Multimedia Tools and Applications, Springer, 2015, accepted for publication.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/s11042-015-3125-0

A web-based tool for fast instance-level labeling of videos and the creation of spatiotemporal media fragments

Anastasia Ioannidou, Evlampios Apostolidis, Chrysa Collyda, Vasileios Mezaris

***Abstract*—This paper presents a web-based interactive tool for time-efficient instance-level spatiotemporal labeling of videos, based on the re-detection of manually selected objects of interest that appear in them. The developed tool allows the user to select a number of instances of the object that will be used for annotating the video via detecting and spatially demarcating it in the video frames, and provide a short description about the selected object. These instances are given as input to the object re-detection module of the tool, which detects and spatially demarcates re-occurrences of the object in the video frames. The video segments that contain detected instances of the given object can be then considered as object-related media fragments, being annotated with the user-provided information about the object. A key component for building such a tool is the development of an algorithm that performs the re-detection of the object throughout the video frames. For this, the first part of this work presents our study on different approaches for object re-detection and the finally developed one, which combines the recently proposed BRISK descriptors with a descriptor matching strategy that relies on the LSH algorithm. Following, the second part of this work is dedicated to the description of the implemented tool, introducing the supported functionalities and demonstrating its use for object-specific labeling of videos. A set of experiments and a user study regarding the efficiency of the introduced object re-detection method and the performance of the developed tool indicate that the proposed framework can be used for accurate and time-efficient instance-based annotation of videos, and the creation of object-related spatiotemporal media fragments.**

***Index Terms*—Instance-level video labeling, Object re-detection, BRISK descriptor, Locality Sensitive Hashing**

I. Introduction

Nowadays there is a rapidly growing number of videos available on the web, either on video-sharing web-sites (e.g., YouTube and Vimeo) or within on-line archives of content providers, such as broadcasters and news organizations. A main prerequisite for the consumption of this content by wide audiences is to make it searchable and easy to access (e.g., via some form of links between related content, similar in principle to the hyperlinks between text). To this direction, the semantically coherent fragments of a video must be somehow defined and annotated with appropriate (text-based) labels, that would make these pieces of media searchable and linkable with related content.

Motivated by the lack of video analysis/editing tools that can support media fragmentation and annotation tasks, we built an interactive tool that can be used for semi-automatic instance-specific labeling of video content and the creation of self-contained spatiotemporal media fragments, based on the re-detection of manually pre-specified objects that appear in these pieces of media. To this end, we developed a fast and accurate object re-detection algorithm and we integrated it in a web-based framework that supports real-time interaction between the user and the content. With this tool, a user can select a number of different instances of an object by spatially demarcating appearances of it in the video frames, while after this a brief description about the object can optionally be given and used as tag for video labeling. Based on this input, re-occurrences of the selected object are automatically detected and highlighted in the video frames (by drawing a colored bounding box around them), while the required processing time is only a very small fraction of the video's overall duration. Considering a video segment that is defined by the first and the last detected instance of the given object as an object-specific media fragment labeled by the user-defined tag, we end up with a piece of annotated multimedia content that is available for searching and linking purposes.

Starting from a baseline approach for image matching that relies on the use of local descriptors, the main contributions of this paper are:

- an overview of the relevant literature and the reporting on: (a) methods that can be used at each specific step of this basic analysis pipeline, namely the detection of interest points, the extraction of descriptor vectors, the matching of the computed descriptors and the geometric validation of the defined matches, and (b) techniques that follow a different approach for performing object re-detection or tracking in videos
- a step-by-step evaluation process where: (a) the efficiency of a number of different approaches that can be utilized at each step of the processing pipeline is assessed based on the needs for accurate and time-efficient analysis, and (b) robust conclusions regarding the most suitable methods are made and used for progressively defining the methodology that we eventually propose for object re-detection in videos
- the introduction of a technique for quick and accurate object re-detection in videos which consists of an off-line analysis part where the detection and extraction of descriptor vectors from the video frames is applied, and an on-line analysis part where the re-detection of the

A. Ioannidou, E. Apostolidis, Ch. Collyda and V. Mezaris are with the Information Technologies Institute/Centre for Research and Technology Hellas (CERTH), Thermi 57001, Greece (email: {ioananas, apostolid, ckol, bmezaris}@iti.gr).

object in the frames of the video is performed, based on descriptor matching and a structure-based frame sampling strategy that accelerates the analysis
- the description of a web-based interactive tool that integrates the proposed method for object re-detection and can be used for object-specific spatiotemporal labeling of videos
- a comprehensive evaluation of the proposed framework (via a user study and considering both the algorithm's performance and the tool's efficacy) as a means for time-efficient and highly accurate instance-based video annotation, and a performance comparison with other object tracking methodologies, that highlights the superiority of the introduced method for object re-detection in videos

The rest of the paper is organized as follows: Section II presents an overview of related work and Section III introduces the proposed object re-detection algorithm, which is experimentally evaluated in Section IV. Section V gives a detailed presentation of the developed web-based tool for video labeling, while Section VI presents the design and the findings of the user study that was performed for evaluating the proposed method and the developed tool. Finally, the conclusions of this work are presented in Section VII.

II. Related Work

The detection of a given object within a collection of images or a set of video frames is most commonly performed based on the application of an image matching strategy, where the decision about the existence/absence of the object within an image (or a video frame) is taken after evaluating the visual similarity among them. Baseline approaches for image matching usually start by representing the visual content of the images with the help of visual descriptors. Methods that focus on the representation of local features of the images (such as edges and corners) are more preferable than techniques that describe global characteristics of the images (such as histograms) for this type of analysis, since they allow a finer matching of features between a pair of images. Following, a matching strategy is applied for finding correspondences between the extracted sets of descriptors, while afterwards a filtering step is usually performed for removing outliers. Finally, the decision about the existence/inexistence of the object in the image is taken via thresholding the number of matched pairs of descriptors.

Concerning the part related to the representation of the visual content of an image, a variety of different methods were introduced for the detection of local features, such as the Harris [22] corner detector, the Difference of Gaussians (DoG) [34] or the Maximally Stable Extremal Regions (MSER) [37]. These algorithms are then used in combination with scale- and rotation-invariant local descriptors, such as SIFT [34] and SURF [7]. Several works that aim to reduce the computing efforts needed for local feature detection and description, and for matching the extracted descriptor vectors, have been described in the literature (such as the PCA-SIFT [27] and the Gradient Location and Orientation Histogram (GLOH) [38] descriptors), however the time performance of most of these methods, with some exceptions (e.g., [2], [17]), makes them inefficient for real-time user-interactive applications.

To this direction, recent efforts led to the development of new, lightweight descriptors which can be used in visual analysis applications for mobile devices (i.e., smartphones and tablets), that have restricted computing capabilities compared to modern desktops. Most of these techniques perform simple pixel-based comparisons and build binary vectors, such as the BRIEF [10], the BRISK [30], the FREAK [3] and the ORB [44] descriptors. These descriptors offer reduced computational complexity compared to SIFT and SURF, and are significantly faster when used in matching tasks, since the latter is performed via bitwise operations of the modern CPUs. Recent surveys (see [11], [39]) that aimed to evaluate the performance of different detectors and descriptors, indicated the matching efficiency and the significant speed-up that binary descriptors can provide.

As mentioned before, the accurate matching of sets of descriptors is one of the main processing steps that are performed by object re-detection techniques, and for this reason descriptor matching strategies have attracted a lot of attention in the last few years. Some of these strategies address the finding of correspondences between sets of descriptors as a k Nearest Neighbor (k-NN) search problem. Exact nearest neighbors searching techniques, such as the simple exhaustive search approach, offer reliable results but in most cases are extremely time-consuming, thus, being inappropriate for real-time applications. Aiming to shorter searching time, a variety of methods for approximate nearest neighbor search has been proposed, and can be divided into approaches that rely on the use of trees and techniques that employ hashing algorithms. The first group of methods includes, but is not limited to, the KD-trees [8], [20], the randomized KD-trees [46], the priority search k-means tree [40], the hierarchical k-means tree [21] and the hierarchical clustering tree [40]. The second group contains several hashing algorithms such as coherency sensitive hashing [29], supervised hashing with kernels [32], spherical hashing [45], spectral hashing [51] and locality sensitive hashing (LSH) [16]. The latter is widely used as a quick and effective technique for image matching and retrieval applications, with a number of variations of it been proposed over the last years [4], [24], [36], [53].

No matter which nearest neighbor search technique is applied, most commonly the outcome of the matching procedure is not completely precise. For this reason, different methods have been used for filtering outliers (i.e., pairs of erroneously matched descriptors), such as distance ratio tests that consider the distance between the computed nearest neighbors after a 2-NN search [34], or more elaborate geometric verification criteria such as the RANSAC algorithm [18] and variations of it [13], [14], [43].

Nowadays, motivated by the processing capability of modern CPUs (Central Processing Units) and GPUs (Graphics Processing Units) that enable multi-core-based or multi-threading-based parallel processing, a number of works that aim to improve the time performance of many visual analysis algorithms were proposed. Parallel implementations of several feature extraction and description algorithms, such as SIFT and SURF,

were introduced over the last years, relying either on CPU- or GPU-based processing [31], [33], [50], [52], [54]. Moreover, a parallelized version of the KLT (Kanade-Lucas-Tomasi) method that is mentioned later in this section, was described in [19]. Regarding the descriptor matching process, [26] and [47] present parallelized versions of the Brute-Force method, while [56] and [42] described GPU-based implementations of the KD-tree and the LSH algorithm respectively. Last but not least, a parallel implementation of the RANSAC algorithm that is widely used for the geometric validation of the matched pairs of descriptors was presented in [43].

Besides the analysis workflow for image matching described extensively above, another set of methods for object re-detection in videos are the ones related to the so-called long-term model-free object tracking. These methods get as input an instance of the object in a video frame and then perform detection and localization of the object in all consecutive frames of the video. The accurate detection of the object's occurrences that appear under different illumination, viewing position and can be partially occluded, is a key challenge in this task as well. Many techniques of this category perform object re-detection based on motion information. One of them is the KLT method of [35], [49], which belongs to the so-called Sparse Optical Flow algorithms. The latter, contrary to the Dense Optical Flow algorithms, estimate the displacement of only a selected number of pixels in the image, relying on local information extracted from small windows surrounding them. Based on the original algorithm, a pyramidal-based variation of it was proposed in [9]. Another technique is the SURFTrac algorithm [48] which combines SURF descriptors and motion information in order to predict the position of the interest points at the subsequent frame of the video, leading to a smaller search area and, thus, reducing the needed processing time. Moreover, the efficiency of hybrid image feature trackers was examined in [1]. In this work, local detectors and descriptors, such as FAST, BRIEF and SURF, were combined with feature motion trackers such as the KLT, and the results were compared with the ones obtained using each tracker separately. A different approach is the Mean-Shift algorithm [15] which re-detects a given object in a video frame by performing block-level histogram comparisons, with the best target location being the one that maximizes the similarity measured by the Bhattacharyya coefficient. The Mean-Shift method was also combined with SIFT [55] in a mutual support mechanism that results in robust performance. Another approach that relies on the use of the BRISK detector and descriptor [41], combines matching and tracking techniques in order to re-detect interest points extracted from the given object on subsequent frames of the video. The reported evaluations indicate that this technique outperforms other state-of-the-art methods.

Alternatively, a number of introduced techniques address the object re-detection task as an on-line learning problem. The method of [23] performs a training phase in the Fourier space, exploiting a circulant matrix representation which leads to detection at high frame rates. Another framework for long-term object tracking [25] decomposes the overall task into 3 parts: tracking, learning and detection. The algorithm initially estimates the object's motion from frame to frame, the detector scans every incoming image in order to localize all object's appearances detected so far, while the learning component of the algorithm monitors the overall process and creates positive and negative examples in order to avoid future errors, thus improving the detection accuracy.

A baseline approach for object re-detection that follows the main steps of the workflow described at the beginning of this section was presented in [6]. According to this, the SURF algorithm is used for extracting and describing the local features of images. Then, the matching of the extracted descriptors is performed in a Brute-Force manner, applying each time a 2-NN search in combination with a filtering criterion about the distance of the computed neighbors [34]. Subsequently, a geometric verification of the remaining matches based on the RANSAC algorithm is applied to enhance the algorithm's robustness. Building on this, the authors of [6] introduced a number of modifications and extensions, resulting in a technique for fast object re-detection in videos that exhibits improved performance both in terms of detection accuracy and time efficiency.

III. Proposed Approach

Aiming to develop an object re-detection method that would be integrated into the proposed video labeling tool, and motivated by the fact that the object-specific video annotation is a semi-automatic procedure that involves real-time interaction between the user (i.e., the video editor) and the video content, we concluded that the analysis has to be performed in a sufficiently quick way (without compromising the re-detection accuracy). Based on this assumption, we initiated our study from the method described in [6], which exhibited high re-detection performance, trying, at the first place, to assess the needed processing time for each analysis step that is depicted in Fig. 1. As illustrated in this block diagram, the pipeline includes: (a) the detection of interest points from each image, (b) the extraction of descriptor vectors for the detected interest points, (c) the pair-wise matching of the computed descriptor vectors, and (d) the filtering of the outliers based on geometric validation. After this, a final decision about the matching is taken based on a simple thresholding of the number of matched pairs of descriptors. For the needs of object re-detection in videos, the parts related to the detection and description of interest points from the given object that will be re-detected throughout the video are applied only once, while the entire chain of analysis is performed for every frame of the video during its matching against the object.

