1st Int. Workshop on Media Fragment Creation and reMIXing (MMIX'13) at the IEEE Int. Conf. on Multimedia and Expo (ICME 2013), San Jose, CA, USA, July 2013.

FAST OBJECT RE-DETECTION AND LOCALIZATION IN VIDEO FOR SPATIO-TEMPORAL FRAGMENT CREATION

Evlampios Apostolidis, Vasileios Mezaris, Ioannis Kompatsiaris[*]

Information Technologies Institute / Centre for Research and Technology Hellas, Thermi 57001, Greece
{apostolid, bmezaris, ikom}@iti.gr

ABSTRACT

This paper presents a method for the detection and localization of instances of user-specified objects within a video or a collection of videos. The proposed method is based on the extraction and matching of SURF descriptors in video frames and further incorporates a number of improvements so as to enhance both the detection accuracy and the time efficiency of the process. Specifically, (a) GPU-based processing is introduced for specific parts of the object re-detection pipeline, (b) a new video-structure-based sampling technique is employed for limiting the number of frames that need to be processed and (c) improved robustness to scale variations is achieved by generating and employing additional instances of the object of interest based on the one originally provided by the user. The experimental results show that the algorithm achieves high levels of detection accuracy while the overall needed processing time makes the algorithm suitable for quick instance-based labeling of video and the creation of object-based spatio-temporal fragments.

Index Terms— Object Re-detection, SURF descriptors, RANSAC algorithm, Shot Segmentation, GPU processing

1. INTRODUCTION

Object re-detection can be seen as a particular case of image matching, aiming at finding occurrences of a specific object within a single video or a collection of videos. For doing so, an object re-detection algorithm takes as input an image that depicts the object of interest and evaluates its similarity with different pieces of the video content. Moreover, if the object of interest has been annotated with an appropriate label, this object re-detection procedure could assign this label to the detected re-appearances, thus providing a tool for instance-level annotation of the video. Based on this, object re-detection could contribute to the realization of interactive and linked television, since the identification of object-specific spatio-temporal fragments in the video and their association with appropriate labels are important steps for finding related content fragments and for establishing links between them.

[*]This work was supported by the European Commission under contracts FP7-287911 LinkedTV and FP7-318101 MediaMixer.

2. RELATED WORK

One of the most popular state-of-the-art approaches for estimating the similarity between pairs of images is based on extracting and matching scale- and rotation-invariant local descriptors. Local salient features of the image can be extracted using edge (e.g. Canny [1]) or corner detectors (e.g. Harris-Laplace [2]) or even by applying a dense sampling strategy (i.e. sampling image patches on a regular dense grid). Subsequently local descriptors, like the well-known SIFT [3] and SURF [4] ones, can be used for the representation of the detected interest-points, while indicatively other approaches include the BRISK [5] descriptor, an affine-invariant extension of SIFT, called A-SIFT [6], and binary approaches like the BRIEF [7] descriptor. Moreover, the development of GPU-based implementations of these descriptors' extraction software offers a significant boost in time performance. Such implementations for the SIFT and SURF descriptors have been described for example in [8], [9].

A simple method for matching the descriptors in the following step, is to perform k-nearest neighbor (k-NN) search for each extracted descriptor in a brute-force manner, while other techniques accelerate the matching procedure by applying hashing algorithms such as the Dichotomy Based Hash (DBH) [10]. After computing the matched pairs of descriptors a refinement step is usually performed aiming to filter-out the outliers. A common strategy for this purpose is to apply some ratio test regarding the distances of the calculated nearest neighbors [3], while a geometric verification between the matched pair of images can be performed using for example the RANSAC algorithm [11].

