the classification module, which consists of a clustering module that uses the K-means algorithm, and a module used for the identification of the tracked object cluster. The object motion cluster consists of the motion vectors that are relevant to the object being tracked.

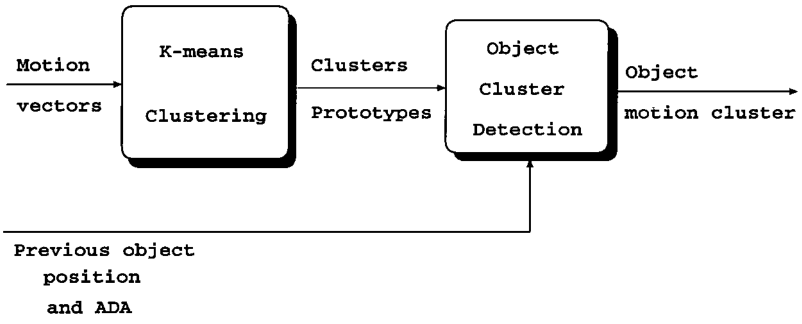

FIG. 9. Classification module.

5.2.1. K-means clustering. Let us consider an initial set X of n objects in a p-dimensional space,

$$X = \{\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}, \tag{1}$$

where

$$\overline{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{ip}\}. \tag{2}$$

A function that measures the quality of different clusterings of the same data set is used for the construction of a partition of the n points into a fixed number of K clusters. Clusters of points can be described by a centrally located object in the cluster, called a *representative object*. An alternative way of characterizing a cluster is through the use of its *centroid*. The centroid $\overline{x}$ of a cluster v is defined as a point in the p-dimensional space given by

$$\overline{x}(v) = \{x_1(v), x_2(v), \ldots, x_p(v)\}, \tag{3}$$

such that

$$x_j(v) = \frac{1}{n_v} \sum_{i \in C_v} x_{ij}, \tag{4}$$

where C_v is the set of indices of cluster v containing n_v objects.

One possible measure for the tightness of clusters is the sum of square errors defined as the sum of the squares of the Euclidean distances between the objects in a cluster and the cluster centroid. A solution to the clustering problem would be to seek a partition of the original set of points into K nonempty subsets that minimize the sum of squared errors for the entire clustering. Methods that operate based on this principle are called variance minimization clustering techniques. One of these methods is the K-means clustering algorithm [32].

We use the K-means clustering algorithm, applied to the acquired motion vectors, in order to determine the motion clusters and their prototypes (centroids) inside the ADA. Currently, we make the assumption that occlusion, if present, consists of the tracked object occluded by a different object in the image. Thus, we assume the possibility of two motion clusters and select $K = 2$. If the object was initially not occluded (when detected by the object recognition module), the tracked object will have different motion characteristics than those of the occluding object. The target and the occluding object will gradually be positioned inside the ADA of the tracked object when occlusion increases and separated when occlusion decreases. This is the information used by the clustering algorithm, whose role is to separate the target from the occluding object. The clustering process operates on the motion vector information that has been gathered in the ADA of the tracked object. The motion vectors are characterized by their magnitude and phase. The clusters are in turn represented by their corresponding centroids in terms of magnitude and phase, as produced by the K-means algorithm.

5.2.2. Object cluster detection. By computing the distance between the resulting clusters—the distance between cluster centroids—a decision about the presence or absence of occlusion is made. The distance between the motion clusters is compared to a threshold, which, if exceeded, signals occlusion. The threshold is formulated in terms of magnitude and phase, with magnitude representing the difference in motion speed between the target and occluding object and the phase representing the difference in movement direction. Furthermore, the most recent known position, ADA, and motion of the object from the reference picture are also available. Once occlusion is detected, the cluster that corresponds to the tracked object is identified as the cluster that yields the minimum distance with respect to the magnitude and phase between the generated motion centroids and the most recent object position update.

Normally, without the presence of occlusion, the motion filtering and analysis modules following the classification module would use the valid hits and motion information delimited by the target's ADA in the reference picture. However, if occlusion was detected and the target motion cluster was separated from the occluding object, only the correct subset of valid hits and motion vectors will be used for updating the information about the tracked object in the current picture. The position of the object in the reference picture will be recalculated using only the determined valid hit subset. Also, through a feedback mechanism, the original ADA that was previously computed in the reference picture will be modified to reflect the separation of the tracked and the occluding objects. The correct valid hit subset identified by the classification module is used to modify the previously computed ADA by using the coordinates of the outermost valid hits from the set to determine the boundaries of the new ADA. ADA recomputations based on occlusion detection feedback are illustrated in Fig. 10. The figure shows motion information situated in both the reference picture (valid hits and ADA) and the current picture (origins of motion vectors).

The classification stage has also the dual effect of filtering out isolated motion vector outliers, outside the context of occlusion detection.

5.3. Filtering of Motion Information

The motion compensation process takes place at the level of macroblocks in the picture. Thus, the origins of the motion vectors are situated on the grid determined by the macroblocks, resulting in a coarse spatial quantization. Also, the motion vectors are often

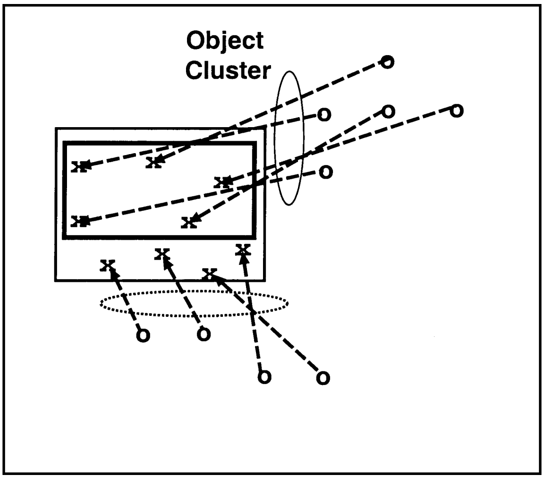

FIG. 10. Occlusion detection and feedback. ×, Valid hits; ○, origins of motion vectors in current picture; —, original ADA; —, Recomputed ADA.

noisy and can therefore be filtered in order to improve the quality of motion information that is used for tracking. We use the singular-value decomposition (SVD) and its properties for the purpose of filtering motion information. Given the fact that P pictures are used as further reference pictures, which is not the case for B pictures, and taking into account the picture coded order, the filtering is applied only for P pictures. The filtering approach can be extended also to B pictures, if the precision of the interpolation-based computation of object position and size in B pictures introduced later in this section are considered to be insufficient (see Section 5.4.1).