Our first experiment aimed at measuring and assessing the required processing time by each analysis step (i.e., the steps (a) to (d) described above). The results of this evaluation are reported in Section IV-B. From this experiment it was clearly shown that the most time-consuming parts of the analysis are the ones related to the detection and description of interest points from the images (steps (a) and (b) from Fig. 1), as well as the matching of the calculated descriptor vectors (step (c) from Fig. 1). Based on this outcome we studied alternative options for each of these steps. This was performed in a progressive way, investigating each time a number of

Fig. 1 Block diagram of the core analysis steps that are applied by the object re-detection algorithm of [6]

different algorithms for one of these processing step and using a standard pair of techniques for the other two. Every time, the efficiency of each defined combination of methods was evaluated based on the required processing time, while the object re-detection accuracy was also taken into account.

For interest point detection we compared the SURF algorithm, that is employed in [6], against three other techniques. Specifically, we considered the MSER [37], the BRISK [30] and the ORB [44] algorithms for interest point detection from images, since, according to the findings reported in these works, these methods can efficiently detect the descriptive parts of the images that will be consequently used for descriptor extraction and matching. The results of this evaluation are presented in detail in Section IV-C. The conducted experiments indicated that SURF detector is not the most efficient in terms of running time (in fact, it is the third faster among the evaluated methods), however its re-detection accuracy makes this algorithm the best choice for this kind of analysis.

Having identified the most effective method for interest point detection, we then tried to investigate alternatives for the extraction of descriptor vectors. The SURF descriptors used in [6] have been indicated as one of the most efficient ones for visual content representation in the relevant literature, outperforming in many case the effectiveness of other gradient-based approaches (e.g., SIFT or GLOH). Motivated by the current research trend in using more lightweight binary descriptors (compared to the floating point representations of the gradient-based descriptors) for computer vision applications, we decided to evaluate the performance of SURF descriptors against some of these new approaches, choosing in particular, the BRISK, the ORB, the FREAK and the BRIEF descriptors. As before, we compared both the time performance and the re-detection accuracy of these binary descriptors with the performance of the SURF algorithm. Regarding time performance, we considered both the time needed for computing the descriptor vectors and the time required for their matching, since the latter also affects the overall time efficiency of these methods. The experimental results described in Section IV-D outline the effectiveness of the BRISK descriptor, which ensures similar object re-detection accuracy with SURF, while being several times faster.

The output of interest point detection and description after processing the i–th frame of the video are the matrix $\mathbf{C_i}$ of size $B \times 2$ and the matrix $\mathbf{D_i}$ of size $B \times 64$, where B is the number of detected interest points from this frame and 64 is the length of the computed descriptor vectors. Specifically, $\mathbf{C_i}$ stores the (x,y) coordinates of each detected interest point in frame F_i and $\mathbf{D_i}$ stores the computed 64-element binary vectors for these points.

Based on the findings of our evaluations so far, we determined the most preferred methods for the detection (i.e., SURF) and description (i.e., BRISK) of interest points. So, in the next step we focused our study on techniques for matching the extracted descriptor vectors. Contrary to the Brute-Force 2-NN search applied in [6] we examined an approach for approximate, yet fast, nearest neighbor search that relies on the LSH algorithm. This choice was based on the fact that the LSH method has been shown (as reported in relevant literature, e.g., [44]) to be effective for fast matching of binary features. After a set of experiments for tuning the LSH algorithm's parameters, an approach that uses 3 hash tables and 5 bits for the hash key was finally selected, since it exhibited the best trade-off between matching accuracy and required processing time. For each tested combination of methods (i.e., SURF descriptors and Brute-Force matching, BRISK descriptors and LSH matching), we initially evaluated the descriptor matching efficiency based on the number of correctly matched pairs of descriptors that occur after also applying the RANSAC-based filtering of erroneous matches. Then, we assessed the effectiveness of each approach in terms of required processing time and provided object re-detection accuracy. The results of these evaluations are reported in detail in Section IV-E, denoting that the combination of BRISK descriptors with the LSH-based matcher is the most efficient approach, both in terms of time efficiency and re-detection accuracy.

Despite the fact that the use of the BRISK and LSH methods for descriptor extraction and matching respectively led to a significant reduction of the required processing time, as indicated by the findings reported in Sections IV-C, IV-D and IV-E, the parts of the analysis related to the detection and description of interest points (i.e., steps (a) and (b) of the block diagram of Fig. 1) are still quite time-demanding and consume over 75 % of the overall processing time. Based on this observation and thinking of ways to further accelerate the video annotation process, we tried to indicate parts of the analysis that need to be performed during the re-detection of the object throughout the video (on-line), and other parts that could be applied during a prior (off-line) analysis process. So, we ended up to the applicable and functional scenario where interest point detection and description are applied on the entire set of video frames during a, prior to video annotation, off-line processing step, and the computed data for each frame are stored in a file. Then, during the on-line processing where the re-detection of instances of the given object in the video frames is performed, interest points and descriptor vectors are computed only once for the user-defined object, and the pre-computed files from the off-line analysis are loaded and used for matching. For this, we explored several options regarding the storage of the computed BRISK descriptors. Among a variety of file formats, such as text files, binary files and xml files, and driven by the size of the created files which strictly affects the storage cost of the algorithm, we ended up to the use of binary

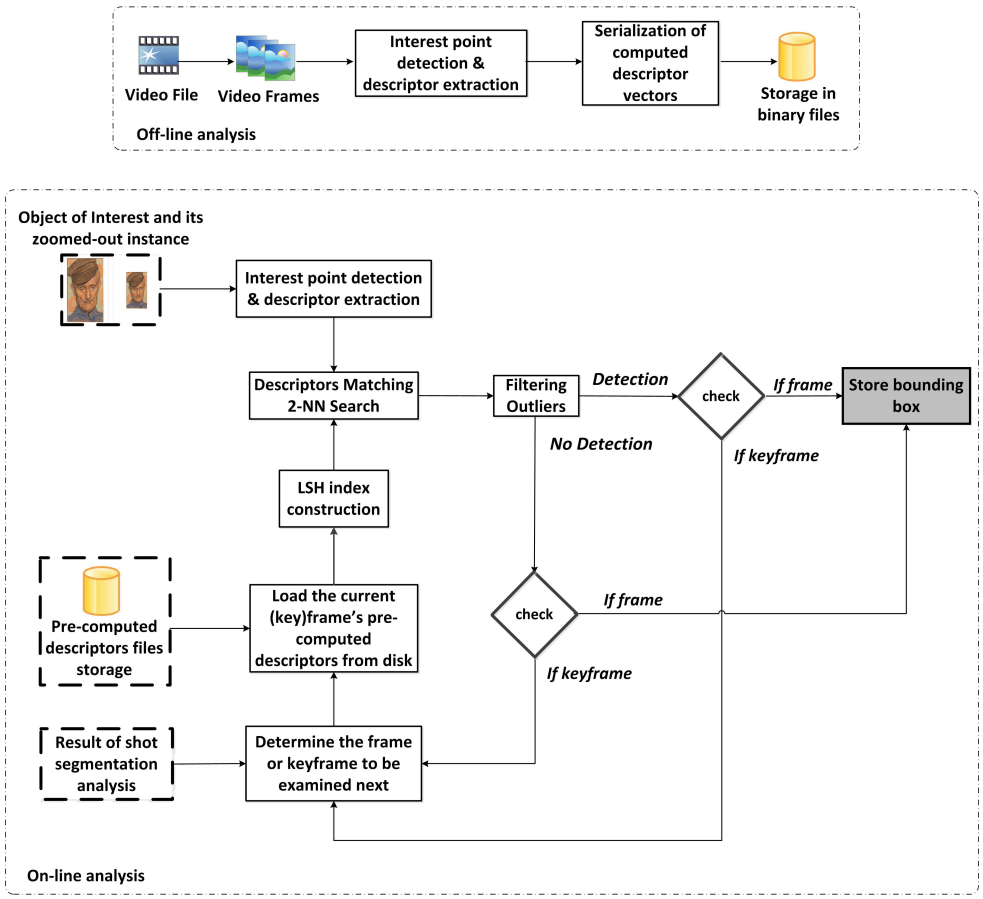


Fig. 2 The proposed object re-detection framework. The dashed line boxes indicate the algorithm's input, while the gray shaded one represents the output

files due to their small file size. Moreover, aiming to further improve the computation efficiency of the off-line step of this methodology, we serialized the computed matrices of interest points and descriptor vectors before storing them. Detailed experiments about the speed-up achieved using pre-computed data during the object re-detection procedure are presented in Section IV-F. This section also reports the findings of the conducted evaluations regarding the time performance of the entire analysis pipeline (i.e., interest point detection based on SURF, interest point description based on BRISK and descriptor matching based on LSH) when this pre-processing step is applied.

From the outcomes of this set of evaluations we concluded to a new framework for object re-detection that is depicted in Fig. 2. The developed approach was realized using the OpenCV[1] (ver. 2.4.9) library for visual analysis and the Boost [2] (ver. 1.55.0) library for serialization and storage of the computed descriptor vectors. The proposed technique consists of two parts, namely the off-line and the on-line analysis part. During the off-line analysis (see the upper box with the dot-dashed bounding line in Fig. 2), interest point detection and descriptor extraction are applied for every video frame F_k (with $k = 1,...N$ and N being the number of video frames) using the SURF and the BRISK algorithm respectively, and the computed data are serialized and stored as binary files. The on-line analysis (see the bottom box with the dot-dashed bounding line in Fig. 2) gets 3 different types of data as input. The first one is the set of manually selected instances of the object O_c (with $c = 1,...W$ and W being the number of selections) that will be re-detected throughout the video. The developed algorithm creates downscaled versions of them (via pixel sub-sampling) that will be used for detecting extremely distant appearances of the object in the video frames (see the upper box with the dashed bounding line in the left area of Fig. 2). The second one is the serialized and stored data about the detected interest points and the extracted descriptors of each frame that were created by the off-line analysis (see the middle box with the dashed bounding line in the left area of Fig. 2). The third one is a matrix S that contains information about the shot-level structure of the video, extracted after processing the video with a shot segmentation algorithm, e.g., from [5].

After having all the needed input available, the algorithm extracts descriptor vectors for each instance of the object (both the provided ones and the automatically created downsized versions of them), using the SURF and BRISK methods for interest point detection and description respectively. Then, the analysis of the video is performed as depicted in Fig. 2 and

[1] http://opencv.org/
[2] http://www.boost.org

presented in detail in Algorithm 1, by applying the frame-sampling strategy of [6]. For doing so, the algorithm reads the given matrix S, where $S_{i,j}, j = 1,...5$ are the information on the i-th shot of the video. 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 keyframes of this shot. Then, starting from the first shot of the video it compares each given instance $O_{c,1}$ and its downsized version $O_{c,2}$ with the 5 characteristic frames of this shot, where the frames of the first shot are identified in matrix S (i.e., $S_{1,j}, j = 1,...5$). For this, an LSH index is constructed for each frame after loading its pre-computed descriptors from the binary file. The created index is then used for matching the indexed descriptors with the descriptors extracted from the instance $O_{c,l}$ (with $c = 1,...W$ and $l = 1,2$), by applying a 2-NN search and the distance ratio criterion of [6]. Erroneously matched descriptors are subsequently filtered out based on geometric verification using the RANSAC method. The final decision about the existence/absence of the instance $O_{c,l}$ in the matched frame is performed after thresholding the number of defined pairs of descriptors. In case of detection the algorithm proceeds with comparing the object's instances $O_{c,l}$ against all frames of that shot, starting with the instance that was matched during the initial comparison with the characteristic frames of the shot. Otherwise it is considered that the sequence of frames that corresponds to this shot of the video does not contain any appearances of the given object, and the analysis continues with the characteristic frames of the next shot until all shots of the video have been checked. For each detected occurrence of the object in the video frames the algorithm computes a bounding box and stores the coordinates of its upper-left corner (x,y), its width and its height. For the remaining frames (the ones without detected instances of the object) a bounding box of the form [0 0 0 0] is stored. A temporal filtering of the overall detection results 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 is applied, forming the final output of the analysis. By applying this frame 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 keyframes of these shots) and quickly rejects all remaining parts by performing a small number of comparisons.

Having decided about the different core analysis components of the new object re-detection framework we evaluated more extensively its performance, both in terms of time efficiency and detection accuracy, using a large set of objects and videos (see Section IV-A). In this evaluation the final combination of the proposed method (the one that involves the off-line analysis of the video) was compared against the algorithms from [6] and [41]. The results of this experiment are presented in Section IV-G and highlight the detection effectiveness of the proposed algorithm and the considerable gain in the time required for analysis, despite the fact that the entire re-detection process runs in CPU. Additional comparative evaluations of the proposed algorithm were carried out with the help of the interactive tool described in Section V; these are reported in Sections VI-B and VI-C.

IV. Experiments

This section describes the conducted experiments that helped us to evaluate the performance of the studied algorithms which can be utilized at each step of the analysis pipeline (i.e., the detection of interest points, the extraction of descriptor vectors and the matching of the computed descriptors), to select the most appropriate among them (considering both time performance and detection accuracy) and to justify our choices that resulted in the finally proposed method for object re-detection in videos. After this, an extensive performance evaluation and comparison using a large dataset of videos and objects and involving other algorithms from the state-of-art, allowed us to extract concrete conclusions regarding the effectiveness of the developed framework.