Some researchers have also tried to extend the matching procedure for detecting and tracking still objects in videos. In [12] the SURFTrac algorithm was proposed, combining SURF descriptors and motion information in order to predict the position of the interest points at the subsequent frame of the video and thus reducing the search area and the needed processing time. Another method for automatic object labeling was presented in [13], which is based on interest point tracking and a multiple region matching strategy for propagating region labels from a single frame to the next ones, while a similar approach for fast video object-based annotation was described in [14]. This method propagates the bounding box of the object by performing interest-point matching, enhanced by a color adaptation scheme and a bounding box repositioning algorithm. Alternative approaches for real-time object detection that are based on a prior fast learning step have also been proposed. In [15] a database with the object's patches, corresponding to a range of possible camera viewpoints, is initially created and during the detection the algorithm matches the incoming interest points of the video frames against the stored patches, while a similar approach which combined incremental interest-point matching with an optical flow tracker and a RANSAC pose estimator is presented in [16]. The lack of need for computing descriptors makes these algorithms fast at run-time. Finally, some approaches for the detection of specific instances of objects have been described in methods for instance-based video retrieval that participated to the Instance Search Task of TRECVID. An example of these techniques is the algorithm in [17] which creates a Bag-of-Words (BoW) representation [18] of the object of interest, combining a hessian-affine region detector with a SIFT-variant descriptor, called RootSIFT [19]. Then, matching between the BoW descriptors is based on a ratio test and the geometric consistency is tested using a RANSAC-based algorithm, proposed in [20].

Although these methods exhibit good performance in terms of detection accuracy, their time-efficiency, as reported in the corresponding publications, makes them inadequate for real-time processing of high resolution professional videos with 25 or 30 fps frame-rate. For this purpose we designed a fast and accurate technique for object re-detection in videos, whose needed processing time is several times shorter than the video's actual duration. As indicated by the experimental evaluation, the implemented technique constitutes a fast and reliable tool for the creation and labeling of object-based spatio-temporal fragments in videos.

3. BASELINE APPROACH FOR OBJECT RE-DETECTION

After reviewing the state-of-the-art we developed a baseline OpenCV implementation that builds on the extraction and matching of local descriptors. The algorithm takes as input an image that depicts the object of interest *O* and a video *V* of *N* frames and detects instances of this object by evaluating its similarity against all the frames of this video successively. For each frame of the video an appropriate bounding box is stored by calculating the coordinates of the upper-left corner (x, y) and its width and height. When no instance is detected the stored bounding box is [0 0 0 0]. After processing all video frames the algorithm performs a temporal filtering on the overall detection results, aiming to minimize false positives (i.e. erroneous detections) and false negatives (i.e. erroneous misses). This procedure is summarized in Algorithm 1.

Algorithm 1 Baseline object re-detection algorithm.

```
Notation: $O$ is the manually selected object of interest, $V$ is
    the video to be processed containing $N$ frames, $F_i$ is the
    i-th frame of the video and $\approx$ symbolizes the matching
    operation.
Input: An image that depicts the selected object of interest
    $O$ and a video $V$ of $N$ frames.
Output: A file of $N$ entries where each one of them corre-
    sponds to a video frame and contains the coordinates of
    the calculated bounding box.
1: for $i = 1 \to N$ do
2:     Perform $O \approx F_i$
3:     Store the bounding box
4: end for
5: Perform temporal filtering on the detection results.
```

Interest-point detection and description are performed using the SURF algorithm [6], while the following matching step is implemented in a brute-force manner (i.e. each extracted descriptor from the object *O* is matched against all the extracted descriptors from the i-th frame F_i) looking each time for the 2 best matches via a k-NN search for k = 2. So, for each detected interest point of *O* we search for the 2 best matches in F_i that correspond to the 2 nearest neighbors N_1 and N_2. Erroneous matches are then discarded by applying the following rule: keep an interest point in *O* and its corresponding best match in F_i if $\|DistN_1\|_1/|DistN_2\|_1 \leq 0.8$, where $\|\|_1$ is the Manhattan distance between the interest point in *O* and each of the calculated nearest neighbor. After this matching is performed for all points in *O*, additional outliers are filtered-out by estimating the homography between *O* and F_i using the RANSAC algorithm [11]. If a sufficient number of pairs of descriptors remains after this geometric validation step then the object is said to be detected in F_i and an appropriate bounding box is calculated, otherwise a bounding box of the form [0 0 0 0] is stored for this frame. The work-flow of the baseline implementation for object re-detection is illustrated in Fig. 1. The final filtering step of the overall detection results is based on a sliding window of 21 frames and a set of temporal rules that decide on the existence or absence of the object *O* in the middle frame of this window.

4. PROPOSED APPROACH FOR OBJECT RE-DETECTION

Building on this baseline implementation we tried to improve performance, both in terms of detection accuracy and time efficiency, by working on the following three directions: (a) GPU-based processing, (b) video-structure-based sampling of video frames and (c) improved robustness to scale variations. Our work on each of them is described in the following sections.

