An Approach to Trajectory Estimation of Moving Objects in the H.264 Compressed Domain

Christian Kaes, Henri Nicolas

An Approach to Trajectory Estimation of Moving Objects in the H.264 Compressed Domain

Christian Käs and Henri Nicolas

LaBRI, University of Bordeaux,
351, cours de la libération, 33405 Talence, France
{kaes,nicolas}@labri.fr
http://www.labri.fr

Abstract. This paper presents a simple and fast method for unsupervised trajectory estimation of multiple moving objects within a video scene. It is entirely based on the motion vectors that are present in compressed H.264/AVC or SVC video streams. We extract these motion vectors, perform robust frame-wise global motion estimation and use these estimates to form outlier masks. Motion segmentation on the spatio-temporally filtered outlier masks is performed to detect moving regions in the scene, which are analyzed over time in order to identify similar objects in adjacent frames. The construction of so-called Object History Images (OHIs) is proposed to stabilize the trajectories, which are finally interpolated with X-splines. The system enables real-time analysis with standard hardware.

Key words: Scene Analysis, Trajectory estimation, H.264-AVC/SVC compressed domain

1 Introduction

The detection and tracking of moving objects in video scenes is an interesting and challenging research topic. Possible applications of such algorithms include video surveillance, retrieval tasks and scene analysis. Video processing tasks working at pixel level are usually computationally very expensive.

We aim at providing a method for efficient and fully automatic trajectory estimation of multiple objects, that is applicable to scalable state-of-the-art streams encoded by H.264/SVC, without implying any constraints on the nature of the moving objects. We assume that we have separated video scenes without any cuts or transitions. This can be achieved by first applying a compressed domain shot boundary detector, one of which was proposed by Bruyne [1] specifically for H.264 streams.

In general, object tracking in the pixel domain is more robust and performs better than compressed domain methods, since more and more precise information is available. Nevertheless, the motivation for compressed domain analysis

remains and is driven by fast processing speed and the fact that videos are primarily stored in compressed form. Faster processing becomes possible due to the fact that motion information is already present in the stream. Decreased robustness of motion-based, compressed domain approaches usually results from the noisy nature of the motion vectors, which are optimized in terms of coding efficiency and represent a sparse and noisy version of the real optical flow.

2 Related Work

A large number of compressed domain object segmentation and tracking algorithms appeared over the years. Some publications concerning pure object *segmentation* in the MPEG domain include [2–6].
Babu et al. [2, 3] proposed an accumulation of motion vectors (MVs) over time, followed by a K-Means clustering to determine the number of objects in the scene and the EM algorithm for object segmentation. Zeng et al. [4] employ a block-based Markov Random Field (MRF) model to segment moving objects from the sparse MV field, which is extracted from H.264 compressed streams. The proposed method is limited to static cameras.

The proposed tracking approaches in the compressed domain rely either on MVs, residual information, or both. A lot of these works exploit the information found in MPEG-1/2 streams, where MVs and DCT coefficients are easily accessible. Hesseler et al. [7] perform the tracking initialization on decoded I-frames and use histograms of MVs of the MPEG-2 stream to perform tracking. The method does not support rotating objects and changes in size. Lie et al. [8] proposed a system that tracks single macro-blocks (MBs) under consideration of residual information. Trajectories are afterwards merged to obtain a moving object segmentation. Other MPEG-2 based methods have been proposed in [9–15].

Though most of the mentioned work can generally be ported to the H.264-AVC/SVC domain, some basic assumptions are no longer valid. The often used AC and DC coefficients (e.g., [9, 13–15]) of intra-coded blocks in H.264/AVC are transformed from spatially intra-predicted values instead of the original pixel values, so full decoding is necessary. Concerning our goal of unsupervised, compressed domain scene analysis, other shortcomings of former approaches include manual tracking initialization (e.g., [12, 15]), no support for camera motion (e.g., [11, 15]) and no support for multiple, occluding objects (e.g., [10]).