A. Dataset

The used dataset in our evaluations is twice the size of the dataset described in [6] and consists of (a) 12 episodes of a cultural heritage show of the Dutch public broadcaster AVRO [3] called "Antiques Roadshow" with 545 minutes total duration (video frame resolution 720×480 and video frame-rate 25 fps), and (b) 60 manually selected objects that appear in these videos. The selected objects include paintings, cards, plates and teapots, small carpets, pieces of jewellery and clocks. Some of them are 2-dimensional, such as paintings, carpets, cards and posters, while others are clearly 3-dimensional, shown from different viewpoints, such as clocks, jars and small statues. Indicative samples from the used collection of objects are illustrated in Fig. 3. Based on the ground truth for this dataset (created via human observation), 127.764 video frames contain at least one instance of the considered objects, whereas none of the selected objects appears in the remaining 689.469 frames of our video collection.

For the first experiments that focused on specific analysis steps of the image matching procedure (i.e., the detection and description of interest points and the matching of descriptor vectors), we employed a subset of this dataset, composed of 5 different objects. These objects were selected as some of the most challenging ones, since they were displayed in a significant number of frames and under all considered viewing conditions. For each object three video frames, including a zoomed-in (denoted as "zi" in the following tables), a zoomed-out (denoted as "zo" in the following tables) and a rotated/occluded instance (denoted as "r/o" in the following tables) of it respectively, were also used in these evaluations, being part of this smaller dataset. The employed group of images is illustrated in Fig. 4.

The used metrics in our evaluations were (a) the Precision (P), (b) the Recall (R) and (c) the F-Score as defined by the formulas below:

$$P = \frac{tp}{tp+fp}, \quad R = \frac{tp}{tp+fn}, \quad F-Score = \frac{2 \times (P \times R)}{P+R} \quad (1)$$

where tp is the number of true positives (i.e., correctly detected instances of the object), fp is the number of false positives

[3] http://avro.nl

Algorithm 1 Object re-detection with efficient sampling of video frames and robustness to scale variations

Notation: $O_{c,l}$ ($c = 1,...W$ and $l = 1,2$) are the W selected instances of the object $(l = 1)$ and the created downsized versions of them $(l = 2)$, V is the video to be processed consisting of N frames, F_p is the p-th frame of the video and $\approx$ symbolizes the matching operation. $S_{i,j}$ are the elements of the matrix S with the boundaries and the keyframes of each shot, and M is the number of shots.

Input: A set of W selected instances that depict the object of interest $O_{c,1}$ $(c = 1,...W)$, a video V of N frames, the serialized binary files, and the matrix with the shot segmentation analysis results S.

Ensure: A file of N entries where each one of them corresponds to a video frame and contains the coordinates of the calculated bounding box.

```
for c = 1 → W do
    Read O_{c,1} and create the downsized instance O_{c,2} via pixel sub-sampling
end for
for i = 1 → M do
    Flag = MISDETECTION
    for j = 1 → 5 do
        Load pre-computed data for F_{S_{i,j}}
        Create LSH index for F_{S_{i,j}}
        for c = 1 → W do
            for l = 1 → 2 do
                Perform O_{c,l} ≈ F_{S_{i,j}}
                if success then
                    for k = S_{i,1} → S_{i,2} do
                        Load pre-computed data for F_k
                        Create LSH index for F_k
                        for c = 1 → W do
                            for l = 1 → 2 do
                                Perform O_{c,l} ≈ F_k
                                if success then
                                    Store [x, y, w, h] of the bounding box
                                    Flag = DETECTION
                                    Break
                                else
                                    if c = W && l = 2 then
                                        Store [0 0 0 0] (no detection)
                                    end if
                                end if
                            end for
                            if Flag = DETECTION then
                                Break
                            end if
                        end for
                    end for
                    Flag = DETECTION
                    Break
                end if
            end for
        end for
        if Flag = DETECTION then
            Break
        end if
    end for
end for
Perform temporal filtering on the detection results
Store the final outcome of the analysis
```

(i.e., erroneously detected instances of the object) and fn is the number of false negatives (i.e., erroneously missed instances

Fig. 3 Examples of selected objects for our experiments. **(a)** Static 2D or 3D objects, **(b)** Instances of 3D objects shown from different points of view, **(c)** Multiple instances of 3D objects demonstrated on a rotating disk

of the object).

The time efficiency of each tested approach was evaluated by expressing the needed processing time as a factor of real-time processing, i.e., comparing these times with the actual duration of the processed videos, thus a factor below 1 indicates faster-than-real-time processing. The experiments were conducted on a system with an Intel Core i7 CPU at 3.4 GHz, 8GB RAM memory and a CUDA-enabled NVIDIA GeForce GTX560 graphics card.

B. Time efficiency of the method proposed in [6]

As described in Section III, we initially tried to indicate the most demanding parts of the algorithm from [6], in terms of needed processing time. For this purpose, we measured the time consumed separately by each step of the image matching process using the set of images of Fig. 4. The results (average values) of this assessment are presented in Fig. 5.

From this evaluation it was shown that the most computationally expensive part of the analysis is the one related to the extraction of descriptor vectors, which accounts for 46 % of the overall needed time. The detection of interest points and the matching of the extracted descriptors correspond to 28 % and 24 % of the required time for analysis, respectively. The applied geometric validation for filtering-out erroneously matched pairs of descriptors corresponds to a very small fraction (only 2 %) of the total processing time. Based on these findings, we decided to focus on the investigation of alternative techniques for implementing the first three and most computationally demanding parts of the image matching process.

C. Comparison of different approaches for interest point detection

For interest point detection we compared the already used SURF detector (i.e., an approximation of the Hessian matrix using box-type filters and integral images to speed-up the computation) against 3 different algorithms. Specifically, we considered (a) the MSER [37] detector which detects regions that remain stable after a number of intensity thresholdings of the image, (b) the ORB [44] detector which relies on the FAST corner detector enhancing it with the use of an image scale pyramid and the Harris corner measure, and (c) the BRISK [30] detector which exploits another variation of FAST, called AGAST corner detector, and detects interest points in a scale-space pyramid. The Hessian parameter of the SURF detector was set to 400, as in [6]. For the rest of the evaluated

Fig. 4 Used objects in our matching experiments (left column) and 3 video frames depicting appearances of them under different viewing conditions

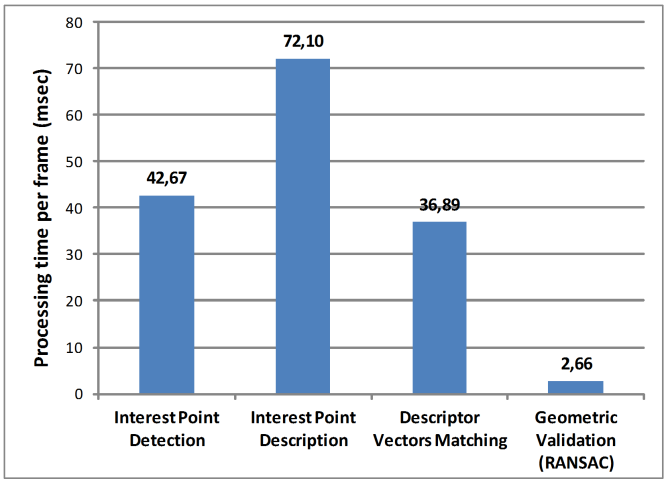


Fig. 5 Required processing time (in msec) of each step of the algorithm proposed in [6]

detectors, the default OpenCV parameter values were used. As mentioned in Section III, the efficiency of each method was evaluated based on the needed time for the detection of interest points and the accuracy of the image matching process. The description of the detected interest points was based on the SURF algorithm, while the matching of the extracted descriptor vectors was performed using the Brute-Force 2-NN searching approach combined with distance ratio test of [6].

The results of this assessment are presented in Tables I and II. Table I contains data (average values) about the time performance of each evaluated method, reporting the number of detected interest points (first row), the needed processing time (second row) and the time consumed per interest point (third row). From these values it seems that the ORB and BRISK detectors are the most time-efficient. These techniques exhibit similar performance requiring around 10 msec for analysing a frame and 0.018 msec per interest point. SURF is almost 4 times slower compared to BRISK and 5 times slower compared to ORB, however this difference is smaller if the number of detected interest points is taken under consideration. In particular the number of SURF-based detected interest points is 2 and 2.5 times larger than the number of the detected interest points using the BRISK and the ORB method respectively. The latter highlights the discriminative ability of the SURF algorithm and indicates that this method is less than two times slower compared to BRISK and ORB in terms of needed processing time per interest point. Finally, MSER exhibited the worst time performance among the tested approaches, detecting at the same time the smallest number of interest points per frame.

Table II illustrates the cases where each examined approach succeeded (✓) or failed (X) to detect the given object in the corresponding frames that included scaled (i.e., zoomed in (zi) or zoomed out (zo)) and rotated/occluded (r/o) instances of it. These data indicate that SURF detector clearly outperforms the other evaluated methods, resulting in successful detection of the object in all considered cases. The other approaches

TABLE I Time performance of the tested approaches for interest point detection

	SURF	MSER	BRISK	ORB
Detected interest points	1340	378	688	500
Processing time/frame (msec)	42.67	90.23	12.14	8.83
Processing time/interest point (msec)	0.032	0.24	0.018	0.018

TABLE II Object re-detection accuracy of the tested approaches for interest point detection

	Interest point detectors											
	SURF			MSER			BRISK			ORB		
	zi	zo	r/o	zi	zo	r/o	zi	zo	r/o	zi	zo	r/o
Object 1	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	X	✓
Object 2	✓	✓	✓	✓	✓	✓	X	X	X	✓	X	X
Object 3	✓	✓	✓	X	✓	X	X	✓	X	X	✓	✓
Object 4	✓	✓	✓	✓	✓	X	X	✓	✓	✓	✓	✓
Object 5	✓	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	✓

led to a number of false negatives which varies between 20 % and 53 % (for MSER and BRISK detector respectively). Based on the outcome of this evaluation and motivated by our goal to accelerate the analysis without compromising the detection accuracy of the algorithm (as we already stated in Section III), we concluded to use the SURF algorithm for detecting interest points from images, despite the fact that this method is not the fastest among the tested ones.

D. Comparison of different approaches for interest point description

Our evaluations with methods for extracting descriptor vectors for each detected interest point included (besides the SURF algorithm that was used in [6]) a number of binary descriptors provided by the OpenCV library, as reported in Section III. In particular, the considered approaches were the BRIEF descriptor [10], the ORB descriptor [44], the BRISK descriptor [30] and the FREAK descriptor [3]. The BRIEF descriptor sets the value of each bit of the computed vectors as the result of the comparison of two points determined by a sampling pattern of randomly pre-selected points around each detected interest point. The ORB descriptor extends BRIEF by exploiting orientation data (i.e., from the FAST detector) and by replacing the random sampling pattern with one constructed via machine learning techniques. A symmetric, circular pattern around each interest point is employed by the BRISK descriptor, which is used for the construction of a set of long-distance point pairs and a set of short-distance point pairs. The first set is used for estimating the orientation of each interest point, while the latter is used to form the descriptor vector. The FREAK algorithm also uses a circular sampling pattern which is though, inspired by the retinal pattern in the eye. A cascade for comparing the point pairs obtained from the pattern is used for defining discriminative descriptor vectors.

The results from these experiments are presented in Tables III and IV. Similarly as before, Table III focuses on the time efficiency of these methods, considering the needed time for both extracting and matching the descriptor vectors, while Table IV reports the achieved object re-detection performance of each evaluated approach. More specifically, Table III shows the size of each vector computed by each algorithm (first row), which directly affects the time needed for its extraction (second row) and matching (third row). From these measurements the BRIEF algorithm seems to be the fastest one requiring only 4 msec per frame for descriptor extraction, while the ORB and BRISK methods follow, being almost 2 times slower. The most time-consuming binary descriptor is the FREAK which is several times (5 to 10) slower compared to the other ones, while the floating point representation used by the SURF descriptors makes their extraction by far the most computationally expensive, and thus the slowest approach. However, taking into account the time required for matching the computed descriptor vectors, since this part is directly affected by the employed descriptor, it occurs that the three fastest approaches mentioned above (i.e., the BRIEF, the ORB and the BRISK methods) exhibit similar overall time performance, being significantly faster (approximately 5 times) compared to the SURF algorithm that was used in [6].

The findings related to the object re-detection accuracy of these methods are presented in Table IV. As shown, the SURF method achieves the highest accuracy, while competitive performance is also exhibited by the BRISK and the FREAK algorithms. The remaining techniques (i.e., the BRIEF and the ORB methods) appear to be the most inefficient ones, resulting in a remarkable number of false negatives. Based on the outcome regarding the time efficiency of the evaluated approaches (as reported in Table III), it can be easily concluded that the BRISK descriptor is the most suitable for this kind of analysis, since it ensures high levels of object re-detection accuracy, similar to the one obtained by the SURF and FREAK descriptors, while being the least time-consuming among them (up to 5 times faster).