For the general case of a rectangular matrix, SVD is a decomposition that reduces it to diagonal form. This is obtained by premultiplying and postmultiplying the original matrix by two different orthogonal matrices. Given the original matrix **X** (data matrix) of size $L \times M$, there exist an $M \times M$ orthogonal matrix **V** and $L \times L$ orthogonal matrix **U** such that

$$\mathbf{U}^{\mathbf{T}}\mathbf{X}\mathbf{V} = \mathbf{\Sigma}, \tag{5}$$

where $\mathbf{\Sigma}$ is an $L \times M$ nonnegative diagonal matrix defined by

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{S} & 0 \\ 0 & 0 \end{bmatrix}, \tag{6}$$

and

$$\mathbf{S} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_W), \tag{7}$$

with

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_W > 0. \tag{8}$$

The diagonal elements of the decomposition, $\sigma_1, \sigma_2, \ldots, \sigma_W$ and $\sigma_{W+1} = \cdots = \sigma_M = 0$, are called the *singular values* of matrix **X**. The columns of **V** are called the *right singular vectors* of **X**, and the columns of **U** are called the *left singular vectors* of **X**.

Let us assume that the current picture is a P picture. As a result of motion acquisition we obtain a set of n motion vectors, corresponding to valid hits in the reference picture and with corresponding origins in the current picture. Let us denote this set of motion vectors by $d = \{\overline{l_1}, \overline{l_2}, \ldots, \overline{l_n}\}$. The values of the Cartesian motion components of $\overline{l_i}$ are denoted by $\{l_{z,i}, i = 1 \cdots n\}$, where z indexes the x and y directions.

We form the square data matrices $\mathbf{H_z}$ of size $n \times n$, using the corresponding vectors $\{l_{z,i}, i = 1 \cdots n\}$:

$$\mathbf{H_z} = \begin{bmatrix} l_{z,1} & 0 & 0 & \cdots & 0 \\ l_{z,2} & l_{z,1} & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \\ l_{z,n} & l_{z,n-1} & \cdots & & l_{z,1} \end{bmatrix}. \tag{9}$$

Thus, $\mathbf{H_z^T H_z}$ corresponds to the correlation matrix for vector $\{l_{z,1}, l_{z,2}, \ldots, l_{z,n}\}$. The SVD of the motion data matrix $\mathbf{H_z}$ is computed in order to determine its singular values and the right and left singular vectors. Using the notation introduced above, we have the following representation of the data matrix,

$$\mathbf{H_z} = \mathbf{U_z \Sigma_z V_z^T}, \tag{10}$$

where

$$\mathbf{\Sigma_z} = \begin{bmatrix} \mathbf{S_z} & 0 \\ 0 & 0 \end{bmatrix}, \tag{11}$$

with $\mathbf{S_z} = \mathrm{diag}(\sigma_{z,1}, \sigma_{z,2}, \ldots, \sigma_{z,r})$ and $\mathbf{U_z}$ and $\mathbf{V_z}$ having as columns the left and right singular vectors of $\mathbf{H_z}$, respectively.

The resulting vector of nonzero singular values $S_z = \{\sigma_{z,1}, \sigma_{z,2}, \ldots, \sigma_{z,n}\}$, where r is the rank of $\mathbf{H_z}$, is ordered in decreasing sequence. Given that the ordered set of nonzero singular values is characterized by the fact that the first few values are comparatively much larger than the others in the set, it is possible to discard the smaller singular values. Thus, we shall identify the first large gradient in the vector of ordered, nonzero singular values S_z and use it to retain only the subset of singular values that is significantly larger than the rest of the values in the set. Assuming that only the first $m(m < r)$ values are retained, we obtain a new vector S'_z:

$$S'_z = \{\sigma_{z,1}, \sigma_{z,2}, \ldots, \sigma_{z,m}, \ldots, 0, 0, \ldots, 0\}. \tag{12}$$

We shall now use the truncated set S'_z to construct a new diagonal matrix $\mathbf{\Sigma'_z}$ of size $n \times n$,

$$\mathbf{\Sigma'_z} = \begin{bmatrix} \mathbf{S'_z} & 0 \\ 0 & 0 \end{bmatrix}, \tag{13}$$

with $\mathbf{S'_z} = \mathrm{diag}(\sigma_{\mathbf{z,1}}, \sigma_{\mathbf{z,2}}, \ldots, \sigma_{\mathbf{z,m}})$, and $m < r$. The modified matrix of singular values $\mathbf{\Sigma'_z}$ is used together with the singular vector matrices $\mathbf{U_z}$ and $\mathbf{V_z}$ to obtain a new matrix $\mathbf{H'_z}$ given by

$$\mathbf{H'_z} = \mathbf{U_z \Sigma'_z V_z^T}. \tag{14}$$

Effectively, by retaining only certain nonzero singular values $\{\sigma_{z,1}, \sigma_{z,2}, \ldots, \sigma_{z,m}\}$, as we did above, only the corresponding singular vectors will play a role in the computation of $\mathbf{H'_z}$, as it is observed by examining Eq. (14). The first column of matrix $\mathbf{H'_z}$ gives the *filtered* set of motion vectors $\{l'_{z,1}, l'_{z,2}, \ldots, l'_{z,n}\}$ that can now be used by the motion analysis module. Figure 11 shows both the original motion vectors and their filtered versions along with their corresponding new origins.

5.4. *Motion Analysis*

After the clustering and filtering stage, we proceed with the motion analysis in order to determine the new position and ADA of the tracked object in the current picture. Depending on whether the previous filtering module is activated or not, the motion analysis module operates using filtered or unfiltered motion information for P pictures. The diagram of the motion analysis block is shown in Fig. 12.

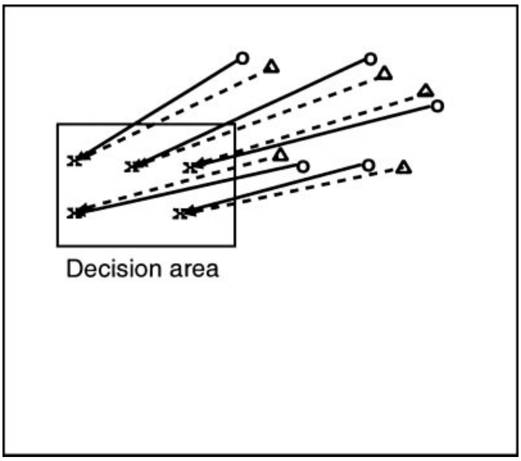

FIG. 11. Motion vector filtering. ○, Origins of unfiltered motion vectors; △, Origins of filtered motion vectors.