A few approaches specific to MPEG-4 and H.264-AVC/SVC have been proposed in the literature. Sutter et al. [16] presented a lightweight tracking algorithm for MPEG-4/FGS. No indication for the performance in the case of multiple occluding objects are given and the system has to be initialized by the user. You et al. [17] perform tracking of feature points selected by the user. The matching of these points uses the dissimilarity energies related to texture, form, and motion. Therefore, they partially decode the stream around the Region-of-Interest (ROI) back to pixel level and fully decode I-frames.

3 Compressed Domain Trajectory Estimation

The presented approach consists of the stages depicted in Fig. 1. We extract the MVs from the compressed stream, perform global motion estimation (GME), filter the outliers and perform object detection on the resulting masks. A simple matching algorithm is then applied to solve object correspondence. We introduce Object History Images (OHIs) as a tool to stabilize the trajectories. Finally, the center of gravity-based trajectories are represented by smooth splines. In the following, we further explain each of these steps. Our method does not imply

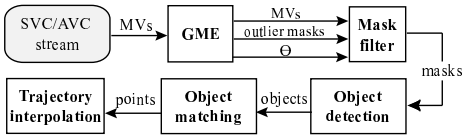


Fig. 1. Overview of trajectory estimation steps

constraints on the nature of the objects and can cope with moving cameras. However, as also stated in [7], object detection and tracking that relies solely on block-based MVs implies some requirements on the video scene. Our method is subject to the following limitations:

- The scene background should be largely static in itself. Problematic areas are water or trees in the wind. In the case of present camera motion, the background should be well textured to limit the impact of noisy MV fields.
- Moving objects should neither be too numerous nor should they occupy the whole viewable image area.

If these constraints are met, the global motion estimation will deliver valid and reliable results, which builds the basis for further processing.

3.1 H.264 AVC/SVC Test Sequences

H.264/AVC (MPEG-4/Part 10) is the successor of MPEG-2 and gains more and more popularity due to its superior performance and efficiency. H.264/SVC [18] is the scalable extension to AVC. Figure 3a shows an example of the macroblock partitions and MVs of a B-slice of the AVC-compatible base layer with a resolution of 480x272 pixels, extracted from a SVC stream with Full-HD (1080p) resolution at top level. Except for *Hall Monitor* and *Surveillance*, all of our test sequences are encoded in this format. *Hall Monitor* and *Surveillance* are single-layer streams with 352x288 pixels and 480x360 pixels, respectively.

We used the SVC reference software JSVM [19] in version 9.8 for our experiments. In case of High-Definition (HD) streams with spatial scalability, we only process the AVC-compatible base layer to save computing time. We encoded all streams with temporal scalability, enabled by the *hierarchical B-picture prediction* of SVC, with a Group-of-Picture (GOP) size of 8.

3.2 Global Motion Estimation (GME)

We adopted a similar robust motion estimation algorithm as proposed in [20] and [21], which proved to deliver good results. It basically consists of an iterative re-weighted least squares estimation of the well known 2-D 6-parameter affine model and is followed by a camera motion characterization. We estimate the global motion for each video frame.

In order to obtain the MV values in quarter-pel precision, the entropy coding of H.264 has to be reversed as the only decoding step. For each B-frame MB, depending on the prediction mode (LIST_0, LIST_1, direct or bi-prediction), we get a MV from LIST_0 and one from LIST_1. The choice between LIST_0 or LIST_1 MVs as active estimation support has shown to be arbitrary, since the distance to the reference frames in both temporal directions is the same (*hierarchical prediction structure* of SVC). We further process only forward-predicted LIST_1 MVs. To obtain uniform results, we scale all MVs by the distance to its respective reference picture. To obtain an estimate for I-frames, we take the mirrored LIST_0 vectors from the subsequent B-frame in display order as an estimation basis. MBs in *skip*-mode are excluded from the estimation support.