E. Comparison of different approaches for matching descriptor vectors

As reported in Section III, the next step of our study included the evaluation of different methodologies for matching the computed descriptor vectors. The considered combinations were (a) the extraction of SURF descriptors and the search of the best matches via a Brute-Force 2-NN search [6], and

TABLE III Time performance of the tested approaches for interest point description

	SURF	BRISK	ORB	FREAK	BRIEF
Descriptor vector size	128	64	32	64	32
Description time (msec)/frame	72.10	7.77	7.93	44.04	4.07
Matching time (msec)/frame	26.83	12.74	9.27	7.37	13.68
Total time (msec)/frame	98.93	20.51	17.20	51.41	17.75

TABLE IV Object re-detection accuracy of the tested approaches for interest point description

	SURF/SURF			SURF/BRISK			SURF/ORB			SURF/FREAK			SURF/BRIEF		
	zi	zo	r/o	zi	zo	r/o	zi	zo	r/o	zi	zo	r/o	zi	zo	r/o
Object 1	✓	✓	✓	✓	✓	✓	X	X	X	✓	X	✓	X	X	X
Object 2	✓	✓	✓	✓	✓	✓	X	X	✓	✓	✓	✓	X	✓	✓
Object 3	✓	✓	✓	✓	✓	X	X	✓	X	✓	✓	✓	X	✓	✓
Object 4	✓	✓	X	✓	✓	X	X	✓	X	✓	✓	X	X	✓	✓
Object 5	✓	✓	✓	✓	✓	✓	X	✓	✓	✓	✓	✓	✓	✓	X

(b) the extraction of BRISK descriptors and the search of the approximate best matches utilizing the LSH algorithm.

Our first experiment focused on the efficiency of each approach expressed by the number of defined correspondences (i.e., matches) between the paired sets of descriptors before and after filtering outliers via the applied geometric verification step. Specifically, based on the evaluation procedure described in [28] we measured (a) the number of initially matched descriptors (denoted as IM from now on) after applying the 2-NN search and the distance ratio test, and (b) the number of filtered matches (denoted as FM from now on) after the geometric verification via applying the RANSAC algorithm. Then, we computed the percentages of initially matched pairs of descriptors and the percentages of correct matches that passed the RANSAC test, using the formulas below:

$$\text{Match Rate } (\%) = \frac{\text{IM}}{\min(O_p, F_p)} \times 100$$
$$\text{Correct Match Rate } (\%) = \frac{\text{FM}}{\text{IM}} \times 100 \quad (2)$$

where O_p and F_p is the number of detected interest points by the SURF algorithm in the object of interest (O) and the video frame (F) respectively.

The results of this experiment are reported in Table V. As shown in this table, the extraction of BRISK descriptors and their matching using the LSH indexes results in equivalent or higher percentages (up to 10 % in the most challenging case of rotated and/or occluded instances) of both initial and correct matches compared to the corresponding approach that relies on SURF descriptor extraction and Brute-Force matching. This finding enhances the belief that BRISK descriptors are more suitable for image matching purposes compared to SURF, a conclusion that becomes more important considering the improvement in terms of required processing time, as will be reported below (see Table VI).

The low percentages of matched descriptors reported in Table V are explained by the fact that only a restricted subset of the extracted descriptors from the object of interest can be matched when pairing the object with a video frame that includes a scaled/rotated/occluded instance of it, and not with just a transformed (i.e., scaled and/or rotated) instance of the object, as is the case in [28]. Specifically, a zoomed-in instance may be missing some parts of the object of interest, and so the descriptors extracted from these parts of the object cannot be matched with any of the descriptors extracted from the zoomed-in instance. The same applies for the video frames containing occluded instances of an object. On the other hand, a zoomed-out instance covers only a small fraction of the overall video frame, so the extracted descriptors from the object of interest are matched with a very small set of descriptors that were extracted from this specific part of the video frame.

For evaluating the time efficiency of each combination we used the same dataset as before (i.e., the 5 objects and 3 frames for each of them, depicting scaled and rotated/occluded instances of the objects) and we measured the required time for matching the computed descriptors. Moreover, the object re-detection performance of each method was also evaluated. The outcomes of these experiments are presented in Table VI. These results indicate that the combination of BRISK descriptors with the LSH-based matching strategy can achieve competitive performance with the method of [6] in terms of object re-detection accuracy. However, the significant speed-up of the descriptor matching process (2.5 times faster analysis) justifies the employment of LSH as a suitable approach for matching the extracted BRISK descriptors.

F. Accelerating the object re-detection via prior analysis of the video content

Having made evaluations and decisions about each specific part of the analysis pipeline we studied to which extent a prior analysis of the video could further accelerate the re-detection of the object throughout the video. This pre-processing part could be performed off-line (i.e., prior to the semi-automatic (on-line) object-specific labeling of the video by the video editor), in order to extract information that is used during the re-detection of the specified object in the video frames. So,

TABLE V Performance comparison of the different methodologies, in terms of percentages of matched pairs of descriptors before and after filtering outliers

	SURF/BF					
	zi		zo		r/o	
	Match rate (%)	Correct match rate (%)	Match rate (%)	Correct match rate (%)	Match rate (%)	Correct match rate (%)
Object 1	8.02	91.67	7.35	100.00	6.68	96.67
Object 2	10.51	86.32	1.77	100.00	7.30	84.85
Object 3	0.88	66.67	14.62	98.00	1.61	54.55
Object 4	6.26	66.04	25.97	94.55	6.26	58.49
Object 5	17.19	82.24	29.07	94.55	3.96	42.86
Average	8.57	78.58	15.76	97.42	5.16	67.48
	BRISK/LSH					
Object 1	10.91	91.84	8.69	100.00	7.57	100.00
Object 2	11.06	91.00	1.99	83.33	7.19	87.69
Object 3	0.88	66.67	20.32	98.56	1.02	71.43
Object 4	7.08	58.33	32.00	95.57	8.15	52.17
Object 5	19.57	87.86	37.90	97.61	4.52	77.50
Average	9.90	79.14	20.18	95.02	5.69	77.76

TABLE VI Performance evaluation, in terms of needed time (in msec), of the two tested combinations for object re-detection

Combination	SURF-SURF-BF		SURF-BRISK-LSH	
Object	Matching time	Correct detection	Matching time	Correct detection
1 zoomed in	17.72	✓	14.45	✓
1 zoomed out	17.64	✓	12.61	✓
1 rotated/occluded	19.04	✓	12.55	✓
2 zoomed in	21.12	✓	9.98	✓
2 zoomed out	17.26	✓	7.57	✓
2 rotated/occluded	13.26	✓	8.75	✓
3 zoomed in	64.50	✓	9.42	✓
3 zoomed out	5.82	✓	5.56	✓
3 rotated/occluded	44.50	✓	6.17	X
4 zoomed in	46.61	✓	19.13	✓
4 zoomed out	17.96	✓	13.21	✓
4 rotated/occluded	40.24	X	13.13	X
5 zoomed in	23.51	✓	14.20	✓
5 zoomed out	17.59	✓	13.16	✓
5 rotated/occluded	37.19	✓	13.15	✓
Average	26.93		11.54	
SD (σ)	15.95		3.59	

this off-line analysis could involve the detection of interest points and the extraction of descriptor vectors for each frame of the video, and the storage of this information in a way that facilitates its use during the on-line analysis. For this, we initially used 6 videos from the dataset and we counted the average number of frames that can be processed per second during the re-detection of the object throughout the video. The evaluated methodologies included (a) the use of the SURF algorithm for both detection and description of interest points, (b) an accelerated version of (a) using GPU-based (where GPU stands for Graphics Processing Unit) parallel computing, (c) a combination of GPU-based parallelized SURF interest point detector and CPU-based BRISK descriptor extractor, and (d) the loading of binary files with the pre-computed BRISK descriptors. Aiming to measure only the time needed for interest point detection and description, we did not take into account neither the time needed to read each frame of the video nor the time spent for transferring data between CPU and GPU.

According to the findings reported in Table VII, the use of GPU-based parallel processing can accelerate the corresponding CPU-based implementation 6 times, resulting in a faster-than-real-time analysis. Moreover, the replacement of the GPU-based parallelized SURF algorithm for descriptor extraction by the related CPU-based BRISK method accelerates further (by a factor of 1.5) the analysis, making it over 2

TABLE VII Average processing frame rate for different on-line interest point detection and description combinations in comparison to loading files with pre-computed descriptors

Video resolution	SURF (frames/sec)	GPU_SURF (frames/sec)	GPU_SURF & BRISK (frames/sec)	Loading bin. files (frames/sec)
720x480	6	36	55	326

times faster than real-time processing. However, the loading of binary files with the pre-computed descriptors for the entire set of video frames increases rapidly the processing speed, making the analysis 13 times faster than real-time processing.

After this clear indication about the effectiveness of having pre-analysed the video content, we made a number of experiments for evaluating the time performance of the entire object re-detection pipeline, using again the dataset of Fig. 4. In particular we compared the required processing time, (expressed as a factor of real-time processing with a value below 1 indicating faster than real-time analysis), of the following four approaches: (a) the method of [6] when only CPU-based processing is employed, (b) the new designed method that combines SURF interest point detector with BRISK descriptor extractor and LSH-based matching, (c) the method of (b) when a prior video analysis is performed, and (d) the method of [6] which takes advantage of GPU-based parallel processing and also enables faster-than-real-time-analysis. The results of this evaluation are presented in Table VIII.

As shown in this table, the findings of our study regarding the time performance of the core analysis parts of the object re-detection pipeline led to a new combination that employs the SURF algorithm for interest point detection, the BRISK algorithm for interest point description and the LSH algorithm for descriptor matching. When only CPU-based processing is used, the developed method is over 2 times faster than the corresponding method that relies on the SURF algorithm for interest point detection and description and the Brute-Force approach for matching them. Moreover, our idea about pre-processing the video frames during an off-line analysis, and then loading the pre-computed data (i.e., interest points and descriptor vectors) for re-detecting the object in the video frames during the on-line analysis (i.e., the re-detection of the given object in the video frames), has proven to be quite effective resulting in a considerable reduction of the required time for re-detection. Specifically, the needed processing time is 4 to 5 times smaller compared to the time needed if this pre-processing step was not performed, while it is 2 times faster than the algorithm of [6] which employs GPU-based parallel processing.

G. Overall performance comparison

After experimentally justifying each parameter of the analysis pipeline and every decision that needed to be made before the definition of the finally proposed framework for object re-detection in videos, we extensively evaluated its performance both in terms of time efficiency and detection accuracy. For this, we used the large set of different objects and videos described in Section IV-A. The proposed approach was compared against the methods of [6] and [41], where for the latter we used the software provided by the authors, which takes as input the coordinates of a bounding box that spatially demarcates the first occurrence of the object in the video and re-detects it in the subsequent video frames. For this reason the performance of this technique was evaluated using a subset of our dataset, composed by 43 objects for which only one instance was used for the re-detection.

The results of this experiment are shown in Table IX and indicate that all the evaluated methods exhibit high Precision scores (over 97 %), which means that a remarkably low number of false positives was detected by each one of them. However, the Recall scores reported in Table IX clearly show that the detection accuracy of the algorithm from [41] is significantly lower compared to the accuracy of the other methods. Specifically, approximately 50 % of the objects' occurrences in the video frames were successfully detected by this approach, while the corresponding values for the proposed framework and the method of [6] are 85 % and 90 % respectively. The F-Score values reported in Table IX as a measurement about the overall performance of each approach, highlight the effectiveness of the developed approach and the algorithm of [6] in fulfilling the accuracy requirements of the object re-detection task. As illustrated by the indicative examples of Fig. 6, the introduced algorithm is able to accurately detect and demarcate an instance of the given object in the video frames, that is shown under a variety of different viewing conditions which may combine scaling (i.e., zoom-in/out), rotation (both vertical and horizontal) and partial occlusion of the object. However, it should be noted that great changes in viewing distance and angle may lead to non-detection of the object's appearances in the video frames, resulting at a number of false negatives which is reflected by the reported Recall score for the proposed method. Nevertheless, as will be presented in Section V, the developed tool allows the user to select and use more than one instances of the object during the re-detection and labeling of the video, which eliminates this problem leading to a remarkably limited number of non-detected occurrences of the object in the video frames (the reader is referred to Fig. 12 of Section VI-B).

Regarding the time performance, the proposed algorithm is the most efficient one despite the fact that the entire re-detection process runs using CPU-based processing only. Specifically, the algorithm is approximately 6 times faster compared to [6] (which employs GPU-based processing) and approximately 100 times faster than [41], requiring for analysis only 3 % of the video duration (average value). The wide range of time values reported in the last column of Table IX is explained by the varying number of objects' occurrences in the video, which directly affects the number of shots and video frames where a more detailed analysis must be performed by

TABLE VIII Performance evaluation, in terms of needed time as a factor of real-time processing (a value below 1 indicates faster than real-time analysis), of the four tested combinations for object re-detection.