5.4.1. Object position estimation. For P pictures, the position estimation operates based on the data obtained as a result of the classification and filtering modules (the latter, if filtering is activated). Thus, the motion vectors that are used next can be the original ones as extracted from the bit stream or, if filtering is used, their filtered versions. For B pictures, the position interpolation operates by using the original motion vectors as recovered from the bit stream. For simplicity, we shall consider the case of tracking a single object. As mentioned in the Introduction, the algorithm has the capability to track multiple objects simultaneously. We shall also restrict our analysis to the case of rigid objects; i.e., we assume that the objects do not disintegrate. Image Cartesian positions are denoted by $\bar{p} = [x \quad y]^T$ and vectors by $\bar{l} = [l_x \quad l_y]^T$. Next, it is assumed that the latest position of the object in the reference picture is such that the object overlap with the slices area exceeds a specified threshold. In other words, it is suitable for the tracking component of the algorithm to be activated.

If the current picture is a P picture, let us assume that the object is present in the reference picture at some position that we shall denote by $\bar{p}_{\text{ref}}$. This position might have been

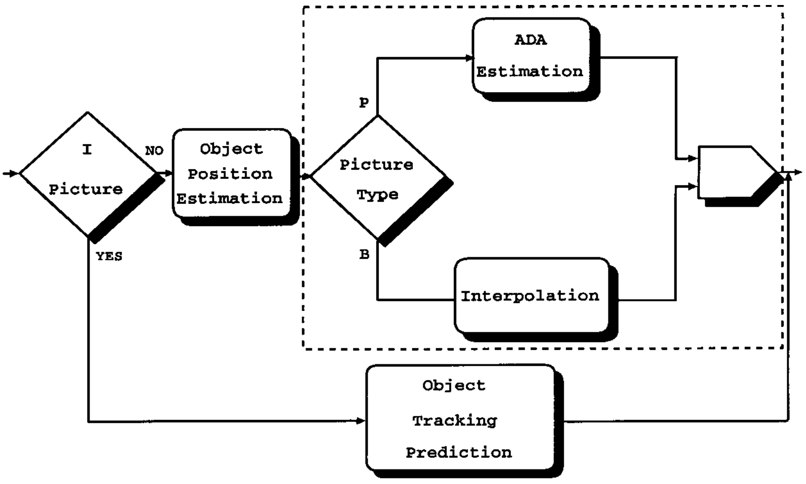

FIG. 12. Motion analysis.

determined as a result of an object recognition method, if the reference picture is a selected I picture, or as a result of a previous step in our tracking algorithm. The valid hits and the corresponding motion vectors are processed by the classification and filtering modules as described in the previous sections. An estimate the motion of the object $\overline{l}_{\text{est}}$ with respect to $\overline{p}_{\text{ref}}$, with the motion vectors being designated by $\overline{l}_i$, is given by

$$\overline{l}_{\text{est}} = \frac{1}{n}\sum_{i=1}^{n}\overline{l}_i, \tag{15}$$

where n is the number of valid hits in the reference picture (including relevant skipped and intramacroblocks). Therefore, the new position of the object in the current P picture $\overline{p}^{P}_{\text{new}}$ can be determined based on its known position in the reference picture and the estimated motion information:

$$\overline{p}^{P}_{\text{new}} = \overline{p}_{\text{ref}} + \overline{l}_{\text{est}}. \tag{16}$$

For a B picture, let us denote by $\overline{p}^{f}_{\text{ref}}$ the detected position of the target in the previous to most recent reference picture (which corresponds to forward prediction) and by $\overline{p}^{b}_{\text{ref}}$ the detected position of the target in the most recent reference picture (which corresponds to backward prediction). Therefore, for B pictures, we have two known reference positions $\overline{p}^{f}_{\text{ref}}$ and $\overline{p}^{b}_{\text{ref}}$, depending on the type of macroblock prediction. The algorithm also counts the number of forward and backward valid hits in two variables n_f and n_b, respectively. We estimate the object motion with respect to its position in the forward and backward reference pictures by computing the corresponding displacements $\overline{l}^{d}_{\text{est}}$,

$$\overline{l}^{d}_{\text{est}} = \frac{1}{n_d}\sum_{i=1}^{n_d}\overline{l}_i, \tag{17}$$

where d represents the indices f or b, depending on the prediction direction considered, and n_f and n_b are the numbers of forward and backward motion vectors (including the case of skipped and intramacroblocks). We now compute two intermediary new positions of the object in the current picture based on its estimated motion relative to the reference positions

$$\overline{p}^{d}_{\text{new}} = \overline{p}^{d}_{\text{ref}} + \overline{l}^{d}_{\text{est}}. \tag{18}$$

Next, we use $\overline{p}^{f}_{\text{new}}$ and $\overline{p}^{b}_{\text{new}}$ to compute a weighted update of the object position $\overline{p}^{B}_{\text{new}}$ in the current picture based on motion information relative to both reference pictures,

$$\overline{p}^{B}_{\text{new}} = (1/n)(n_f\,\overline{p}^{f}_{\text{new}} + n_b\,\overline{p}^{b}_{\text{new}}), \tag{19}$$

where $n = n_f + n_b$.

Figure 13 illustrates the process of updating the object position in nonselected I pictures. The position of the object is computed by using a one-step prediction based on the most recent tracking information from the previous P picture (magnitude and direction).

5.4.2. Adaptive decision area estimation. The necessity of using an adaptive decision area, due to valid hit distribution in the reference picture, is connected to the issue of scaling and rotation of the tracked object in the image. It must be assumed that the object undergoes

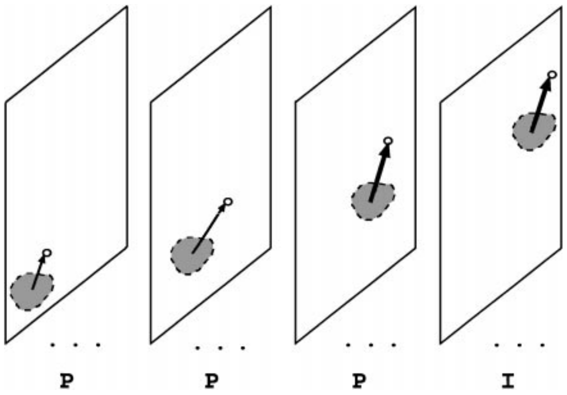

FIG. 13. Tracking prediction in I pictures. ○, Computed object position in current picture.

some degree of change in appearance from frame to frame, due to scaling, rotation, and possible occlusion. Attempting to track the object with a fixed ADA in conditions of affine transformations would mean that gradually only areas of the object will remain inside the ADA (e.g., object scale increase), or additional macroblocks that may belong to the background or other objects will be tracked along with the target (e.g., decrease in object scale). Eventually, the track might be degraded or lost. Thus, we must estimate the ADA associated with the tracked object as part of the motion analysis. Figure 14 portrays a situation where the mean of the valid hits inside the ADA is about the same in (a) and (b)—approximately equal to the corresponding spatial position of the center of ADA. However, the variances of the two sets of valid hits are very different—the valid hits in (a) are widespread compared to the concentrated valid hit distribution in (b).