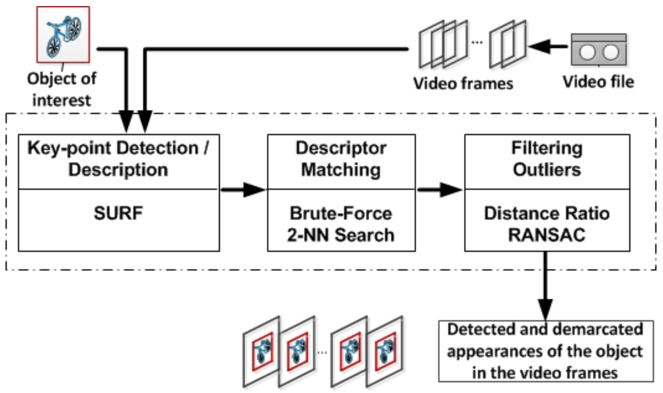

Fig. 1. The overall work-flow of the baseline implementation for object re-detection.

4.1. GPU-accelerated object re-detection

Some parts of the baseline implementation are accelerated by exploiting the processing power of the modern Graphic Processing Units (GPUs). More specifically in this time-improved version, the GPU is used for: (a) the decomposition of the video into frames, (b) the feature extraction and description, using a GPU-based implementation of the SURF algorithm, (c) the matching of the extracted descriptors, using a GPU-based version of the Brute-Force matcher with 2-NN search and (d) for drawing the estimated bounding boxes on the video frames. These GPU-based processes are realized using code included in version 2.4.3. of the OpenCV library. The only parts of the algorithm that are still handled by the CPU are: (a) the filtering of erroneous matches after the descriptor matching step and (b) the post-processing of the detection results with temporal filtering. After these modifications the algorithm exhibited significant reduction of the processing time, preserving the same levels of detection accuracy.

4.2. Video-structure-based sampling of video frames

Aiming at further reduction of the overall processing time we designed a new algorithm that realizes an efficient sampling of the video frames, replacing the sequential processing with a structure-based one. The improved object re-detection algorithm utilizes the analysis results of automatic shot segmentation (using an algorithm proposed in [21]). So, besides the object of interest *O* and the video *V*, the improved algorithm reads a matrix *S* that is the output of the shot segmentation algorithm, where $S_{i,j}, j = 1, ...5$ are the information on the i-th shot. Specifically, $S_{i,1}$ and $S_{i,2}$ are the shot boundaries, i.e. the index of the starting and ending frames of the shot and $S_{i,3}$, $S_{i,4}$, $S_{i,5}$ are the indices of three representative key-frames of this shot.

The re-detection procedure is accelerated since the new algorithm initially tries to match the object *O* with the 5 frames of the i-th shot that are identified in matrix S (i.e. $S_{i,j}, j = 1, ...5$), and only if matching is successful, it proceeds with comparing *O* against all frames of that shot. It then continues with the key-frames of the next shot, until all shots have been checked. By applying this sampling strategy the algorithm analyses in full only the parts (i.e. the shots) of the video where the object appears (being visible in at least one of the key-frames of these shots) and quickly rejects all remaining parts by performing a small number of comparisons. The updated algorithm is described in Algorithm 2.

Algorithm 2 Object re-detection with efficient sampling of video frames.

Notation: Further to the notation used in Alg 1, $S_{i,j}$ are the elements of the matrix *S* with the boundaries and the key-frames of each shot and *M* is the number of shots.
Input: Same as for Alg. 1, plus the matrix *S*.
Output: Same as for Alg. 1.

```
1: for $i = 1 \rightarrow M$ do
2:   for $j = 1 \rightarrow 5$ do
3:     Perform $O \approx F_{S_{i,j}}$
4:     if success then
5:       for $k = S_{i,1} \rightarrow S_{i,2}$ do
6:         Perform $O \approx F_k$
7:         Store the bounding box
8:       end for
9:       Break
10:     end if
11:   end for
12: end for
13: Perform temporal filtering on the detection results.
```

4.3. Robustness to scale variations

The last extension of our algorithm aims to improve the robustness to scale variations. Major changes in scale of the object, which are common in realistic scenarios, may lead to detection failure due to the significant limitation of the area that is used for matching. E.g. a highly zoomed-in instance of the object may correspond to a very small portion of the searched object *O*, while an instance that is seen from a very distant viewing position may occupy only a small part of the overall image. Trying to handle these levels of scale transformation, we automatically generate a zoomed-out O_2 and a zoomed-in O_3 instance of the object *O* (denoted as O_1 from now on) and we utilize them in the matching procedure. Specifically, when the object O_1 is not matched against a frame or key-frame of the video, then the algorithm repeats the matching attempt using firstly the zoomed-out instance O_2 and then the zoomed-in one O_3. So, the pipeline of the final version of the object re-detection method is presented in Algorithm 3.