The 2-D 6-parameter affine motion model is given by

$$\begin{aligned} d_x &= a_1 + a_2(x - x_0) + a_3(y - y_0) \\ d_y &= a_4 + a_5(x - x_0) + a_6(y - y_0), \end{aligned} \tag{1}$$

where $(x_0, y_0)^T$ denotes the reference point in the image (e.g., the image center) and $(x, y)^T$ the MB center. We estimate the model in the weighted least squares sense with a Gaussian weighting function. The process is repeated iteratively and outliers are discarded after each iteration. It showed that convergence is reached after approximately 4 iterations. The result of the GME process is the vector $\theta = (pan, tilt, zoom, rot)$, containing the frame-wise camera operation parameters. A mapping from the parameters $a_1..a_6$ to $pan..rot$ is performed according to [21].

3.3 Outlier Masks

The outlier masks which are output of the GME process contain noise (see Fig. 2) due to the block-based estimation process. Spatio-temporal filtering of the raw outlier masks is performed to alleviate the influence of miss-detected MVs. The temporal filtering window is set to the intra-period of the coded video, which is 8 frames in our experiments. Within this window, outlier MBs are median-filtered along their motion trajectories, followed by morphological filters to fill small holes in object masks and to remove background-noise.

3.4 Object Detection

The filtered outlier masks represent silhouette images and give a rough separation of the scene in background and foreground objects. We split the masks into

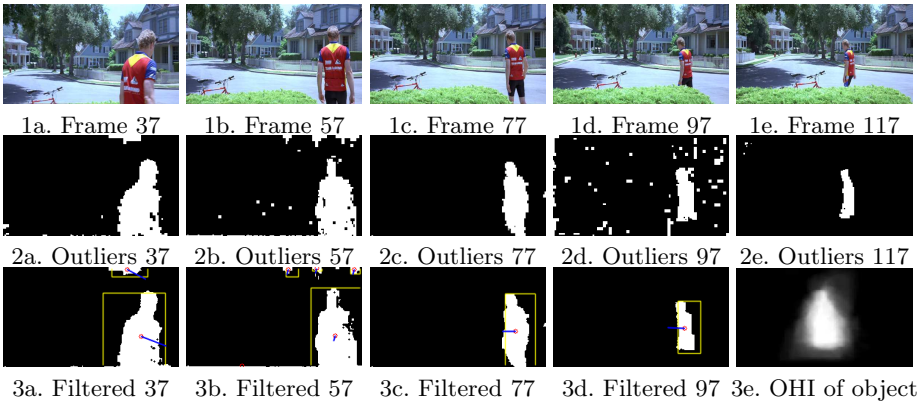
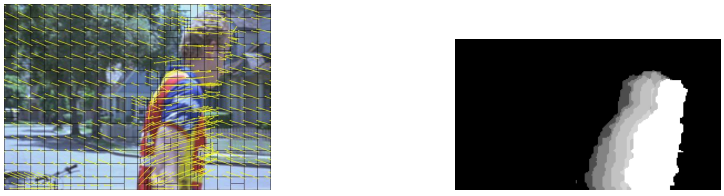


Fig. 2. Example of raw and filtered outlier masks. 3a-d) detected objects. Local object motion is represented by a vector leaving the centroid, which is represented by a circle. 3e) OHI of main object. Sequence *street with trees and bicycle* ©*Warner Bros. Advanced Media Services Inc.*

single moving objects by using a simple motion segmentation algorithm similar to the approach of timed *Motion History Images* (MHI) from Bradski and Davis [22].

Motion history images store the motion history for multiple frames in one single channel image. The MHI is updated by setting the corresponding mask pixels in the MHI to the current time stamp. Figure 3b shows an example of a MHI for the *street* sequence. Each connected region in the MHI sharing the most



a. MV field b. Motion history image

Fig. 3. Example of MVs (a) and a MHI (b) of the sequence *street_with_trees_and_bicycle*. Image (a) was magnified and cropped for better visibility.

recent time stamp is considered as one independent object. All regions smaller than a minimal, pre-defined threshold-size MIN_SIZE are discarded. We set MIN_SIZE to a region height or width smaller than 3% of the image height or width, respectively. Each detected object is labeled and a some object properties are calculated and stored for further processing: i) The object mask, defined by the connected region in the tMHI; ii) The centroid; iii) The size, i.e., the number of pixels in the mask and iv) the object motion parameters (*pan..rot*).