	# frames depicting the object	Processing time (as a factor of real-time processing)			
		SURF/SURF BF/CPU	SURF/BRISK BF/CPU	SURF/BRISK/LSH (Pre-computed data)	SURF/SURF BF/GPU
Object 1	4914	0.455	0.198	0.065	0.090
Object 2	4256	0.312	0.208	0.027	0.077
Object 3	1819	0.289	0.119	0.026	0.062
Object 4	1648	0.277	0.136	0.036	0.089
Object 5	1428	0.592	0.091	0.038	0.074
Average	-	0.385	0.150	0.038	0.078
SD (σ)	-	0.126	0.047	0.015	0.011

TABLE IX Performance comparison between the developed framework and the algorithms of [6] and [41]

	Precision	Recall	F-Score	Time ($\times$ real-time)
proposed approach	0.999	0.851	0.919	0.006 - 0.084
method of [6]	0.997	0.909	0.951	0.038 - 0.445
method of [41]	0.976	0.496	0.658	0.742 - 7.005

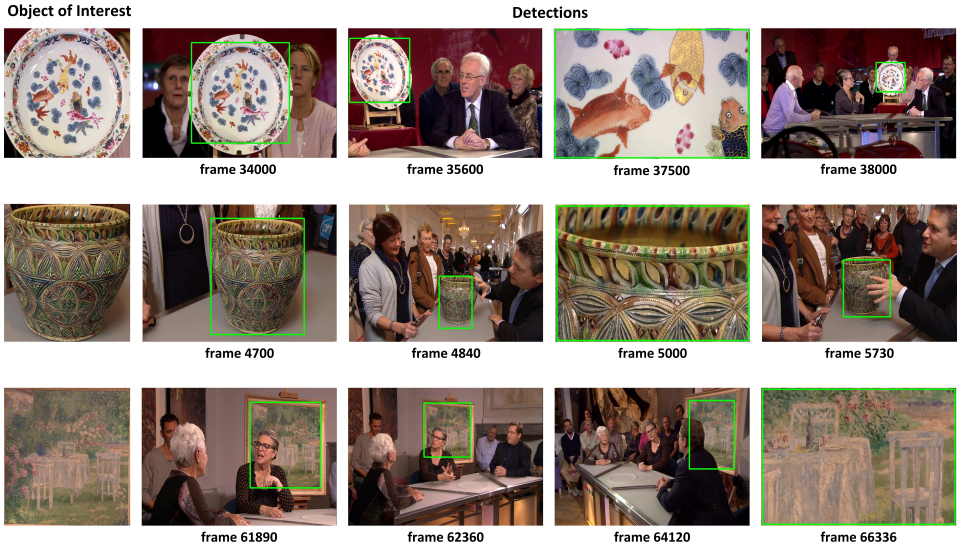


Fig. 6 Objects of interest (left column) and their detected instances (demarcated by the green bounding boxes) in the video frames, shown under different conditions such as zoomed in/out, occluded, and occluded-rotated

the proposed framework and the algorithm of [6] (according to the applied video-structure-based frame-sampling strategy). Other factors that also affect the execution time of these methods are the number and size of the object's instances that are given as input. Concerning the technique of [41], its time performance is directly influenced by the number of detected interest points from the given object that need to be matched and tracked over sequences of frames.

V. Interactive Tool for Object-based Video Labeling

Driven by the constantly increasing needs for having online available media annotated with text labels that make these media accessible via text-based searching and enable the establishment of hyperlinks between related content, and motivated by the lack of tools for time-efficient and accurate instance-based labeling of videos, we developed a web-based framework that can be used for semi-automatic object-specific annotation of videos and thus, the creation of object-related and labeled spatiotemporal media fragments. The users of this tool can be professionals and experts from the area of media editing, enrichment and hyperlinking, that aim to integrate all the available information to a piece of media in order to make it easily accessible, more simply conceivable and thus, more widely consumable. Moreover, a slight extension of this tool that would allow the user to set as input the image of an

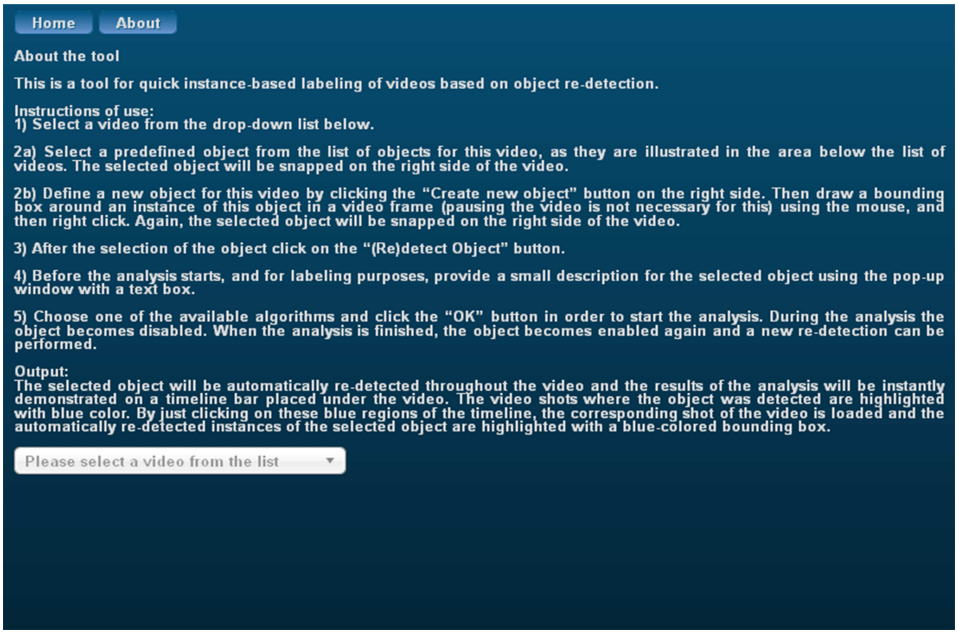


Fig. 7 Screenshot of the initial screen of the tool, containing brief instructions-of-use and a drop-down list for selecting a video

object (instead of selecting it from one of the video frames) and then perform a quick search for occurrences of this object in the video would make it a useful tool for object-based video retrieval and video forensics (e.g., searching for the existence of inappropriate content via the appearance of brand names and logos). Nevertheless, the developed tool can be also used by amateur users which may need to process and categorize personal collections of videos based on objects or other visual features that appear within them. No prior knowledge about the underlying object re-detection algorithm is required for using the tool. The developed graphical user interface (GUI) is based on an interactive browser-based application with a video player which uses media fragment URI references with the new HTML5 video tag. The application allows the selection of instances of the object either during playing the video or after pausing it by using JQuery.

The interface of the developed tool for instance-based spatiotemporal annotation of videos is the one illustrated in Fig.7. This initial screen briefly informs the user about the usage of the tool and the options that are given. By following these instructions a user can select one video from the drop down list that is placed on the left side, right below the box with the instructions-of-use. After selecting a video the video player of the tool appears supporting all standard functionalities such as play/pause the video, adjust the sound level and toggle the video in a full screen mode, while for testing purposes a set of example objects for this video is provided at the upper side of the video player (see Fig. 8). The user can select any of these objects simply by clicking on it, and then the object automatically moves to the right side of the video player. After pressing the "Re-detect Object" button that is enabled for this object, the re-detection of the latter throughout the video starts. In an actual video editing scenario, after selecting a video the user is prompted to specify the object that will be re-detected and spatially demarcated throughout the video. The selection of the object can be done by using the mouse and drawing a bounding box around an instance of the object in one of the video frames, either during playing the video or after pausing it (see Fig. 9a). After this initial selection is done, the tool allows the user to adjust the position and the size of the bounding box in order to end up with the most appropriate (i.e., the most accurate) spatial demarcation of the object's instance. When the spatial re-arrangement of the bounding box ends, the selection of the object's instance is performed simply by right-clicking on it, so the selected area is snapped on the right size of the video player and a pop-up window appears prompting the user to enter a brief description about the object (see Fig.9b). This description will be used as tag during the video labeling process. The instance selection process can be repeated as many times as the user wants in order to define additional instances of the object of interest that are necessary for its re-detection. The latter applies in the case of 3-dimensional objects that appear under different viewing positions and all different views of these objects need to be re-detected and labeled, while it can be also proven useful for re-detecting 2-dimensional objects that appear in the video frames under a widely different viewing distances and angles. If a description about the object was given during the selection of the first instance, then the user can just skip this step and all the additionally defined instances of the object will have the same description as the first one. Finally, the re-detection of the manually selected object using the entire set of the user-defined instances of it, is initiated when the user presses the "OK" button in the tag-related pop-up window (see Fig.9a).

The first check during the analysis is about the number of interest points that can be extracted from the selected

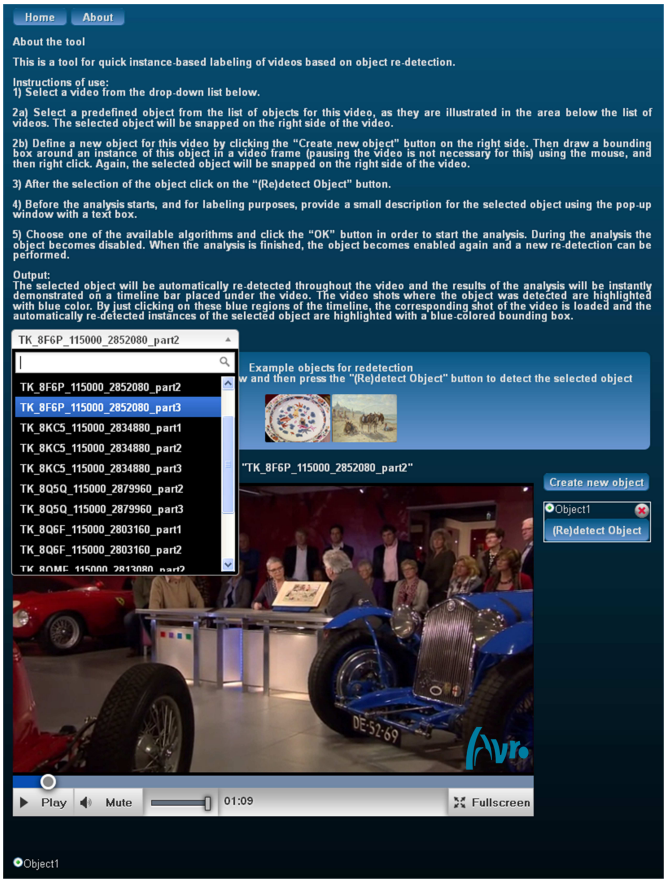


Fig. 8 Screenshot of the tool illustrating the drop-down list with the available videos, the video player and the area with the provided example objects, placed on the upper side of the video player

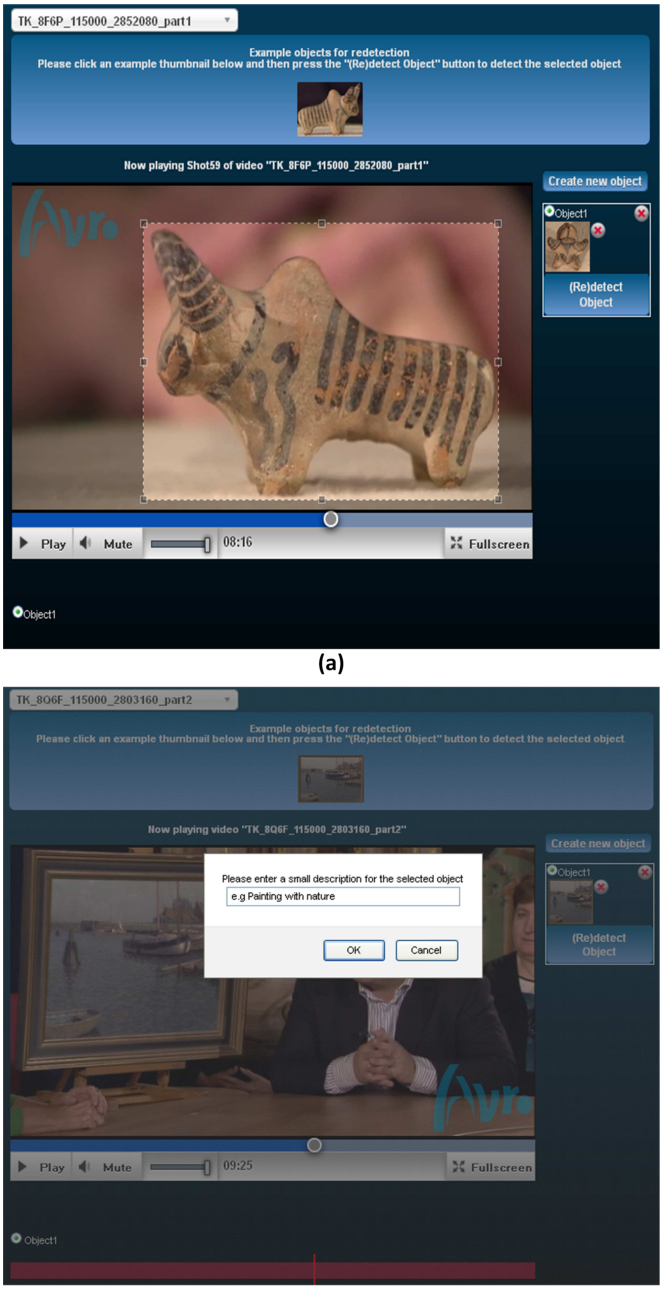
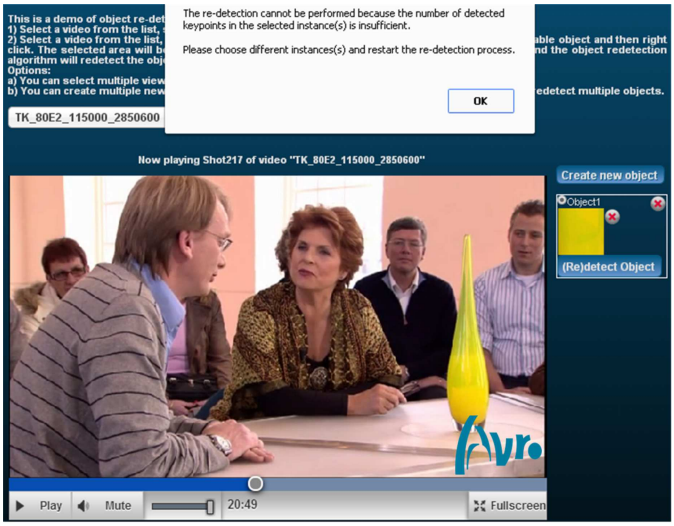


Fig. 9 Screenshots of the tool illustrating: **(a)** the selection of object's instances from the video frames, and **(b)** the pop-up window that appears before the re-detection starts, prompting the user to enter a brief description about the selected object

instance(s) of the object. As described in Section III, the number of detected interest points defines the number of extracted descriptor vectors that are going to be used during the matching of the object and the video frames for its re-detection. If this number is very small (< 4), as in the case of chromatically uniform objects with lack of edges and details, the selected instance cannot be used for re-detection; if the latter is the only one selected by the user, a pop-up window automatically appears informing him/her about this restriction and prompting the selection of a different instance of the object (see Fig.10a where a part of a yellow glass jar was selected by the user for re-detection), otherwise this specific instance is disregarded and the re-detection of the object is performed using the remaining user-defined instances of the object that meet this requirement. In a different case, the processing of the video frames starts by loading the file with the shot segments for this video and a red timeline bar automatically appears under the video player of the tool.