The construction of the zoomed-in instance O_3 was based on the selection of a center-aligned sub-area of the original object O_1 and the enlargement to the actual size of *O*

Algorithm 3 Object re-detection with efficient sampling of video frames and robustness to scale variations.

```
Notation: Further to the notation used in Alg. 2, $O_l, l = 1, ...3$ are the originally selected object of interest and
    the automatically generated zoomed-out and zoomed-in instances of it respectively.
Input: Same as for Alg. 2.
Output: Same as for Alg. 2.
1: Read $O$ and generate additional instances
2: for $i = 1 \to M$ do
3:   Flag = MISDETECTION
4:   for $j = 1 \to 5$ do
5:     for $l = 1 \to 3$ do
6:       Perform $O_l \approx F_{S_{i,j}}$
7:       if success then
8:         for $k = S_{i,1} \to S_{i,2}$ do
9:           for $l = 1 \to 3$ do
10:            Perform $O_l \approx F_k$
11:            if success then
12:              Store a bounding box
13:              Break
14:            else
15:              if $l = 3$ then
16:                Store [0 0 0 0] (no detection)
17:              end if
18:            end if
19:          end for
20:        end for
21:        Flag = DETECTION
22:        Break
23:      end if
24:    end for
25:    if Flag = DETECTION then
26:      Break
27:    end if
28:  end for
29: end for
30: Perform temporal filtering on the detection results.
```

by applying bilinear interpolation. So, the choice is about the size of the centralized area of the original object O that we want to focus on. In our experiments the size of the selected area was occupying 70% of the actual size of O_1, which corresponds to a 140% zoom-in factor. The construction of the zoomed-out instance O_2 was performed by shrinking the original image O_1 into a smaller one using nearest neighbor interpolation. The zoom-out factor for each selected object was determined by the restrictions of the GPU-based implementation of SURF. These restrictions are related to the number of octaves used for the analysis. Based on the parameterization of the used SURF descriptor (4 octaves) the zoom-out factor is calculated according to the formula $F^* = max\{184/O_h, 184/O_w\}$, where O_h and O_w is the height and width of the object O_1 respectively.

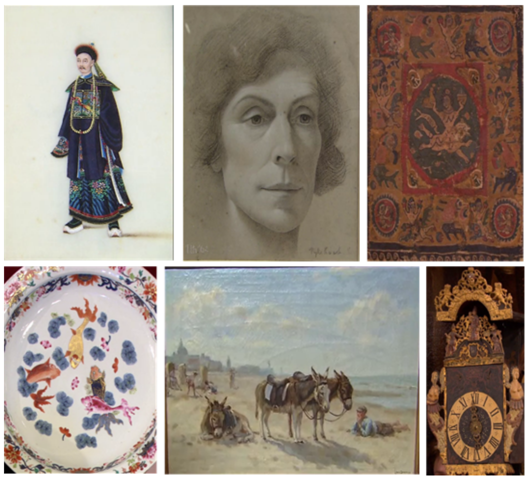
Fig. 2. Examples of tested objects, selected from a cultural heritage show.

5. EXPERIMENTS AND RESULTS

For the evaluation of the algorithm's performance, both in terms of detection accuracy and time efficiency, we conducted a set of experiments on a system with an Intel Core i7 processor at 3.4GHz, 8GB RAM memory and a CUDA-enabled NVIDIA GeForce GTX560 graphics card. The used dataset consists of (a) 6 videos of 273 minutes total duration and 409.087 frames and (b) 30 manually selected objects. The videos are episodes from a cultural heritage show of the Dutch public broadcaster AVRO[1], called "Antiques Roadshow", and the selected objects are objects that appear in these shows like paintings, cards, plates and teapots, jewelleries and clocks, small carpets, etc. (see Fig. 2). According to the ground-truth data (generated via manual annotation of the overall set of frames) 75.632 frames contain at least one of these objects, whereas none of the selected objects appears in the remaining 333.455 frames. Regarding the size of these objects, after initial experiments we concluded that an instance O with an average size (among the various appearances of this object in the video) provides the best detection performance, and this practice was used for the selection of appropriate instances of the used objects in the experiments, so as to maximize the detection accuracy of the baseline that we compare with.