The object motion parameters are estimated similarly to the global motion (see Sec. 3.2), using all MVs covered by the mask. Global motion is compensated before the estimation and the reference point is set to the center of gravity of the mask. The quality of these parameters depends on the number of MVs covered by the object, indicated by the object size. The two translational parameters $a_1 \stackrel{\wedge}{=} pan$ and $a_4 \stackrel{\wedge}{=} tilt$ are robust to small estimation supports, whereas the significance of the parameters *zoom* and *rot* decreases.

Examples showing the temporal evolution of the local object motion are given in Fig. 4 for the man in the *street* sequence and for the pedestrian in *parkrun* (see Fig. 6 for screenshots). The small estimation support in the latter case leads to very noisy results for zoom and rotation. For the *street* sequence, the estimation reflects well the real object motion. The indicated zoom-in and zoom-out around frames 15-50 and 120-170 represents the objects' motion towards and away from the camera. In both figures, the curves for *zoom* and *rot* have been scaled for the sake of better comparability to *pan* and *tilt*.

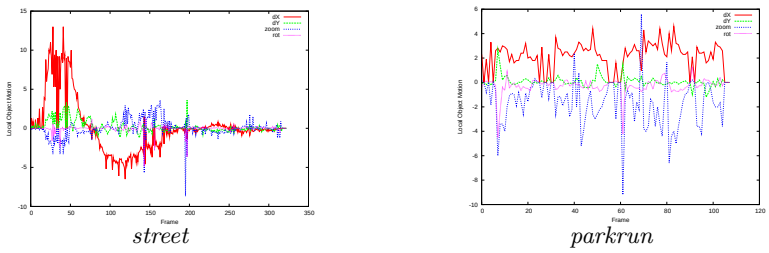


Fig. 4. Local object motion for sequences *parkrun* and *street_with_trees_and_bicycle*. The parameter *zoom* is very noisy for the pedestrian in *parkrun*, because only a very small number of MBs is covered by the mask.

3.5 Object Matching

At this stage, we have the frame-wise, independent object detection results as described above. The most important step in the trajectory estimation process is to track the detected objects over time, i.e., to identify similar objects in adjacent frames and to define a reference point within the object that represents its current position. We treat these problems separately in the following.

A temporal analysis of the calculated object properties (see Sec. 3.4) allows to draw certain conclusions about what is happening in the scene:

▷ **mask:** Represents regions in motion. Its position gives an indication if the object enters or leaves the scene.
▷ **size:** Continuous changes in size are usually caused either by objects leaving or entering the scene, by changes of the visible object surface (occlusions), by changes of the distance to the camera or by a non-rigid object that partially

stops or resumes moving. Rapid, significant changes in the object size indicate split-and-merge situations.
▷ **centroid:** Center of gravity of moving region. Rapid changes of position also indicate split-and-merge situations.
▷ **motion:** The translational motion parameters *pan* and *tilt* indicate the moving direction and predict the position in the next frame (relative to the camera position). If the estimation support is sufficiently large, *zoom* may give an indication if the object approaches or moves away from the camera.

Object Correspondence The initialization takes place when the first objects are detected at time t_i. Each object is assigned with a unique label and is kept in memory along with its properties. The expected position in the following frame is estimated using the translational motion parameters. At time $t_i + 1$, the algorithm searches a limited area of 20 pixels around the predicted position for new input objects. If an object with similar size and moving direction is found in the search area, we assign the same label to it. Otherwise we mark the object as *inactive*.

If an object of significantly larger size is found in the search area, we check if that new object coincides with the predicted position of another object. If this is true, the objects "merged" and we assign both labels to that joint object. Otherwise we check for inactive objects that have been lastly detected at this position (with compensated global motion). If there seems to be no such explanation for the abrupt change in size, we however copy the same object label to it and set a flag of uncertainty. Possible other explanations include fast objects re-appearing behind occluding obstacles, or a merging with another previously static object (e.g., a pedestrian takes a bike and rides away).