The analysis is performed in a shot-by-shot manner, starting from the shot where the last instance of the object was selected from. When the algorithm finishes with the processing of the final shot of the video, it moves to the first shot and continues the processing until reaching the shot right before the first analysed shot of the video. This shot-by-shot processing can be seen via the color change in parts of this timeline bar, since during the analysis the parts of this bar that correspond to video shots with detected instances of the object are highlighted with dark blue color, while the parts with no detected instances of the object within them are colored by grey color (see Fig. 10b). By selecting any of these dark blue regions of the timeline bar the user moves to the corresponding shot of the video. After pressing the play button of the video

player the re-detected instances of the object in the frames of this shot are shown highlighted by a green bounding box, while a tag including the user-defined description for this object appears in the upper right corner of the player whenever a re-detected instance of the object appears (see Fig.10b).



(a)

(b)

Fig. 10 **(a)** A case where the object re-detection cannot be performed due to limited number of extracted descriptors, and the message that pops-up in order to inform the user, and **(b)** re-detection in progress where the timeline bar indicates the current status; blue regions represent shots with detected instances of the object, gray regions represent shots without detected instances of the object, and red region represents shots that have not been processed yet

During or after the analysis of a video for the re-detection of an object, the user can select and re-detect another, entirely new object from this video. For doing so, the user must press the "Create new object" button from the area in the right side of the video player and then iterate the selection and the re-detection procedure described above for this new object. After this, an additional timeline bar will be shown right below the one related to the last selected object and the analysis results for this new object will be presented in this bar, allowing the user's navigation to parts of the video that contain re-detected instances of this new object.

VI. User Study

Aiming to evaluate the efficiency of the developed interactive tool for object-specific labeling of videos and to further assess the performance of the proposed object re-detection approach, we performed a user study. In this study the implemented object re-detection method was compared against 3 different techniques for object tracking in videos, in terms of required processing time and re-detection efficiency, through a set of user trials. Afterwards the participants were requested to fill-in a questionnaire, providing feedback about the performance of each tested algorithm and a number of different aspects that indicate their overall experience with the tool.

A. User study setup

After incorporating the proposed method for object re-detection (denoted as method 1 in the sequel) in the developed web-based framework, the latter was further extended by integrating implementations of 3 techniques for object tracking in videos (found in the BoofCV [4] library for real-time computer vision applications). In particular we considered the Sparse Flow tracker which is a modification of the pyramidal KLT tracker [9] (denoted as method 2 in the sequel), the Circulant tracker [23] (denoted as method 3 in the sequel), and a fast variation of the recently proposed TLD tracker [25] (denoted as method 4 in the sequel). Similar to the majority of the introduced object tracking techniques from the relevant literature, these methods process sequentially the entire set of video frames for extracting motion information, a procedure that may correspond to more than 15 minutes of waiting time when analyzing a 45-minutes video from our dataset, using a fast technique. Based on this observation, for enabling a concrete and extensive evaluation regarding the efficiency of the proposed tool and the performance of the integrated methods, we defined a set of smaller videos in order to ensure reasonable time requirements for each user trial. This new subset consists of 14 video fragments created after manually segmenting the original videos into pieces that correspond to the presentation of 3 to 4 different objects of art. The duration of these smaller videos varied between 13 and 18 minutes. For each of these video segments we defined a set of 2 to 3 indicative objects for re-detection and we provided it to the users for guidance and testing purposes before the beginning of the main trial period.

Besides the quantitative evaluation of the effectiveness of each tested method (the outcomes of which are reported in the following Section VI-B, based on the questionnaire described

[4] http://boofcv.org/index.php?title=Main_Page

in [12] we prepared a questionnaire for user interface satisfaction (QUIS) in order to record and analyze the participants' feedback regarding the performance of each tested approach, as well as their viewpoint on a variety of aspects that are related to the usability of the tool. The latter refers at (a) the ease-of-use and the interactivity of the developed interface, (b) the accuracy of the object re-detection algorithm and (c) the overall performance of the tool. We adjusted the original questionnaire by removing parts of it that were not applicable to our tool and by keeping only the questions that were relevant in order to meet the needs of our tool's evaluation purposes. Additionally, we modified the 10 scales used in QUIS to 5 in order to make it easier for our user study participants to answer the questions. The findings of the statistical analysis of the users' feedback are presented in Section VI-C.

The participants in this user study were 10 research assistants (8 male, 2 female) between 24 and 33 years old, from the Information Technologies Institute of the Centre for Research and Technology, Hellas. 5 of them reported no previous knowledge or experience with object re-detection or tracking algorithms, 2 of them mentioned limited experience, while the rest were familiar with this research field.

During the first part of the user study, each participant was given time to experiment with the tool in order to get familiar with it, and then was requested to perform 3 different pairs of runs. In each run the participants had to select a video from the drop-down list of the tool and perform a complete annotation of this video based on a selected object, via re-detecting the entire set of its re-occurrences in the video frames. This object could be either one of the example objects of this video or an entirely new one that was selected by the participant by applying the procedure described in Section V. For the re-detection of the selected object throughout the video two different approaches had to be used in each run; the first was the algorithm proposed in this study, i.e., method 1 of the tool, while the second was one of methods 2 to 4 of the tool. Via this blind (the algorithms appeared in the interface with the names "Algorithm 1", "Algorithm 2" etc. and not with their actual names in order to make sure that the participants will not be influenced from previous relevant experience or knowledge) pairwise testing approach, the participants were able to make a direct performance comparison between the considered pair of techniques, since these techniques were used for addressing the exact same task. Due to the fact that some of these techniques fail to re-detect instances of the object after analysing a sequence of frames that either does not contain any instance of it, or includes re-occurrences of the object shown under different viewing conditions, we asked the participants to re-apply the analysis (i.e., the selection and re-detection of another instance of the object) as many times as needed in order to re-detect all the different appearances of it throughout the video. In this case, the result of every new "run" of the selected algorithm was added to the result(s) of the previous run(s), while the fragments of the video where the object was previously found were not examined again. Finally, the number of runs made by each participant for re-detecting a particular object in the video frames using a specific approach, as well as the required time for completing the annotation of the entire video based on the re-detection of this object, were logged and used for evaluating the efficiency of the video labeling tool and the re-detection accuracy of each considered approach. The outcomes of this assessment are presented in Section VI-B. Moreover, aiming to compare the performance of the proposed algorithm (considering both analysis time and re-detection accuracy) against the efficiency of the other object tracking methods in a way similar to the evaluations presented in IV-G, the analysis results of a "one-click annotation" process where the user was allowed to select only one instance of the object and use it for its re-detection throughout the video were also recorded and used for further processing. The conclusions of this evaluation are also reported in the following Section VI-B.

The second part of the user study involved the answering of the designed QUIS questionnaire. Firstly, each participant had to fill-in a set of questions about his/her experience after using the tool, reporting any prior knowledge on algorithms and tools for object re-detection in videos, assessing the completeness of the provided instructions-of-use and giving an overall description regarding his/her practice with the tool. Following, there was a set of questions related to the usability of the developed user interface, where each participant had to report his/her opinion about the spatial arrangement of the provided information, the speed of interaction with the tool, the clarity of the pop-up messages of the tool and any kind of bugs during the use. The next block of questions was relevant to the performance of the integrated object re-detection approaches, where the participants had to qualitatively evaluate the detection accuracy and the time efficiency of each tested technique, and to decide about the most effective among them. The last part of the questionnaire focused on the participants' perspective regarding the overall usability of the tool and its efficiency on addressing the task of object-based video labeling, where a qualitative assessment about the overall experience with the tool, as well as potential suggestions, were requested.

B. Quantitative analysis of video labeling results

The analysis of the recorded results after the "one-click annotation" procedure validated the effectiveness of the proposed approach and highlighted its superiority against the other tested techniques. As reported in Table X the performance of the proposed approach in terms of Precision, Recall and F-Score is remarkably high and slightly better than the one reported in Table IX. The algorithm was able to re-detect occurrences of the given objects in the video frames, exhibiting robustness in scaling (i.e., zoom-in/out), rotation (both horizontal and vertical) and occlusion of the object as illustrated in the upper two objects of Fig. 11, and resilience to slight changes in illumination, as depicted in the bottom two objects of Fig. 11. Regarding the other methods, the Sparse Flow tracker is the only one that exhibits very high Precision score, which means a very limited number of erroneous detections, however the computed Recall score indicates its very poor performance in detecting the appropriate occurrences of the object in the video frames, which decreases dramatically the

TABLE X Performance comparison between the proposed object re-detection approach and the three object tracking algorithms that were integrated into the developed tool for object-specific video labeling, after applying the "one-click-annotation" process

	Precision	Recall	F-Score	Time (× real-time)
proposed approach	0.999	0.891	0.942	0.001 - 0.150
Sparse Flow tracker	0.985	0.055	0.104	0.356 - 0.391
Circulant tracker	0.122	0.220	0.157	0.397 - 0.441
fast TLD tracker	0.224	0.077	0.114	0.352 - 0.472

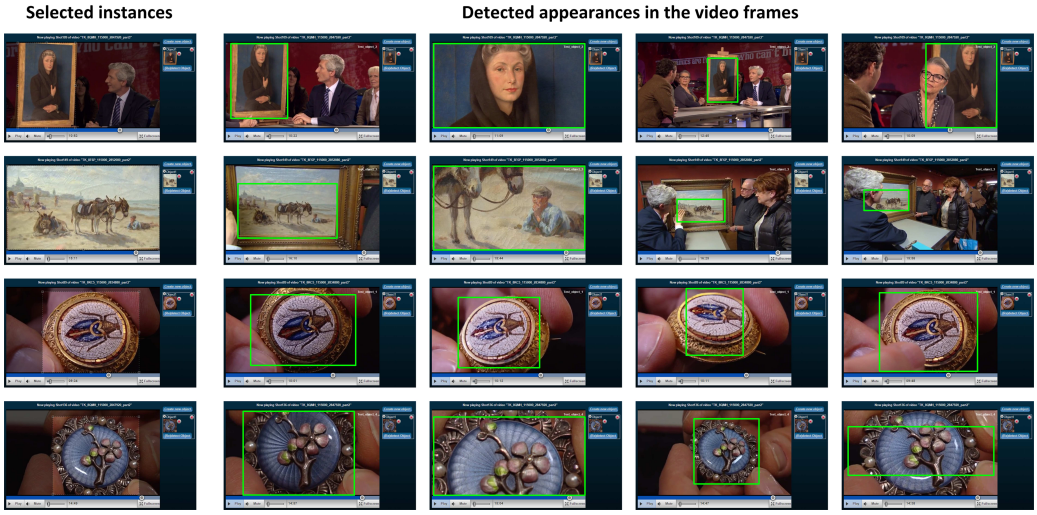


Fig. 11 Selected objects using the crop function of the developed tool (left column) and their detected instances (demarcated by the green bounding boxes) in the video frames, shown under different viewing conditions that may combine scaling, rotation (both vertical and horizontal), occlusion and changes in illumination

overall F-Score of this method. The Circulant and the fast TLD tracker appeared to be very sensitive to false alarms, something that is proven by their very low Precision scores, while they also seemed to be incapable to correctly detect the object in the video frames, resulting in quite small Recall values.

The superiority of the proposed method over the other tested algorithms is also confirmed in terms of needed processing time, since it was shown to be 7 to 100 times faster than real-time processing, depending on the number of object's instances in the video frames, while the other methods are approximately 2.5 times faster than real-time analysis. Please note that the slightly increased values regarding the time performance of the proposed method compared to the values reported in Table IX are explained by the fact that the used video fragments in these experiments contain a significantly larger number of object's occurrences as percentage of the number of video frames, compared to the videos that were used in the evaluations reported in Section IV-G. The latter means that the gain in running time is much bigger when analysing larger videos, where the algorithm filters out considerably more fragments that do not contain occurrences of the object, and thus accelerates even more the analysis built on the structure-based sampling of the video frames.