Moreover, aiming to quantify the improvement that each extension of the baseline approach is responsible for, we tested the following experimental configurations: C_1: baseline implementation, C_2: GPU-accelerated implementation, C_3: GPU-accelerated and video-structure-based sampling implementation and C_4: complete proposed approach (GPU-acceleration, video-structure-based sampling, and robustness

[1]http://avro.nl/

to scale variations implemented).

The detection accuracy of the algorithm is described in terms of Precision, Recall and F-Score. The evaluation was performed in a per-frame basis considering the 30 selected objects and counting the number of frames where these were correctly detected, missed, etc. Time efficiency was evaluated by expressing the processing time of each configuration as a factor of real-time processing i.e. comparing these times with the actual duration of the processed videos (a factor below 1 indicates faster-than-real-time processing). The experimental results are summarized in Table 1. For the last column (Time), a range is reported for each configuration, since processing times can vary significantly depending on the video structure and the percentage of frames in which the sought object appears.

As can be seen all versions of the algorithm exhibited very good results in terms of detection accuracy, while the last one (C_4) that includes the mechanism for the enhancement of the algorithm's robustness to scale transformations achieved the best results. The algorithm performed considerably well for a range of different scales and orientations and when the object was partially visible or partially occluded. Indicative examples of such cases are presented in Fig. 3.

For further quantifying the impact of the extension presented in Section 4.3 to the accuracy of object re-detection, we also run experiments with two specific sets of frames where the object of interest is observed from a very close and a very distant viewing position, respectively. So, we selected from the complete dataset a set of 2.940 frames with highly zoomed-in instances and a set of 4.648 frames with highly zoomed-out instances, and for each set we performed the same experiments as before. The results are summarized in Tables 2 and 3. We can see in these tables that the Recall and F-Score values indicate that our complete proposed approach (C_4) clearly outperforms all the previous configurations.

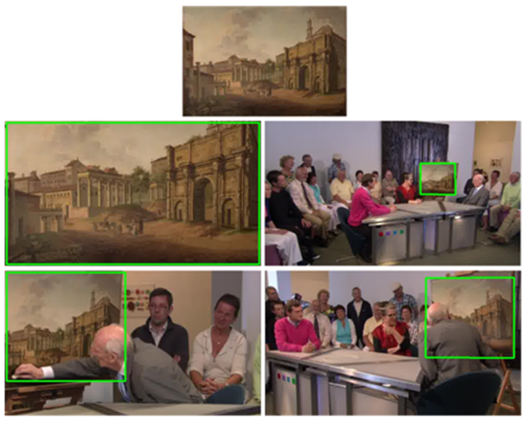

Fig. 3. Object of interest (top row) and in green bounding boxes the detected appearances of it, after zoom in/out (middle row) and occlusion-rotation (bottom row).

Table 1. Evaluation results for configurations C_1 to C_4.

	Precision	Recall	F-Score	Time (x Real-Time)
C_1	0.999	0.868	0.929	2.98-5.26
C_2	0.999	0.850	0.918	0.35-1.24
C_3	0.999	0.849	0.918	**0.03-0.13**
C_4	**0.999**	**0.872**	**0.931**	0.03-0.19

Table 2. Evaluation results for highly zoomed-out instances.

	Precision	Recall	F-Score
C_1	0.999	0.856	0.922
C_2	0.999	0.856	0.922
C_3	1	0.852	0.920
C_4	1	**0.992**	**0.996**

Table 3. Evaluation results for highly zoomed-in instances.

	Precision	Recall	F-Score
C_1	0.999	0.831	0.907
C_2	0.999	0.831	0.907
C_3	1	0.799	0.888
C_4	1	**0.914**	**0.955**

The only case where the proposed method failed to detect the object of interest was for extremely high zoom-in and zoom-out factors. After analysing the experimental results we concluded that the algorithm can effectively detect the object for a wide range of scaling factors that vary from roughly 25% (zoom-out) to 250% (zoom-in) of the actual image size, but fails for lower and higher values.