"Split" situations, where multiple smaller objects replace a big one, are treated similarly. If a crossing of multiple objects occurs (merge-split), we assume the objects' moving directions are hardly affected, so after the split we re-assign the labels according to the closest match to the motion parameters before the merging. This may lead to a false label switching in certain scenarios.

If at a given moment, a new object appears "out of nowhere", i.e., one that is neither entering nor leaving the scene, we search for *inactive* objects in that region to reactivate them. If none is found, we assign a new label to the appearing object.

3.6 Reference Point

At this stage, we have identified similar objects over time. Moving objects are often occluded by obstacles like cars or tables and we look for a reference point within the object that remains as stable as possible. We therefore chose the center of gravity. Problematic are non-rigid objects and occlusions. To give an example, the waiter in the sequence shown in Fig. 5 moves from one table to the next, stops to clean them and is often partially occluded. While wiping the table, the centroid of the mask moves away from the original one, which

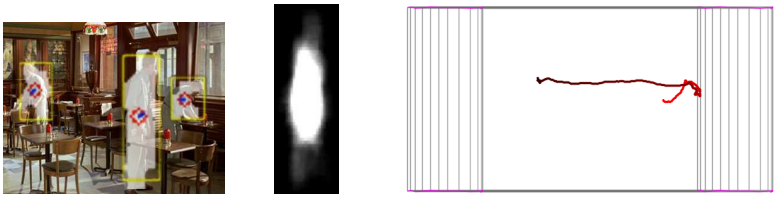


Fig. 5. Left: Exemplary deformations of the same moving object at different moments in time. Red circle is centroid of mask. Middle: OHI of object. Right: Trajectory. Sequence *man_in_restaurant* © *Warner Bros. Advanced Media Services Inc.*

was located around the waistline. In order to stabilize the reference point over time, we propose the construction of so-called *Object History Images* (OHI). We extended the idea from global MHIs [22] to object silhouette construction. The goal is to create a more stable representation of an object than the quickly fluctuating object mask.

At the first occurrence of an objet, we initialize the OHI with the first object mask. Each time a previously present object is detected in the current frame, we project the OHI to the position predicted by $\hat{a}_1$ and $\hat{a}_4$. We superimpose it with the new mask image and increment the value of the OHI at positions where mask pixels are set. If the new mask does not entirely fit into the projected OHI, we enlarge it.

We keep one "long-term" OHI for each detected object and continuously update as long as the object is visible and moving. The OHI represents a silhouette image of the object, where the most rigid regions appear brighter than parts like legs or arms. As the reference point, we compute the center of gravity, which assigns more importance to higher values. Darker zones in the OHI, like moving hands or shadows, only cause slight fluctuations. Examples of OHIs are given in Fig. 5 and Fig. 2.

The most problematic cases regarding the objects' reference points are merged object masks. When we detect a merging situation, we only update the intersection between the past OHI at its predicted position and the merged mask. This way

If the system gets initialized with merged objects that split later on, we only know after the split that the area contained more than one object. We then reset the merged OHI and re-initialize a new OHI for each object.

3.7 Trajectory Construction

We draw the trajectories in the image plane seen by the camera. The trajectories, represented by the centre of gravity of the OHIs over time, are smoothed using X-splines [23] as a final step. X-splines combine the properties of Catmull-Rom splines and cubic B-splines in one curve, adding the feature of sharp bends at abrupt turns. To achieve that, each control point is parameterized by a factor $k \in [-1; 1]$, where $k = -1$ gives Catmull-Rom like behavior (interpolation),

$k = +1$ leads to B-spline like behavior (approximation) and $k = 0$ gives a sharp bend at the control point.