To estimate the ADA, we differentiate between the current picture type (B, P, or I). B pictures are not used for further prediction as reference pictures. Therefore, the ADA for B pictures can be interpolated using the two known ADAs that have already been computed in its two reference pictures (I or P). For P pictures, where the determined ADA will be used for further object position estimation, we use a statistical temporal analysis of the local motion information. In I pictures, the dimensions of the ADA are the same with those of the ADA computed in the previous P picture.

Figure 15 illustrates the computation and use of the ADA for acquiring the valid hits. Let us assign to the current picture an index k in the picture sequence and to its reference picture index $k - 1$. For simplicity, k is used here to index only pictures that may be used as reference by the motion compensation process; i.e., I or P pictures. It is possible that there are additional B pictures placed between any two consecutive I or P pictures. Let us

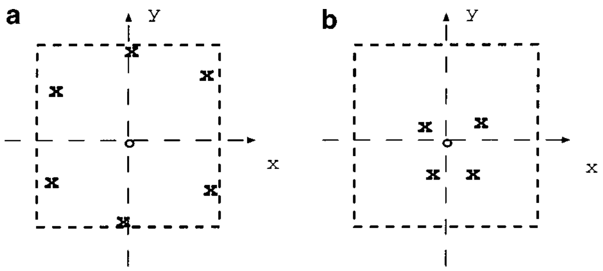

FIG. 14. Hit distribution inside the decision area. ○, ADA center; ×, Valid hits.

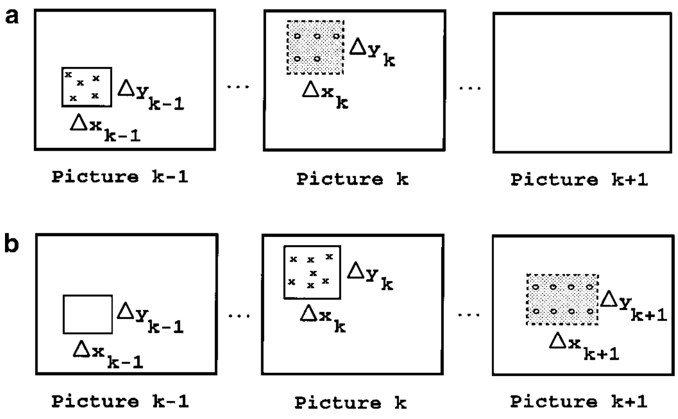

FIG. 15. Adaptive decision area. (a) Computation; (b) valid hit acquisition. $\times$, Valid hits; $\bigcirc$, origins of motion vectors.

assume that the ADA in picture $k-1$ is known and it is used to collect hits that are to be used in computing the new object position in the current picture k. All the valid hits in picture $k-1$ have corresponding motion vectors whose origins are in the current picture k (see Fig. 15a). At a later time, the current picture k will become a reference picture for the following P picture having index $k+1$ (see Fig. 15b). Thus, we shall compute an ADA in the current picture k that will be used to collect the valid hits used for updating the object attributes (position and size) in picture $k+1$. This ADA must take into account the information provided by the distribution of the origins of the motion vectors in picture k, corresponding to the valid hits in picture $k-1$.

In the following development we shall operate separately on the horizontal and vertical dimensions of the ADA, thus allowing for different changes in the x and y directions. In the following equations, z is used to represent x and y, corresponding to the two axes of coordinates. Let Δx_k and Δy_k be the dimensions for the ADA that must be computed in picture k, which will serve for determining the valid hits corresponding to motion vectors originating from picture $k+1$. Let us also denote by Δx_{k-1} and Δy_{k-1} the dimensions of the most recent, known ADA, which is present in reference picture $k-1$. Therefore,

$$\Delta z_k = \Delta z_{k-1} + \delta z_k, \tag{20}$$

where δz_k represents the changes that the ADA undergoes in the x and y directions.

The computation of Δx_k and Δy_k is based on the distribution of the origins $\{x_i^{\mathrm{o}}, y_i^{\mathrm{o}}\}$ of the motion vectors, corresponding to valid hits located in picture $k-1$. The origins of the motion vectors that are used to determine the new ADA in the current picture may correspond to the unfiltered motion vectors or to the SVD-based filtered motion vectors as discussed in Section 5.3. We also take into account the size of the previous decision are in picture $k-1$, and we use the results of a statistical analysis of the local motion information over $M+1$ previous P reference pictures. For this purpose, we compute the mean and standard deviation of the origins of the motion vectors, in each of the $M+1$ reference pictures. The reason for using only P pictures is that I pictures are intracoded and thus have no motion compensation information. The mean and standard deviation of $\{x_i^{\mathrm{o}}\}$ and $\{y_i^{\mathrm{o}}\}$ in picture k, denoted respectively by $m_z^{\mathrm{o}}(k)$ and $\sigma_z^{\mathrm{o}}(k)$, are computed as follows,

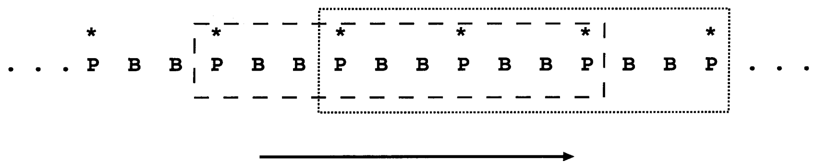

FIG. 16. Moving temporal average of object motion statistics.

$$m_z^o(k) = \frac{1}{n}\sum_{i=1}^{n} z_i^o, \tag{21}$$

and

$$\sigma_z^o(k) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[z_i^o - m_z^o(k)\right]^2}, \tag{22}$$

where n is the number of origins of motion vectors in picture k and z_i^o are the coordinates of those origins. Therefore, δz_k can now be computed,

$$\delta z_k = h\left\{sK_{\mathrm{ADA}}\left(\nabla\sigma_z^o(k)\right)\frac{\sigma_z^o(k)}{\sigma_z^o(k-1)}\right\}, \tag{23}$$

where $\nabla\sigma_z^o(k)$ is given by

$$\nabla\sigma_z^o(k) = \frac{1}{M}\sum_{j=k-(M-1)}^{k}\left|\sigma_z^o(j) - \sigma_z^o(j-1)\right|, \tag{24}$$

s is the sign of the ADA variation, and K_{ADA} is a scaling constant that controls the rate of variation of the ADA. The sign s is determined based on the difference between the variances of the origins of valid motion vectors in the current picture and the reference picture; i.e., $s = \mathrm{sgn}[\sigma_z^o(k) - \sigma_z^o(k-1)]$. Figure 16 illustrates the computation of the average one-step temporal *gradient* of the standard deviations of motion vector origins $\nabla\sigma_z^o(k)$, by using a moving average across $M+1$ reference pictures (P).