Nevertheless, as stated in Section IV-G the introduced algorithm fails to detect appearances of the given object that are shown after major changes at the viewing distance (i.e., significantly zoomed-in/out instances) and angle (i.e., horizontally rotated instances). As illustrated in the upper part of Fig. 12 a number of occurrences of the given objects were not detected through the "one-click annotation" process due to great differences in scale and viewing angle between the selected instances and their appearances in the video frames, combined in some cases with partial occlusion which makes the re-detection process an even more challenging task. For addressing this issue, the developed tool enables the selection of more than one instances of the object of interest (as described in Section V) and the use of these additional instances during the re-detection of the object and the spatiotemporal annotation of the video. The latter results in successful detection and demarcation of even the most extreme variations of these objects in the video frames, as depicted by the lower part of Fig. 12, leading to a zero number of false negatives (i.e., Recall score equal to 1) without affecting the algorithm's Precision (as reported in Table XI).

Concerning the processing time for object re-detection when more than one object's instances are used, the extra time required for the selection of additional instances (which is expected to be limited for a video editor with a detailed knowledge about the video content) is partially gained back due to faster analysis. The latter is explained by the application of less time-demanding operations for the RANSAC-based geometric validation of the matched pairs of descriptors during

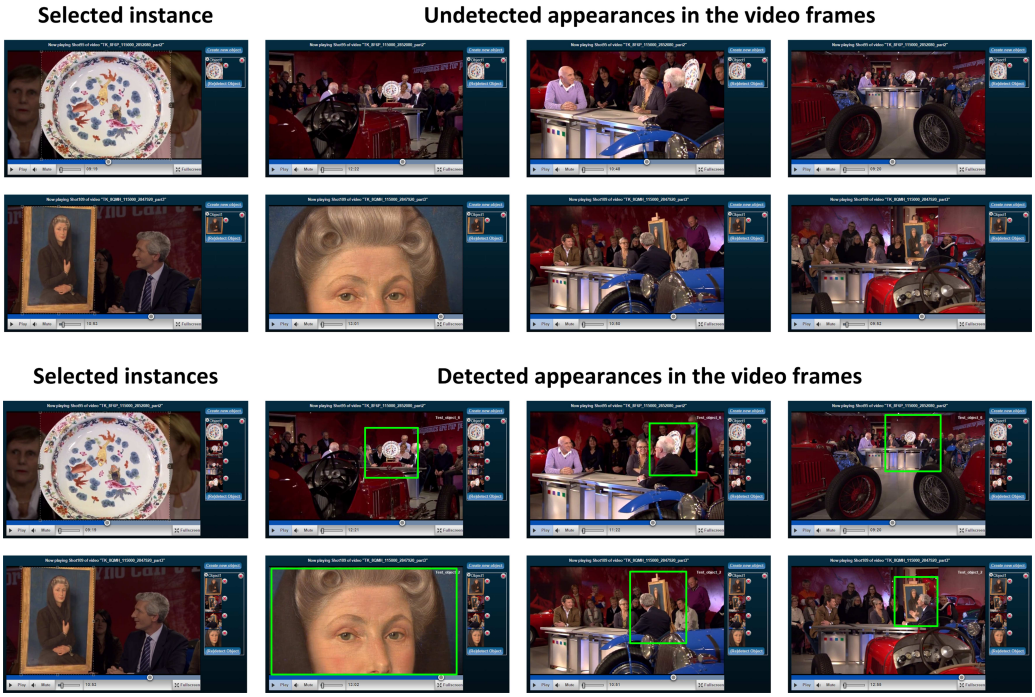


Fig. 12 Upper part: selected objects using the crop function of the tool and their non-detected instances in the video frames due to great changes in viewing distance and angle. Lower part: selected sets of objects' instances using the crop function of the tool and their successfully detected instances in the video frames, demarcated by the green bounding boxes

TABLE XI Performance comparison in terms of Precision, Recall and Processing time when the re-detection of 5 objects in the video frames is based on a single instance (Case 1) and a set of 2 to 4 different instances (Case 2)

	Case 1 (only 1 instance is used)			Case 2 (2 - 4 instances are used)		
	Precision	Recall	Processing time (msec / x real-time)	Precision	Recall	Processing time (msec / x real-time)
Object 1	1.000	0.886	39.41 / 0.036	1.000	1.000	39.03 / 0.036
Object 2	1.000	0.914	30.89 / 0.036	1.000	1.000	17.97 / 0.021
Object 3	0.992	0.874	61.89 / 0.068	0.992	1.000	39.33 / 0.043
Object 4	1.000	0.878	21.50 / 0.022	1.000	1.000	10.56 / 0.011
Object 5	1.000	0.772	9.25 / 0.010	1.000	1.000	8.27 / 0.010

the comparison of the appropriate object's instance with the video frames that contain occurrences of this specific instance. Table XI reports the needed processing time for annotating 5 different video fragments using 5 different objects that appear in these videos under extremely varying viewing positions, considering only 1 instance of the object in the first case (case 1 in Table XI) and 2 to 4 different instances in the second case (case 2 in Table XI). As can be seen, the analysis time is similar or even smaller in the second case when a set of instances is used during the analysis, while at the same time the re-detection performance meets the highest requirements for 100 % accurate object-based labeling of videos.

The last quantitative evaluation focused on the required time for performing an accurate and complete annotation of a video based on a selected object. As described in Section VI-A each participant of the user study was requested to select a number of instances of the object of interest and re-apply the analysis as many times as needed, in order to result to a fully annotated video based on the re-detected appearances of this object. This evaluation highlighted the advantages of the proposed object re-detection methodology over the examined techniques (as well as other motion-based approaches from the relevant literature) for object tracking in videos. In particular, the developed approach is able to detect and demarcate occurrences of the given object throughout the entire video, indicating its appearances in parts of the video that are not necessarily contiguous, and can be intermittent by video segments that do not contain appearances of the object. Contrary, the other 3 assessed methods were able to detect the object's instances only in short sequences of video frames starting right after the frame with the user-defined instance and ending to the final frame of the corresponding shot segment of the video, failing to re-detect it in the next shots of the video even if they contain extremely similar instances with the given one.

TABLE XII Performance comparison between the proposed object re-detection approach and the three object tracking algorithms that were integrated into the developed tool for object-specific video labeling, after applying the complete object-based labeling of the videos

	Precision	Recall	F-Score	Time (× real-time)
proposed approach	0.999	1.000	0.999	0.018 - 0.157
Sparse Flow tracker	0.985	1.000	0.992	2.514 - 13.596
Circulant tracker	0.107	1.000	0.193	2.438 - 12.913
fast TLD tracker	0.378	1.000	0.549	2.325 - 13.267

Moreover, the Sparse Flow and the TLD trackers exhibited limited robustness to gradual changes in scale, failing to detect consecutive appearances of the object in the same shot of the video after gradual minor changes in scale. Due to this limited re-detection performance the users had to re-apply the analysis for labeling the entire video, selecting each time a new instance of the object and repeating this process, at least, as many times as the number of the different video segments (i.e., shots) that contain appearances of the object.

For the used dataset the number of performed iterations for labeling these videos using one of the evaluated object tracking approaches varied between 7 and 44. The data extracted from the files that logged the actions of each participant in the user study are presented in Table XII. The reported values indicate the indisputable advantage of the proposed method over the other object tracking approaches, and demonstrate the efficiency of the developed framework as a tool that enables quick and accurate semi-automatic instance-specific labeling of videos, demanding only a small fraction of the video's duration (varying between 1.8 % and 15.7 %) for analysis. The evaluated object tracking approaches can also lead to high Recall scores (identifying the entire set of objects' occurrences in the video frames via performing iterative instance selection and re-detection), with the Sparse Flow tracker being the only one that can lead to limited number of false alarms. Nevertheless, the required time for completing the object-based labeling of videos with this method is approximately 2.5 to 13.5 times slower than real-time processing, making this approach insufficient for real-time video editing operations.

C. Analysis of users' feedback

After decoding the collected information from the filled questionnaires, the outcomes regarding the usability of the developed tool and the overall experience with the implemented user interface are demonstrated in Fig.13. More specifically, the provided instructions-of-use were judged as "Sufficient/Clear" (average score 4.5 where 1 stands for "Confusing" and 5 stands for "Clear"), while the overall experience of the participants was characterized as "Very Good" (average score 4.1 where 1 stands for "Terrible" and 5 stands for "Excellent"). Concerning the user interface the participants reported that the arrangement of information on the screen was "Sensible" (average score 4.8 where 1 stands for "Confusing" and 5 stands for "Sensible") and the response time of the user interface was "Fast" (average score 4 where 1 stands for "Very Slow" and 5 stands for "Very Fast").

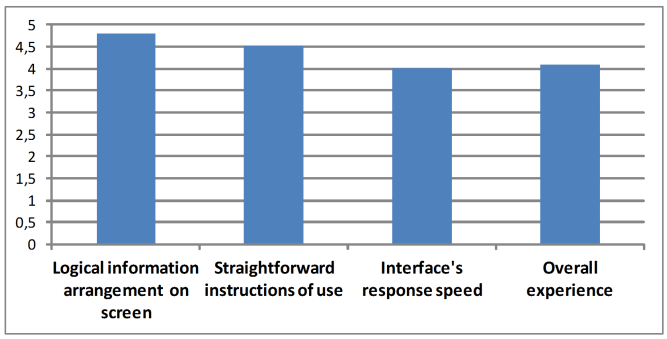


Fig. 13 Participants' feedback (mean values) regarding the usability of the tool

The efficiency of the various tested object re-detection approaches regarding both speed and re-detection accuracy is shown in Fig.14. The best, in terms of time performance, was the algorithm introduced in this study, achieving an average score of 4.7 in the range 1 to 5. Substantially below this, the Sparse Flow and the Circulant trackers followed with a rating of 2.8, while the most time-consuming one was the TLD tracker with a rating of 2.3. Concerning the detection accuracy the proposed method was judged as the best one, getting an average score of 4.4 in the range 1 to 5. The Sparse Flow tracker was rated as the second best with a mean score of 2.9, the TLD tracker with 2.4, while the Circulant tracker got the lowest score of 1.6. The low scores that were assigned by the participants to the Sparse Flow, the Circulant and TLD tracker are explained by the fact that, as mentioned before, after detecting one of the object's instances in a sequence of video frames these methods are very sensitive in the detection of other occurrences that appear in non-consecutive parts of the video or under different viewing positions, either failing to re-detect them (increasing the number of false negatives) or leading to erroneous detections (producing a number of false alarms). As described above, the participants had to re-run these algorithms as many times as the number of different video fragments (i.e., shots) with appearances of the given object in order to re-detect all of them throughout the entire video. As reported previously, this number varied between 7 and 44, with the Sparse Flow tracker being the most demanding one. However, this tracker was the most efficient compared to the other two in terms of Precision causing a zero number of false alarms, contrary to the other object tracking techniques (i.e., the Circulant and the fast TLD tracker) which

TABLE XIII Participants' general feedback (based on the selection from a pre-defined pool of words/phrases) regarding the developed tool

	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10
Attractive				✓	✓				✓	
Complex										
Confusing										
Easy to use	✓	✓			✓			✓		✓
Effective		✓							✓	✓
Frustrating										
Hard to use										
Inspiring								✓		
Straightforward		✓		✓			✓			✓
Time-consuming										
Time-saving			✓	✓	✓					
Useful	✓	✓	✓	✓		✓	✓		✓	✓

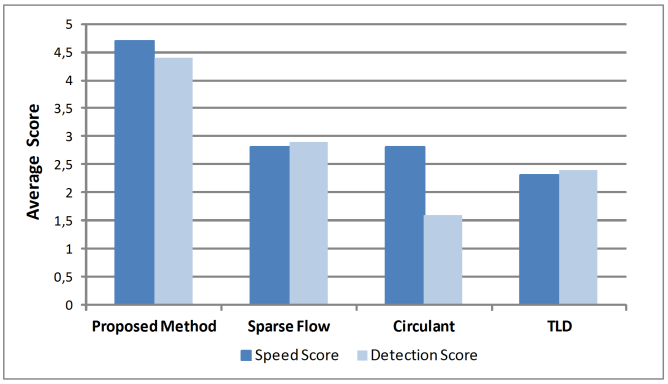


Fig. 14 Participants' feedback (mean values) regarding the efficiency of the tested object re-detection approaches, in terms of processing speed and re-detection accuracy

resulted at a notably large number of false positives that caused the dissatisfaction of the users regarding these algorithms' effectiveness. Based on these values and aiming to determine whether there was a statistically significant difference between the proposed method and the Sparse Flow tracker which was indicated as the second best approach, we performed a Wilcoxon signed-rank test at 5 % significance level for each case, setting as "Hypothesis 0" the fact that there is no actual difference between these techniques. The test statistic for our data was 0 in both cases, while according to the Wilcoxon signed-ranks table the critical value for $a = 0.05$ and $n = 10$ (where n is the number of participants at the user study) is 8. So, the test statistic is way lower than the defined critical value, which means that "Hypothesis 0" is rejected and, thus, there is indeed a statistically significant difference between these approaches.