Regarding time performance, the development of the video-structure-based sampling strategy led to a great reduction of the required processing time. The proposed algorithm needs about 10% (averaging the values of the last column in Table 1) of the video's duration to perform the analysis, while it preserves the same high levels of detection accuracy with the previous slower configurations. Comparing the proposed method with other approaches from the reviewed literature, we can see that among the fastest existing methods is the one of [16], which runs at approximately 28 fps, and is therefore suitable for the real-time processing of videos. However, as mentioned in Section 2, this algorithm initially performs an off-line learning step by building a database of image patches, which significantly increases the overall processing time.

6. CONCLUSIONS

In this paper we presented a method for fast and accurate object re-detection in video that is based on the extraction and matching of local descriptors. Aiming at real-time performance we accelerated some parts of the algorithm with GPU-based processing, and we introduced a new structure-based sampling technique that limits the number of video frames that have to processed. Moreover, we improved the detection accuracy by enhancing the algorithm's robustness to scale variations using two additional automatically constructed instances of the original image in the detection procedure. The experiments show that the proposed technique can be used as a reliable tool for fast and accurate detection and labeling of object-based spatio-temporal fragments in videos.

7. REFERENCES

[1] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, June 1986.

[2] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of 4th Alvey Vision Conference*, 1988, pp. 147–151.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, June 2008.

[5] S. Leutenegger, M. Chli, and R. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proc. of the IEEE Int. Conf. on Computer Vision*, Nov. 2011.

[6] G. Yu and J. M Morel, "A fully affine invariant image comparison method," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing, ICASSP 2009*, April, pp. 1597–1600.

[7] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: binary robust independent elementary features," in *Proc. of the 11th European conference on Computer vision - Part IV*, Berlin, Heidelberg, 2010, ECCV'10, pp. 778–792.

[8] S. Heymann, K. Müller, A. Smolic, B. Fröhlich, and T. Wiegand, "Scientific Commons: SIFT implementation and optimization for general-purpose GPU," 2007, The 15th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision.

[9] N. Cornelis and L. Van Gool, "Fast scale invariant feature detection and matching on programmable graphics hardware," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPRW '08*, June, pp. 1–8.

[10] Z. He and Q. Wang, "A fast and effective dichotomy based hash algorithm for image matching," in *Proc. of the 4th International Symposium on Advances in Visual Computing*, Berlin, Heidelberg, 2008, ISVC'08, pp. 328–337.

[11] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *ACM Communications*, vol. 24, no. 6, pp. 381–395, June 1981.

[12] D.-N. Ta, W.-C. Chen, N. Gelfand, and K. Pulli, "Surftrac: Efficient tracking and continuous object recognition using local feature descriptors," in *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR'09*, June.

[13] J. Fauqueur, G. Brostow, and R. Cipolla, "Assisted video object labeling by joint tracking of regions and keypoints," in *11th IEEE Int. Conf, on Computer Vision, ICCV 2007*, Oct., pp. 1–7.

[14] R. Trichet and B. Merialdo, "Generic object tracking for fast video annotation," in *2nd Int. Conf. on Computer Vision Theory and Applications, VISAPP 2007*, March.

[15] S. Hinterstoisser, O. Kutter, N. Navab, P. Fua, and V. Lepetit, "Real-time learning of accurate patch rectification," in *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR'09*, June, pp. 2945–2952.

[16] J. Mooser, Q. Wang, S. You, and U. Neumann, "Fast simultaneous tracking and recognition using incremental keypoint matching," in *Proceedings of the 4th Int.l Symposium on 3D Data Processing, Visualization and Transmission, 3DPVT'08*, Atlanta, USA, June 2008.

[17] A. Bursuc, T. Zaharia, O. Martinot, and F. Preteux, "Artemis-ubimedia at trecvid 2012: Instance search task," in *Proceedings of the 10th TRECVID Workshop*, Gaithersburg, USA, November 2012.

[18] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proc. of the 9th IEEE Int. Conf. on Computer Vision*, Washington, USA, 2003, ICCV '03, pp. 1470–1477.

[19] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2012*, pp. 2911–2918.

[20] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2007*.

[21] E. Tsamoura, V. Mezaris, and I. Kompatsiaris, "Gradual transition detection using color coherence and other criteria in a video shot meta-segmentation framework," in *15th IEEE Int. Conf. on Image Processing, ICIP 2008*, Oct.