The *limiting function* $h(.)$ is depicted in Fig. 17. The threshold ϵ is used in the process of computing the ADA to eliminate very small variations that are due to noise and numerical

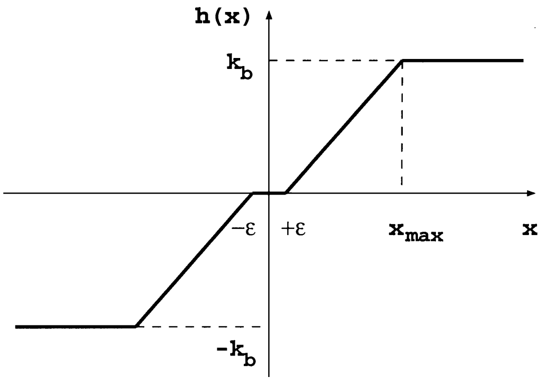

FIG. 17. Limiting function $h(.)$.

artifacts. However, the threshold ϵ must allow for very slow modification in the size of the object and, thus, of the ADA. Also, the ADA update δz_k is not allowed to vary excessively and, therefore, the limiting function $h(.)$ is upper bounded by the *saturation parameter* k_b.

Correct tracking of the object of interest under conditions of affine transformations is ensured through the use of the ADA. The ADA allows the system to account for changes in the appearance of the object and to dynamically add or eliminate macroblocks to the tracking process, based on the behavior of the target in the video sequence.

6. SIMULATION RESULTS

VORTEX was implemented using UNIX C on a SUN SPARC 20, under Solaris 2.5.1. The implementation was superimposed on top of an MPEG-2 coder/decoder available on the Internet, courtesy of MPEG Software Simulation Group.

In the first phase, simulations were conducted in order to assess the single and multiple tracking quality and the execution speed of VORTEX. These simulations were initially run for the case of translation and limited scale variation of the tracked objects, with no occlusion detection. This restriction facilitated the use of template matching in order to automatically and exhaustively detect the positions of the objects in the image. The object positions ascertained from template matching were used as reference positions for the VORTEX simulations. Moving objects present in the scene were selected as image templates for tracking simulation. These objects are marked with black or white frames in the sample images provided. The positions of the objects of interest were initialized in the first I picture of the video sequence. From there on, VORTEX was used to process the video for tracking of the two objects. Table 1 shows the Euclidean positional deviation of VORTEX from the reference positions reported by exhaustive template matching applied to decompressed data. The positional deviation values reported inevitably incorporate some spurious errors made by the exhaustive template matching due to minor affine variations of the desired objects. A subjective evaluation of the tracking simulations confirms that VORTEX is more accurate than exhaustive template matching.

Figure 18 illustrates the multiple target tracking capabilities of the VORTEX system in the case of two objects of interest (cow and train). The multiple object tracking feature is essential in the case of retrieval based on the simultaneous occurrence of multiple objects in the scene or the response to queries based on the spatial relationship of the objects in the image.

In Table 2, we present a comparison of the execution times of the exhaustive template matching on decompressed video data and the VORTEX algorithm, for the two selected objects (cow and train). The execution times of VORTEX are insignificant compared to those of the exhaustive template matching applied to decompressed video data. We note that these execution times include not only VORTEX's processing time but also the execution time of the required MPEG decoder operations such as decoding picture type, macroblock type,

TABLE 1
Euclidean Positional Deviation from Exhaustive Matching

Object	Mean	Standard deviation
Object #1 (cow)	3.52	1.58
Object #2 (train)	6.49	3.17

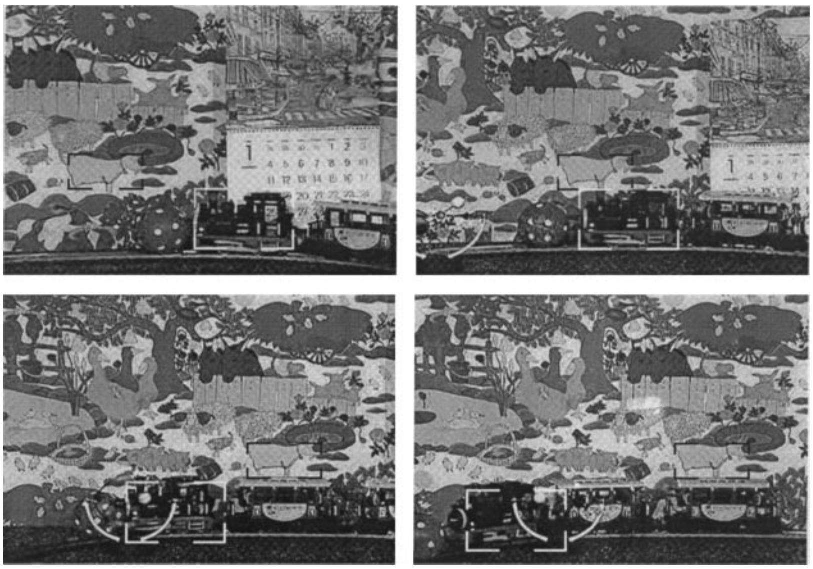

FIG. 18. Snapshots of video tracking for retrieval of multiple templates (cow—black frame and train—white frame).

and prediction type. The MPEG decoder used in our simulations is not optimized for speed. On the same machine (SPARC 20) and with code streamlining and speed optimization, the VORTEX execution time should be further improved. The second phase of testing involved tracking of objects affected by image transformations that include affine transformations (i.e., translation, scaling, and rotation). For the statistical analysis that contributes to the determination of the ADA in a current P picture, a number $M = 4$ past reference P pictures was used. Figures 19 and 20 illustrate the operation of the algorithm under affine transformations.

The video sequence depicted in Fig. 19 is characterized mostly by appearance changes and a small translation of the object of interest. The tracking is successful and provides the necessary variation in the ADA dimensions.

In Fig. 20, tracking was tested in conditions of severe 3D appearance variation of the object, in terms of translation, scale, and rotation transformations. The object is tracked successfully and the ADA is modified accordingly. As part of the object moves out of the picture in the last frames, the still visible area of the target continues to be tracked. The degree of variation of the ADA can be changed by modifying parameters such as the slope of the limiting function $h(.)$ (see Fig. 17) or by scaling δ_z in Eq. (23) (see Section 5.4.2).