For each control point, we assign k as a function of the object size in relation to the size of the OHI. If the mask size is below 30% of the OHIs' size, we assign $k = +1$, otherwise $k = -1$. That means that small masks are considered to be less reliable and their centroids are approximated rather than crossed by the spline. Control points at moments of merged masks with multiple objects are also weighted with $k = +1$, because the position estimation is less reliable due to likely inter-object occlusions.

4 Results

Figure 6 shows the estimated trajectories for some test sequences. Each trajectory plot shows the position of the visible image area over time, represented by one rectangle for every 20th image, and the global camera motion over time, represented by purple curves connecting the rectangle corners. The trajectories are drawn as thick colored lines, and the brightness of the color corresponds to the moment in time. The brightest point denotes the position in the beginning of the sequence, the darkest one the position at the end. Each detected object is represented by a different color. The most-right plots in Fig. 6 show the trajectories obtained manually by users, who we demanded to click on the estimated center of gravity of all relevant objects in each frame.

The two short lines in the top left corner of the *street* sequence trajectory plot (Fig. 6a) are caused by moving branches of a tree. By comparing the main trajectory with the camera motion, it can be noticed that the camera is following the object. This can also be noticed in the *parkrun* sequence, where a pedestrian is walking along the river and is followed by the camera. He always appears in the center of the image. The trajectory of the waiter in the *restaurant* sequence is shown in Fig. 6f. The fluctuations of the most right part of the trajectory are caused by a long period where he stands still while cleaning a table.

Figure 6e shows the results for the well-known *Hall_Monitor* sequence. Both objects are detected and tracked over time, where the jitter in the middle of the left trajectory results from the man stopping at the small table for about 50 frames. To provide an example of two crossing objects, we show the results for a surveillance video showing two pedestrians with opposed trajectories in Fig. 6c.

The largest differences between the estimated and the manually determined trajectories are observed in the *Kung Fu* sequence (Fig. 6d). The system did not recognize that one fighter over-jumped the other, which ducked down, and could not exactly follow during the vigorous fight. However, the object did not get lost and the trajectories reflect well both positions.

Table 1 summarizes the results of the object *detection* stage for the given test sequences. We counted the total number of object occurrences in all frames and provide results for the number of correctly detected objects, the missed detections and false positives. The high numbers of missed objects in the sequences *Hall Monitor* and *Man in Restaurant* appear because objects stop moving for

several frames and we detect only objects in motion. The trajectories are hardly affected of this detection loss, because the objects are correctly re-identified after they continue moving. The false positives in the *Street* sequence represent moving branches of a tree, which we consider as background.

The processing times given in Tab. 1 were measured on a 2.16 GHz Intel Core2Duo with 2 GB of RAM. The simplicity of the algorithm allows real-time processing.

5 Conclusions

We presented an approach to estimating the trajectories of moving objects in the H.264 compressed domain. The method is completely unsupervised and is entirely based on the motion vectors present in the compressed stream. It is able to detect and track multiple objects of any kind, given that they also appear clearly geometrically separated at some moments in time. Our method is computationally efficient and can cope with complex camera motion. An inconvenience is the dependency on reliable global motion estimation results. We will further evaluate and improve our algorithm for different types of videos and applications.