Concerning the functionality and the usability of the tool, by selecting from a pool of 16 predefined words or phrases given in the questionnaire, 80 % of the participants evaluated it as "Useful", 50 % of them characterized it as "Easy to use", while 30 % of the users described it as "Attractive" and "Effective". The words selected by each user are shown in Table XIII. Furthermore, all users declared that they would suggest the tool to another friend or colleague, while some of them provided useful feedback regarding the interface of the tool. Based on this feedback, we were encouraged to mark the temporal boundaries of each shot of the video on the timeline bar of the tool, in order to distinguish the shot segments of the video and make the navigation of the user easier and more precise.

VII. Conclusions

The objective of this study was twofold; initially we introduced an approach for fast object re-detection in videos, and then, we presented a tool that integrates this method and performs quick and accurate semi-automatic object-based spatiotemporal labeling of videos. The latter is realized through an interactive process that allows the re-detection and demarcation of re-appearances of the given object in the video frames. In particular, aiming to improve the time performance of the algorithm from [6], we replaced the use of SURF descriptors by employing the more lightweight BRISK binary descriptors, and we substituted the time-demanding Brute-Force 2-NN search for matching the computed descriptor vectors with a faster, LSH-based descriptor matching strategy that leads to a significant acceleration of the analysis. Moreover, aiming to achieve a further speed-up of the object-based video labeling process we partitioned the applied workflow into off-line analysis, where the descriptor vectors for the entire set of video frames are extracted and stored, and on-line analysis, where the re-detection of the object throughout the video through time-efficient descriptor matching and via exploiting information about the shot-level structure of the video, is performed. The developed object re-detection technique was integrated into a web-based tool that allows users to select a number of instances of an object that appears in a video by spatially demarcating them in the video frames, and then automatically detect and label re-occurrences of this object in consecutive or non-consecutive frames of the video, thus performing object-based spatiotemporal annotation of the video. The conducted experiments showed that the proposed object re-detection algorithm is capable of several times faster-than-real-time

analysis, while being highly accurate due to its robustness in scale and rotation transformations. Moreover the performance comparisons against other object tracking approaches from the relevant literature illustrated the superiority of the developed method, concerning both the time efficiency and the re-detection precision. Finally, the outcomes of the user study indicated that the developed web-based framework can be considered as a complete approach for object-based video labeling, and for the creation of instance-based spatiotemporal fragments.

VIII. Acknowledgment

This work was supported by the European Commission under contract FP7-600826 ForgetIT and FP7-287911 LinkedTV.

References

[1] Abeles P (2013) Examination of Hybrid Image Feature Trackers. International Symposium on Visual Computing (ISVC)

[2] Agrawal M, Konolige K, Blas M.R (2008) CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching. Computer Vision ECCV 2008, 5305:102-115

[3] Alahi A, Ortiz R, Vandergheynst P (2012) FREAK: Fast Retina Keypoint. IEEE Conference on Computer Vision and Pattern Recognition: 510-517

[4] Andoni A, Indyk P (2008) Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. Communications of the ACM, 51(1):117-122

[5] Apostolidis E, Mezaris V (2014) Fast Shot Segmentation Combining Global and Local Visual Descriptors. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP): 6583 - 6587

[6] Apostolidis E, Mezaris V, Kompatsiaris I (2013) Fast object re-detection and localization in video for spatio-temporal fragment creation. IEEE International Conference on Multimedia and Expo Workshops (ICMEW): 1-6

[7] Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding, 110(3):346-359

[8] Bentley J.L (1975) Multidimensional Binary Search Trees Used for Associative Searching. Communications of the ACM, 18(9):509-517

[9] Bouguet J.-Y (1999) Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm. Intel Corporation Microprocessor Research Labs

[10] Calonder M, Lepetit V, Ozuysal M, Trzcinski T, Strecha C, Fua P (2012) BRIEF: Computing a Local Binary Descriptor Very Fast. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(7):1281-1298

[11] Canclini A, Cesana M, Redondi A, Tagliasacchi M, Ascenso J, Cilla R (2013) Evaluation of low-complexity visual feature detectors and descriptors. 18th International Conference on Digital Signal Processing (DSP): 1-7

[12] Chin J.P, Diehl V.A, Norman K.L (1988) Development of an Instrument Measuring User Satisfaction of the Human-computer Interface. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: 213-218

[13] Chum O, Matas J (2005) Matching with PROSAC - Progressive Sample Consensus. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1:220-226

[14] Chum O, Matas J (2008) Optimal Randomized RANSAC. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(8):1472-1482

[15] Comaniciu D, Meer P (2002) Mean shift: a robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5):603-619

[16] Datar M, Immorlica N, Indyk P, Mirrokni V.S (2004) Locality-sensitive Hashing Scheme Based on P-stable Distributions. Proceedings of the Twentieth Annual Symposium on Computational Geometry: 253-262

[17] Ebrahimi M, Mayol-Cuevas W.W (2009) SUSurE: Speeded Up Surround Extrema feature detector and descriptor for realtime applications. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops: 9-14

[18] Fischler M.A, Bolles R.C (1981) Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. ACM Communications, 24(6):381-395

[19] Fleury M, Self R.P, Downton A.C (2004) Development of a Fine-grained Parallel Karhunen-Loeve Transform. Journal of Parallel and Distributed Computing, 64(4):520-535

[20] Friedman J.H, Bentley J.L, Finkel R.A (1977) An Algorithm for Finding Best Matches in Logarithmic Expected Time. ACM Transactions on Mathematical Software (TOMS), 3(3):209-226

[21] Fukunaga K, Narendra P.M (1975) A Branch and Bound Algorithm for Computing k-Nearest Neighbors. IEEE Transactions on Computers, C-24(7): 750-753

[22] Harris C, Stephens M (1988) A combined corner and edge detector. Proc. of Fourth Alvey Vision Conference: 147-151

[23] Henriques J.F, Caseiro R, Martins P, Batista J (2012) Exploiting the Circulant Structure of Tracking-by-detection with Kernels. Proceedings of the 12th European Conference on Computer Vision, Part IV:702-715

[24] Joly A, Buisson O (2008) A Posteriori Multi-probe Locality Sensitive Hashing. Proceedings of the 16th ACM International Conference on Multimedia: 209-218

[25] Kalal Z, Mikolajczyk K, Matas J (2012) Tracking-Learning-Detection. IEEE Transactions on Pattern Analysis & Machine Intelligence, 34(7): 1409-1422

[26] Kato K, Hosino T (2010) Solving k-Nearest Neighbor Problem on Multiple Graphics Processors. Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing: 769-773

[27] Ke Y, Sukthankar R (2004) PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition: 506-513

[28] Khvedchenya E (2012) A battle of three descriptors: SURF, FREAK and BRISK. http://computer-vision-talks.com/articles/2012-08-18-a-battle-of-three-descriptors-surf-freak-and-brisk/. Accessed December 2014

[29] Korman S, Avidan S (2011) Coherency Sensitive Hashing. Proceedings of the 2011 International Conference on Computer Vision: 1607-1614

[30] Leutenegger S, Chli M, Siegwar R (2011) BRISK: Binary Robust Invariant Scalable Keypoints. Proceedings of the IEEE International Conference on Computer Vision: 2548-2555

[31] Liang-Chi C, Tian-Sheuan C, Jiun-Yen C, Chang N.Y.-C (2013) Fast SIFT Design for Real-Time Visual Feature Extraction. IEEE Transactions on Image Processing, 22(8): 3158-3167

[32] Liu W, Wang J, Ji R, Jiang Y. G, Chang S.F (2012) Supervised Hashing with Kernels. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) (Oral session): 2074-2081

[33] Liu Z, Xing B, Chen Y (2013) An Efficient Parallel SURF Algorithm for Multi-core Processor. Computer Engineering and Technology: 27-37

[34] Lowe D.G (2004) Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 60(2):91-110

[35] Lucas B.D, Kanade T (1981) An Iterative Image Registration Technique with an Application to Stereo Vision. Proceedings of the 7th International Joint Conference on Artificial Intelligence, 2:674-679

[36] Lv Q, Josephson W, Wang Z, Charikar M, Li K (2007) Multi-probe LSH: Efficient Indexing for High-dimensional Similarity Search. Proceedings of the 33rd International Conference on Very Large DataBases: 950-961

[37] Matas J, Chum O, Urban M, Pajdla T (2002) Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. Proceedings of the British Machine Vision Conference:36.1-36.10

[38] Mikolajczyk K, Schmid C (2005) A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(10):1615-1630

[39] Miksik O, Mikolajczyk K (2012) Evaluation of local detectors and descriptors for fast feature matching. 21st International Conference on Pattern Recognition (ICPR): 2681-2684

[40] Muja M, Lowe D.G (2014) Scalable Nearest Neighbor Algorithms for High Dimensional Data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(11): 2227-2240

[41] Nebehay G, Pflugfelder R (2014) Consensus-based matching and tracking of keypoints for object tracking. IEEE Winter Conference on Applications of Computer Vision (WACV):862-869

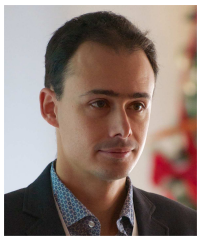
[42] Pan J, Manocha D (2011) Fast GPU-based Locality Sensitive Hashing for K-nearest Neighbor Computation. Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems: 211-220

[43] Romberg S, Lienhart R (2013) Bundle Min-hashing for Logo Recognition. Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval: 113-120
[44] Rublee E, Rabaud V, Konolige K, Bradski G (2011) ORB: An efficient alternative to SIFT or SURF. IEEE International Conference on Computer Vision (ICCV): 2564-2571
[45] Shih-Fu C, Junfeng H, Youngwoon L, Jae-Pil H, Sung-Eui Y (2012) Spherical hashing. IEEE Conference on Computer Vision and Pattern Recognition: 2957-2964
[46] Silpa-Anan C, Hartley R (2008) Optimised KD-trees for fast image descriptor matching. IEEE Conference on Computer Vision and Pattern Recognition: 1-8
[47] Sismanis N, Pitsianis N, Xiaobai S (2012) Parallel search of k-nearest neighbors with synchronous operations. IEEE Conference on High Performance Extreme Computing (HPEC):1-6
[48] Ta D.-N, Chen W.-C, Gelfand N, Pulli K (2009) SURFTrac: Efficient Tracking and Continuous Object Recognition using Local Feature Descriptors. IEEE Conference on Computer Vision and Pattern Recognition: 2937-2944
[49] Tomasi C, Kanade T (1991) Detection and Tracking of Point Features. CMU-CS-91-132, Carnegie Mellon University
[50] Warn S, Emeneker W, Cothren J, Apon A (2009) Accelerating SIFT on parallel architectures. IEEE International Conference on Cluster Computing and Workshops: 1-4
[51] Weiss Y, Torralba A, Fergus R (2008) Spectral Hashing. Advances in Neural Information Processing Systems:1753-1760
[52] Yang D, Liu L, Zhu F, Zhang W (2011) A Parallel Analysis on Scale Invariant Feature Transform (SIFT) Algorithm. Proceedings of the 9th International Conference on Advanced Parallel Processing Technologies: 98-111
[53] Yue L, Deng C, Cheng L (2012) Density Sensitive Hashing. CoRR, abs/1205.2930
[54] Zhang N (2009) Computing Parallel Speeded-up Robust Features (P-SURF) via POSIX Threads. Proceedings of the 5th International Conference on Emerging Intelligent Computing Technology and Applications: 287-296
[55] Zhou H, Yuan Y, Shi C (2009) Object Tracking Using SIFT Features and Mean Shift. Computer Vision and Image Understanding, 113(3):345-352
[56] Zhou K, Hou Q, Wang R, Guo B (2008) Real-time KD-tree Construction on Graphics Hardware. ACM SIGGRAPH Asia 2008 Papers: 126:1-126:11

Anastasia Ioannidou Anastasia Ioannidou received her Diploma degree from the Department of Applied Informatics of University of Macedonia, Greece, and her M.Sc. degree in Digital Media from the Department of Informatics, Aristotle University of Thessaloniki, Greece, in 2010 and 2013 respectively. She is currently working as a Research Assistant at the Information Technologies Institute of the Centre for Research of Technology - Hellas. Her current research interests include image/video processing and analysis, multimedia retrieval, object detection and tracking.

Evlampios Apostolidis Evlampios Apostolidis received a Diploma in Electrical and Computer Engineering from the Aristotle University of Thessaloniki in July 2007, writing a thesis on image processing. Afterwards, he got a Master in Informational Systems from the University of Macedonia in November 2011, writing a dissertation about methods for indexing multi-dimensional spaces. From January 2012 he is a Research Assistant at the Information Technologies Institute (ITI) of the Centre for Research of Technology Hellas (CERTH). His research interests include image and video analysis.

Chrysa Collyda Chrysa Collyda is a postdoctoral research fellow with the Information Technologies Institute at the Centre for Research and Technology Hellas. She received her PhD in Medical Informatics from Medical School of the Aristotle University of Thessaloniki in 2008. Her research interests include bioinformatics, medical informatics, decision support systems, image processing, human computer interaction, internet and multimedia technologies.



Vasileios Mezaris Vasileios Mezaris received the BSc and PhD in Electrical and Computer Engineering from the Aristotle University of Thessaloniki in 2001 and 2005, respectively. He is a Senior Researcher (Researcher B) at the Information Technologies Institute (ITI) of the Centre for Research of Technology Hellas (CERTH). His research interests include image and video analysis, retrieval, event detection, machine learning for multimedia analysis. He served as Associate Editor for the IEEE Transactions on Multimedia (2012-2015) and is a Senior Member of the IEEE.