In some cases, the quality of the motion compensation encoding is poor and thus the motion vectors are very noisy. Prefiltering of motion data can help the subsequent operations. Filtering is applied as described in Section 5.3. Both sets of images in Figs. 21a and 21b

TABLE 2
Average Execution Time per Picture

Template size	Exhaustive matching	VORTEX
33 × 68 (Object #1)	45.22 s	0.0084 s
41 × 77 (Object #2)	39.36 s	0.0092 s

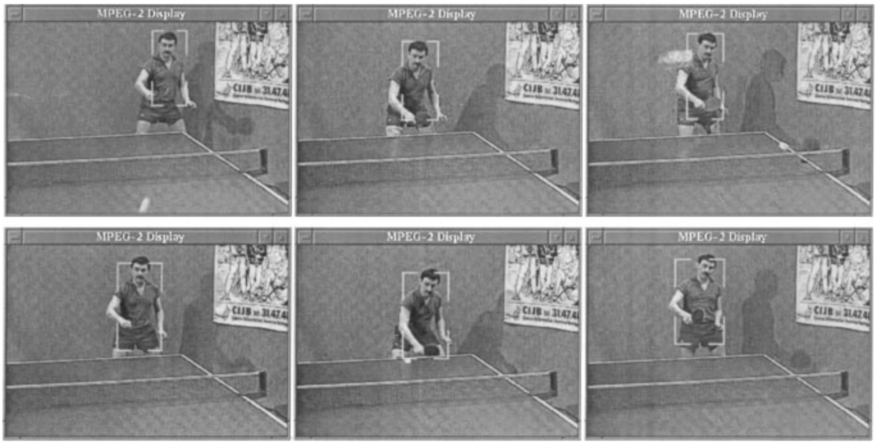

FIG. 19. VORTEX: Affine transformations 1.

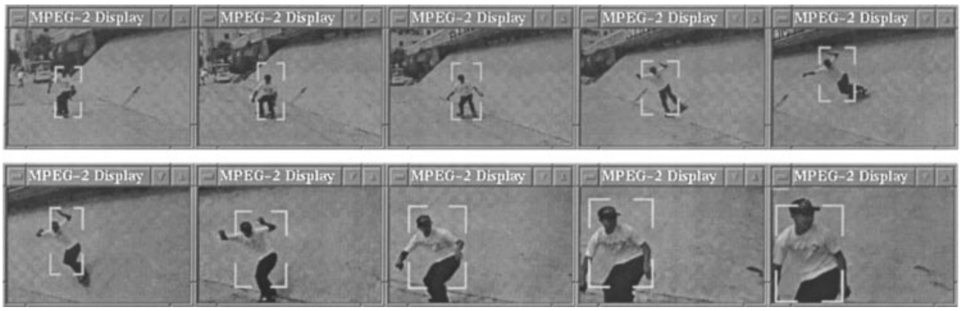

FIG. 20. VORTEX: Affine transformations 3.

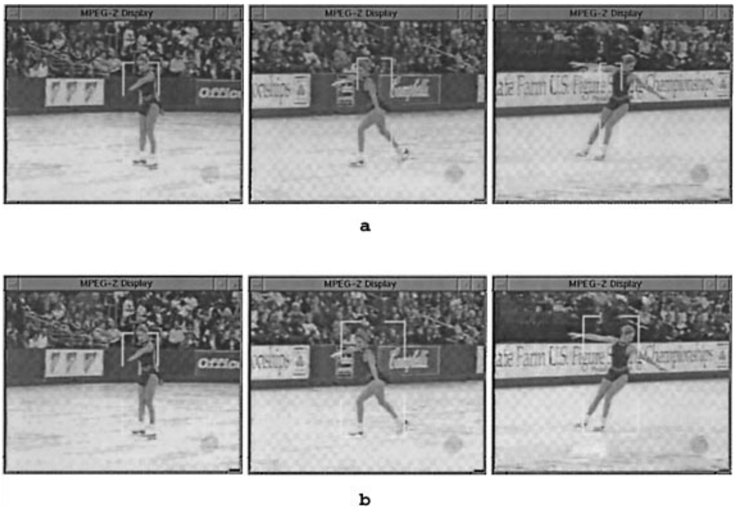

FIG. 21. Motion vector filtering.

have the same temporal position in the video sequence. The prefiltering of motion vectors improves the quality of tracking as illustrated in Fig. 21b with filtering active.

The third phase of simulations consisted of testing the occlusion detection capability of VORTEX. A K-means clustering algorithm with $K = 2$ was used for detection of the tracked and the occluding objects. The decision to accept two motion clusters as indicative of occlusion or severe noise in the area of the object is done by considering the distance between the corresponding magnitudes and phases of the two cluster prototypes. For this purpose, we used a distance threshold in both magnitude and phase for the decision process. The phase difference threshold parameter was set to 80° and the magnitude difference threshold parameter to 50%. Thus, if the two motion prototypes differ in motion direction by more than 80° and the relative difference in magnitudes is more than 50%, the algorithm decides in favor of two clusters as opposed to one. This in turn triggers the reassessment of the motion and ADA data in the reference picture(s) and thus adjusts the tracking of the video sequence, as discussed in Section 5.2. The parameters that control the cluster separation of the motion prototypes as well as the selection of the correct motion cluster are adjustable. The two image rows in Fig. 22 contain snapshots of the operation of the algorithm without occlusion detection depicted in Fig. 22a (top row) and with occlusion detection depicted in Fig. 22b (bottom row) for a video sequence. The video snapshots in the two rows represent images with the same index in the identical video sequence, in order to illustrate the effect that occlusion detection has on the tracking quality. The detection of both increasing and decreasing occlusion is determined by the way the cluster-discriminating threshold parameters are set. In general, it is acceptable that the object of interest be tracked even though some of its area is partially occluded by another object. This situation occurs in the case of increasing partial occlusion. Thus, for some interval of time, both objects are tracked together, with the occluding object covering some of the ADA area. However, detection of decreasing occlusion and separation of the target motion cluster is critical.

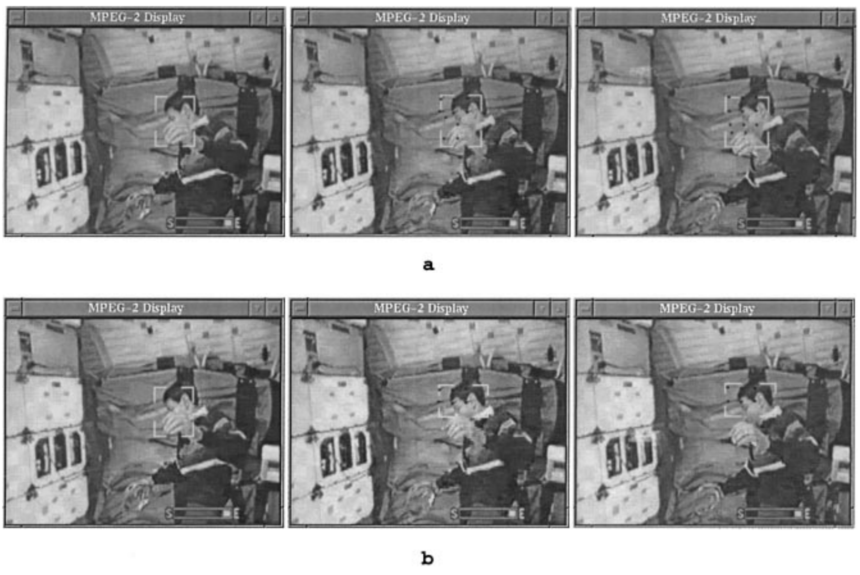

FIG. 22. VORTEX: Occlusion detection.