6 Acknowledgments

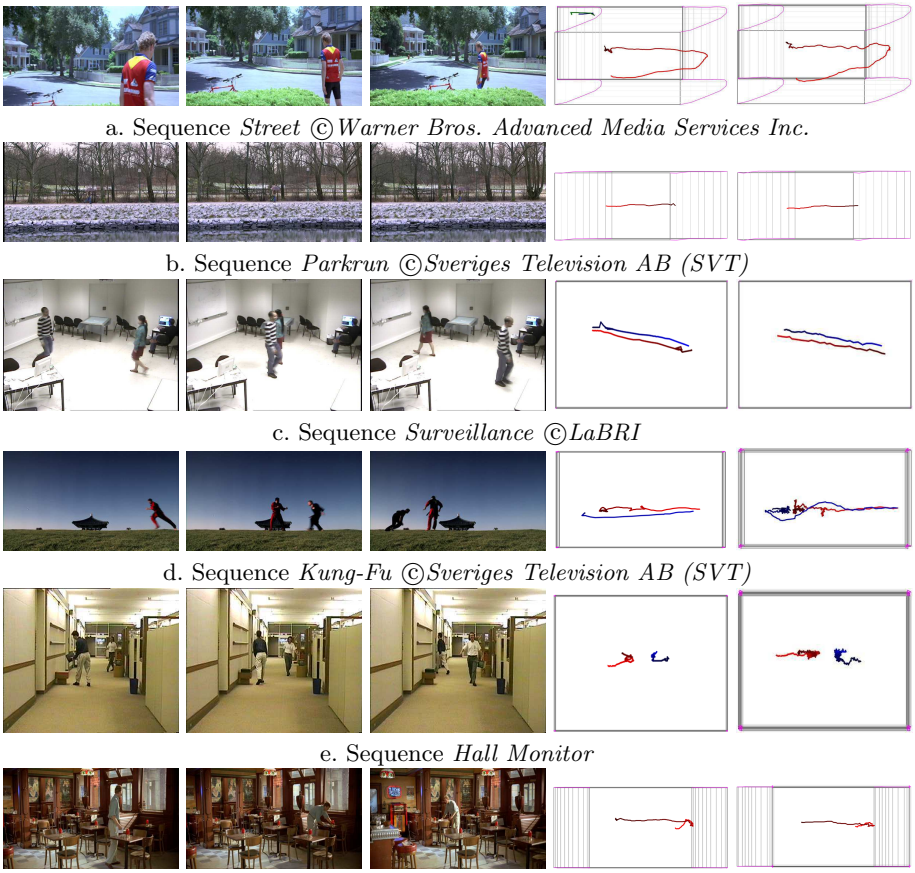
This work has been carried out in the context of the french national project ICOS-HD (ANR-06-MDCA-010-03) funded by the ANR (Agence Nationale de la Recherche).

Table 1. Object Detection Results

Sequence	Duration in frames (sec)	Corr. detected objects	Missed objects	False positives	Processing time in sec (fps)
street	270 (10.8s)	268/270 (99%)	2/270	22	10.1s (26.7 fps)
parkrun	100 (4.0s)	95/100 (95%)	5/100	3	3.5s (28.5 fps)
surveillance	118 (4.7s)	224/236 (95%)	12/236	3	4.32s (27.3 fps)
kung fu	180 (7.2s)	291/303 (96%)	14/303	0	6.7s (26.8 fps)
hall monitor	300 (12.0s)	404/455 (89%)	51/455	0	11.3s (26.5 fps)
man in restaurant	310 (12.4s)	288/310 (93%)	22/310	7	10.9s (28.4 fps)

References

1. Bruyne, S.D., Neve, W.D., Schrijver, D.D., Lambert, P., Verhoeve, P., de Walle, R.V.: Shot boundary detection for h.264/avc bitstreams with frames containing multiple types of slices. In: PCM. (2007) 177–186



a. Sequence *Street* ©*Warner Bros. Advanced Media Services Inc.*

b. Sequence *Parkrun* ©*Sveriges Television AB (SVT)*

c. Sequence *Surveillance* ©*LaBRI*

d. Sequence *Kung-Fu* ©*Sveriges Television AB (SVT)*

e. Sequence *Hall Monitor*

f. Sequence *Man in Restaurant* ©*Warner Bros. Advanced Media Services Inc.*

Fig. 6. Results for some test sequences. From left to right: 1-3) Screenshots 4) Estimated trajectory 5) Manually obtained trajectories