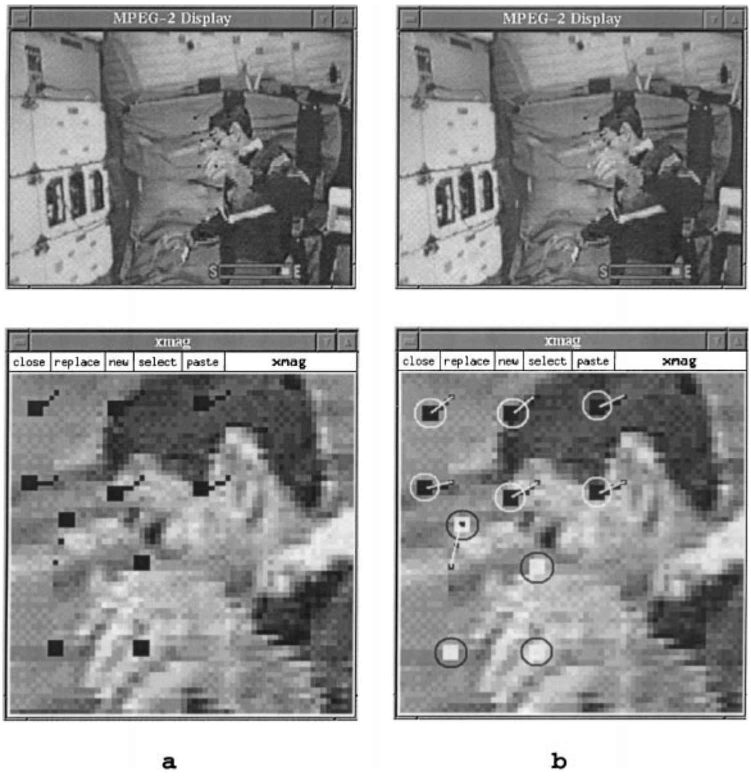
MPEG-2 Display
MPEG-2 Display
xmag
close replace new select paste xmag
xmag
close replace new select paste xmag
a
b

FIG. 23. VORTEX: Occlusion detection (magnified).

This is the case when the occluding object begins to move apart from the tracked object. In the images in Fig. 22, the tracked object is the head of the person in the image, and the occluding object is the hand that moves away from the face. The objective is to separate and discard the motion cluster associated with the hand, since it would introduce erroneous information that can further propagate in the video sequence. With occlusion detection, the correct motion cluster is correctly identified and the necessary adjustments executed. Without occlusion detection, the subsequent tracking quality will be degraded due to the incorporation of incorrect motion information corresponding to the occluding object (hand) in the tracking process.

The above considerations are further illustrated in Fig. 23, which provides magnified images of the area corresponding to the tracked object and the clustering process. The circled black and white squares (2×2 pixels), represent valid hits actually situated in the reference picture (not shown). Motion vectors are also shown (pixel approximation of lines) whose tips are situated at valid hits in the reference picture and whose origins are located on the macroblock grid in the current picture (shown). Thus, the images in Figs. 22 and 23 really show compound motion data from both the current picture being displayed and the reference picture. Precisely, the computed decision area and the origins of motion vectors are located in the current picture, while the valid hits (circled squares) represent positions in the reference picture. As it can be seen in Fig. 23a, without classification and filtering, the motion data which belong to the occluding object (hand) will be incorporated in the subsequent tracking process. However, in Fig. 23b, with the classification and filtering, the occluding object motion information—valid hits and motion vectors—is identified and discarded (white valid hits and associated data). This results in the recomputation of the ADA in the reference picture, which will influence the computation of the current ADA. As seen in Fig. 22b, as a result of motion data classification, the current ADA is reduced in size in order to prevent the inclusion of incorrect motion information in the subsequent tracking process. Simulations were run using MPEG-2 bit streams encoded at 1.5 Mbps. This rate is situated toward the lower end of the range, for applications intended for MPEG-2. The quality of motion information included in the video bit stream is determined by the precision of the prediction process (i.e., motion compensation). Thus, the quality of tracking of the VORTEX system, which depends on the reliability of the prediction by the encoder, will improve with higher data rates. The simulations presented in this section can therefore be regarded as a lower bound on the performance of the system.

7. CONCLUSIONS

In this paper, we present a novel video retrieval and tracking system—VORTEX—that operates on the compressed video data provided by MPEG video sequences. Aside from standard queries that seek to find relevant video sequences based on the occurrence of one or more objects in the scene, logical queries based on relative object positions in images are supported. VORTEX provides points of access to the query-relevant video sequences present in a video archive, such as to enable video playback by the user.

In order to facilitate tracking of possibly multiple objects through the video sequence, the first stage of the algorithm involves selective acquisition of motion information relevant to the tracked objects. This information is further classified in order to enable detection of possible occluding objects or elimination of spurious motion data outliers. An SVD-based analysis of the target motion information is performed by the filtering module. The filtered data are then utilized by the motion analysis module, which estimates both the position and the size represented by the bounding rectangle of the objects of interest. The efficiency of the entire process is ensured by decoding only the minimum information necessary for retrieval and tracking.

Although the presentation has been restricted to the MPEG-2 video compression standard, the basic approach should remain valid across the spectrum of compression schemes that exploit motion compensation (e.g., MPEG-4).

The premise of this work is not restricted to video archives: image databases consisting of similar images such as medical or satellite data can be compressed efficiently using video compression standards. Applications of our approach to video retrieval and tracking to compressed (lossless) tomographical images in medical databases are envisioned.

Future extensions of the proposed approach will focus on real-time object retrieval from compressed video databases. This task will once again be based on the use of motion vectors for the efficient segmentation of scenes in compressed video archives.

REFERENCES

1. B. Cahoon and K. S. McKinley, Performance evaluation of a distributed architecture for information retrieval, *ACM SIGIR International Conference on Research and Development in Information Retrieval, 1996*, pp. 110–118.
2. J. Ashley, R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic, Automatic and semiautomatic methods for image annotation and retrieval in query by image content (QBIC), *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases, 1995*, pp. 24–35.
3. D. Saur, Y.-P. Tan, S. Kulkarni, and P. Ramadge, Automated analysis and annotation of basketball video, *SPIE Conference on Storage and Retrieval for Image and Video Databases, 1997*, pp. 176–187.
4. M. Cherfaoui and C. Bertin, Temporal segmentation of videos: A new approach, *SPIE* **2419**, 1995, 38–47.
5. A. Hampapur, R. Jain, and T. Weymouth, Digital video segmentation, *Proc. of the ACM Multimedia Conference and Exposition, 1994*, pp. 357–364.
6. N. Patel and I. Sethi, Video segmentation for video data management, in *The Handbook of Multimedia Information Management* (W. Groski, R. Jain, and R. Mehrotra, Eds.), pp. 139–165, Prentice Hall, New York, 1997.
7. V. Vinod and H. Murase, Video shot analysis using efficient multiple object tracking, *Proceedings of IEEE International Conference on Multimedia Computing and Systems, 1997*, pp. 501–508.
8. V. Kobla, D. Doermann, and K. I. Lin, Archiving indexing and retrieval of video in the compressed domain, *Proc. of the SPIE Conference on Multimedia Storage and Archiving Systems, 1996*, Vol. 2916, pp. 78–89.
9. J. Meng, Y. Juan, and S.-F. Chang, Scene change detection in a MPEG compressed video sequence, *IS&T/SPIE Symposium Proceedings, 1995*, Vol. 2419.
10. K. Shen and J. Delp, A fast algorithm for video parsing using MPEG compressed sequences, *IEEE Conference on Image Processing, 1995*, pp. 252–255.
11. H. J. Zhang, C. Y. Low, and S. W. Smoliar, Video parsing and browsing using compressed data, *Multimedia Tools Appl.* **1**, 1995, 89–111.
12. B. L. Yeo and B. Liu, A unified approach to temporal segmentation of motion JPEG and MPEG compressed video, *IEEE International Conference on Multimedia Computing and Systems, May 1995*.
13. N. Dimitrova and F. Golshani, Video and image content representation and retrieval, in *The Handbook of Multimedia Information Management* (W. Groski, R. Jain, and R. Mehrotra, Eds.), pp. 95–138, Prentice Hall, New York, 1997.
14. J. Meng and S.-F. Chang, Tools of compressed domain video indexing and editing, *SPIE Conference on Storage and Retrieval for Image and Video Databases, 1996*, Vol. 2670.
15. J. Meng and S.-F. Chang, CVEPS—A compressed video editing and parsing system, *ACM Multimedia Conference, 1996*.
16. J. R. Smith and S.-F. Chang, VisualSEEk: A fully automated content-based image query system, *ACM Multimedia Conference, 1996*.
17. V. Kobla and D. Doermann, VideoTrails: Representing and visualizing structure in video sequences, *Proc. of the ACM Multimedia Conference, 1997*, pp. 335–346.
18. C. Faloutsos and K. Lin, FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets, *Proc. of the ACM SIGMOD Conference, 1995*, pp. 163–174.
19. V. Kobla and D. Doermann, Indexing and retrieval of MPEG compressed video, *J. Electronic Imaging* **7**, 1998, 294–307.
20. Y. Park, F. Golshani, and S. Panchanathan, Interactive classification and indexing of still and motion pictures in VideoRoadMap, *Proc. of the SPIE Conference on Multimedia Storage and Archiving Systems III, 1998*, pp. 122–133.

21. S. Panchanathan, Y. C. Park, and F. Golshani, VideoRoadMap: A system for interactive classification and indexing of still and motion pictures, *IEEE Instrumentation and Measurement Technology Conference, May 1998*.
22. F. Golshani, K. S. Candan, S. Panchanathan, and Y. C. Park, VIMOS: A video mosaic for spatio-temporal representation of visual information, *IEEE South-West Symposium on Image Analysis and Interpretation, 1998*.
23. F. Idris and S. Panchanathan, Review of image and video indexing techniques, *J. Video Commun. Image Representation–Special Issue on Indexing, Storage and Retrieval of Images and Video,* June 1997.
24. M. Yeung and B. Yeo, Video content characterization and compaction for digital library applications, *Proc. of the SPIE Conference on Storage and Retrieval for Still Image and Video Databases, 1997*, Vol. 3022, pp. 45–58.
25. J. Guo, J. Kim, and C.-C. Jay Kuo, Fast video object segmentation using affine motion and gradient-based color clustering, *IEEE Workshop on Multimedia Signal Processing, December 1998*.
26. D. Schonfeld and D. Lelescu, VORTEX: Video retrieval and tracking from compressed multimedia databases, *Proceedings of IEEE International Conference on Image Processing, October 1998*.
27. D. Schonfeld and D. Lelescu, VORTEX: Video retrieval and tracking from compressed multimedia databases—Template matching from MPEG-2 Video Compression Standard, *Proceedings of SPIE Conference on Multimedia Storage and Archiving Systems III, 1998*, Vol. 3527, 233–244.
28. D. Schonfeld and D. Lelescu, VORTEX: Video retrieval and tracking from compressed multimedia databases—Visual search engine, *32nd Annual International Conference on System Sciences (HICSS), January 1999*.
29. D. Schonfeld and D. Lelescu, VORTEX: Video retrieval and tracking from compressed multimedia databases—Affine transformation and occlusion invariant tracking from MPEG-2 Video, *Proceedings of SPIE Conference on Multimedia Storage and Archiving Systems VII, January 1999*.
30. V. Kobla, D. Doermann, and C. Faloutsos, Developing high-level representations of video clips using Video-Trails, *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases VI, 1998*, pp. 81–92.
31. International Standard ISO/IEC 13818-2 1995(E), *Generic Coding of Moving Pictures and Associated Audio Information: Video.*
32. B. Everitt, *Cluster Analysis*, third ed., Arnold, Sevenoaks, 1993.

DAN SCHONFELD was born in Westchester, Pennsylvania, on June 11, 1964. He received the B.S. in electrical engineering and computer science from the University of California, Berkeley, California, in 1986 and the M.S. and Ph.D. in electrical and computer engineering from the Johns Hopkins University, Baltimore, Maryland, in 1988 and 1990, respectively. Since September 1990 he has been with the Department of Electrical Engineering and Computer Science at the University of Illinois, Chicago, Illinois, where he is currently an associate professor. He has authored over 30 technical papers in various journals and conferences. He has served as an associate editor of the *IEEE Transactions on Image Processing on Nonlinear Filtering*. He currently serves as an associate editor of the *IEEE Transactions on Signal Processing on Multidimensional Signal Processing and Multimedia Signal Processing*. His current research interests are in one- and multidimensional signal processing, speech processing, image processing, video processing, computer vision, pattern recognition, medical imaging, and multimedia communication networks.

DAN LELESCU was born in Lugoj, Romania, on June 3, 1967. He received the B.S./M.S. in electrical engineering from the Technical University “Politehnica” of Timisoara, Romania, in 1991. Currently, he is a Ph.D. candidate in the Department of Electrical Engineering and Computer Science at the University of Illinois, Chicago, Illinois. His research areas of interest include multimedia signal processing and communications, signal detection and estimation, image and video processing, and computer vision.