2. Babu, R.V., Ramakrishnan, K.: Content-based video retrieval using motion descriptors extracted from compressed domain. Volume 4., Phoenix, USA (2002) 141–144
3. Babu, R.V., Ramakrishnan, K., Srinivasan, S.: Video object segmentation: a compressed domain approach. IEEE Transactions on Circuits Systems for Video Technology **14**(4) (2004) 462–474
4. Zeng, W., Du, J., Gao, W., Huang, Q.: Robust moving object segmentation on h.264/avc compressed video using the block-based mrf model. Real-Time Imaging **11**(4) (2005) 290–299
5. Sukmarg, O., Rao, K.: Fast object detection and segmentation in mpeg compressed domain. In: 10th IEEE Region Annual International Conference. Volume 3., Kuala Lumpur, Malaysia (September 2000) 364–368
6. Mezaris, V., Kompatsiaris, I., Boulgouris, N.V., Strintzis, M.G.: Real-time compressed-domain spatiotemporal segmentation and ontologies for video indexing

and retrieval. In: IEEE Transactions on Circuits and Systems for Video Technology. Volume 14. (2004) 606–621
7. Hesseler, W., Eickeler, S.: Mpeg-2 compressed-domain algorithms for video analysis. EURASIP Journal on Applied Signal Processing **2** (2006) 1–11
8. Lie, W.N., Hsiao, W.C.: Content-based video retrieval based on object motion trajectory. In: IEEE Workshop on Multimedia Signal Processing. (December 2002) 237–240
9. Radhakrishna, A., Kankanhalli, M., Mulhem, P.: Compressed domain object tracking for automatic indexing of objects in mpeg home video. In: IEEE International Conference in Multimedia and Expo (ICME 2002), Lausanne, Switzerland (August 2002)
10. Park, S.M., Lee, J.: Compressed domain object tracking for automatic indexing of objects in mpeg home video. In: 4th Pacific Rim Conference on Multimedia. Volume 2., Singapore (December 2003) 748–752
11. Lie, W.N., Chen, R.L.: Tracking moving objects in mpeg-compressed videos. In: IEEE International Conference on Multimedia and Expo (ICME'01). Volume 2001. (2001) 245
12. Favalli, L., Mecocci, A., Moschetti, F.: Object tracking for retrieval applications in mpeg-2. In: IEEE Transactions on Circuits and Systems for Video Technology. Volume 10. (April 2000) 427–432
13. Chen, H., Zhan, Y., Qi, F.: Rapid object tracking on compressed video. In: PCM '01: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia, London, UK, Springer-Verlag (2001) 1066–1071
14. Manerba, F., Benois-Pineau, J., Leonardi, R., Mansencal, B.: Multiple moving object detection for fast video content description in compressed domain. EURASIP J. Adv. Signal Process **2008**(1) (2008) 1–13
15. Aggarwal, A., Biswas, S., Singh, S., Sural, S., Majumdar, A.: Object tracking using background subtraction and motion estimation in mpeg videos. In: ACCV06. (2006) II:121–130
16. Sutter, R.D., DeWolf, K., Lerouge, S., de Walle, R.V.: Lightweight object tracking in compressed video streams demonstrated in region-of-interest coding. EURASIP J. Appl. Signal Process. **2007**(1) (2007) 59–59
17. You, W., Sabirin, M., Kim, M.: Moving object tracking in h.264/avc bitstream. In: MCAM07. (2007) 483–492
18. Schwarz, H., Marpe, D., Wiegand, T.: Overview of the scalable h.264/mpeg4-avc extension. In: IEEE International Conference on Image Processing (ICIP'06), Atlanta, USA. (October 2006) 161–164
19. Software, J.R.: Reference software for h.264/svc. http://ftp3.itu.ch/av-arch/jvt-site/
20. Bouthemy, P., Gelgon, M., F.Ganansia: A unified approach to shot change detection and camera motion characterization. Volume 9. (October 1999) 1030
21. Durik, M., Benois-Pineau, J.: Robust motion characterisation for video indexing based on mpeg2 optical flow. In: Proceedings of International Workshop on Content-Based Multimedia Indexing (CBMI'01), Brescia, Italy. (September 2001) 57–64
22. Bradski, G.R., Davis, J.W.: Motion segmentation and pose recognition with motion history gradients. Mach. Vision Appl. **13**(3) (2002) 174–184
23. Blanc, C., Schlick, C.: X-splines: a spline model designed for the end-user. In: SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM (1995) 377–386