Spatio-Temporal Video Copy Detection

R. Cameron Harvey
School of Computing Science
Simon Fraser University
Surrey, BC, Canada
cameron_harvey@sfu.ca

Mohamed Hefeeda
Qatar Computing Research Institute
Qatar Foundation
Doha, Qatar
mhefeeda@qf.org.qa

ABSTRACT

Video copy detection algorithms are used to find copies of original video content even if the content has been altered. Given the prevalence of video recording and copying devices as well as the availability of many Internet sites for hosting videos, detecting video copies has become an important problem especially for companies interested in managing and controlling copyrights of their content. We propose a new content-based video copy detection algorithm. The proposed algorithm creates signatures that capture the spatial and temporal features of videos. These spatio-temporal signatures enable the algorithm to provide both high precision and recall. In addition, these signatures require small storage and are easy to compute and compare. Our extensive experimental analysis with a large video dataset shows that the proposed algorithm achieves high precision and recall values while remaining robust to many video transformations that commonly occur in practice. The algorithm is simple to implement and is more computationally efficient than previous algorithms in literature.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering, Search process*

General Terms

Algorithms, Experimentation

Keywords

Video copy detection, Video fingerprinting

1. INTRODUCTION

Camcorders and digital video recorders (DVRs) are readily available and easily affordable. In fact, multimedia technologies have advanced to the point where video recording capabilities are commonly bundled with electronic devices such as digital cameras, cell phones and personal digital assistants (PDAs). Virtually all modern video recorders store data digitally. Moreover, they are built with connection ports to easily connect and upload the recorded content onto a computer. It is a simple matter to further distribute the content from a computer onto social media websites such as YouTube or facebook. More and more people are uploading and sharing video content. This situation creates issues relating to data management. One issue is database optimization. It is inefficient to store multiple copies of the same video in a database as it creates needless infrastructure expenses and complicates search and retrieval algorithms. If we can detect whether a video already exists in the database, we can make more effective use of storage.

Another serious issue relates to copyright infringement. It is relatively easy to copy commercial content and redistribute it over the Internet. This can result in loss of revenue for a business. It is not feasible to manually sift through the countless hours of videos found on the Internet to see if someone has made an illegal copy. There is a real need to use automated video copy detection techniques to detect copyright violations. Video copy detection can also be used to monitor usage. For example, a company pays a television channel for a commercial advertisement. It would like to monitor the channel for when its advertisement is played and how often. It can use these data to confirm that contract terms have been met.

There are two fundamental approaches to video copy detection: watermarking and content based. Watermarking techniques embed information into the video content. This information is unique to the video and under normal circumstances invisible. Copy detection becomes a matter of searching the video content for this hidden information. This method has several disadvantages. Legacy films which have been released without a watermark cannot benefit from this process. There may be too many un-watermarked copies in existence which can be used as sources for copying. Also, many video alterations (or transformations) affect the watermark. If the video is copied and re-encoded in a way which changes the watermark, then it is no longer useful for copy detection purposes.

A better approach is to extract distinctive features from the content. If two videos have the same features, then one of the videos may be a copy of the other. This is known as Content-Based Copy Detection (CBCD). The underlying premise of Content-Based Copy Detection is that there is enough information within a video to create a unique fingerprint. Another way to think of it is that the content itself

MMSys'12, February 22-24, 2012, Chapel Hill, North Carolina, USA.

is a watermark.

Video copy detection is further complicated by the many edit effects which may be added either intentionally or as a by-product of the copying process. For example, the copied video may be scaled or cropped. It may have its contrast, brightness, or color balance adjusted. It can also be transcoded using different frame rates or different bit-rates. We refer to changes in videos as *transformations*. A good video copy detection system should represent the content of the video by features which are not sensitive to transformations which occur in practice. The feature set must be *robust* and *discriminating*. Robustness is the ability of the features to remain intact and recognizable in the face of encoding distortions and edit effects. A discriminable feature is one which is representative of the video content, but unlikely to be found in videos which do not have the same content. It will allow us to filter out videos that do not have matching content.

In addition to being robust and discriminating, the feature set extracted should be compact. A compact signature requires less storage space and less computation to evaluate how closely two signatures match. The evaluation process must be efficient and fast to effectively determine copies in databases which may contain thousands of hours of video.

In this paper, we propose a novel content-based video copy detection algorithm. The proposed algorithm uses highly discriminative local interest points in a way which avoids the computational complexity usually associated when matching these high dimensional descriptors. Specifically, the proposed algorithm creates combined signatures that capture the spatial and temporal features of videos. These combined signatures enable the algorithm to provide high accuracy (in terms of precision and recall). In addition, these signatures require small storage and are easy to compute and compare. This makes the proposed algorithm more computationally efficient than previous ones in literature.

We have implemented the proposed algorithm and conducted extensive experimental analysis with a large video dataset prepared for video copy detection by TRECVID [15]. We evaluated the precision and recall values achieved by the proposed algorithm under many realistic scenarios, including: (i) short clips of reference videos are copied, (ii) copied video clips are subjected to individual video transformations such as cropping, scaling, blurring, camcording, logo insertion, and contrast change; we considered a total of 16 different video transformations, and (iii) copied video clips are subjected to multiple video transformations at the same time. In addition, we analyzed the running time of the proposed algorithm and how it can be substantially decreased for large-scale video copy detection systems. Our rigorous results show that the proposed algorithm produces high precision and recall values and it is robust to many video transformations.

The rest of this paper is organized as follows. Section 2 summarizes the related works in literature. Section 3 presents the details of the proposed algorithm. Section 4 presents the setup of our experiments, summarizes our results, and compares the performance of our algorithm versus others in literature. Section 5 concludes the paper and outlines possible extensions for this work.

2. RELATED WORK

Some of the early video copy detection efforts focused on using color as the discriminatory feature. For example, Naphade et al. [12] propose a system which creates a histogram in the YUV color space. The luminance, or Y-channel, is quantized into 32 bins and each of the 2 chrominance channels, U and V, are quantized into 16 bins. Each frame in the video sequence is represented by these three histograms. The distance between frames is calculated based on a sliding window approach and the intersection of the histograms. Color-based signatures are weak. Encoding or re-encoding often cause global variations in color. Another weakness of color signatures is that similar colors can be present in very different video clips. An example of this is two video clips with different ocean scenes. The resultant histograms would have a high intersection because each histogram would be predominately blue, but they would not be copies. In general, color signatures are not robust to common color shifts and they are not discriminating enough.

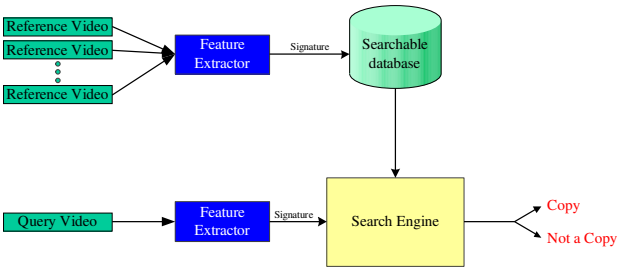


Figure 1: Common architecture for video copy detection systems.

Two of the widely accepted feature detection algorithms are SIFT (Scale Invariant Feature Transform) and SURF (Speeded Up Robust Features) [1]. Both are invariant to many common transformations, but SURF is about 6 times faster [4] in calculating interest points. The feature vector from SIFT has 128 dimensions, while that of SURF has 64 dimensions. In general, SIFT features are more discriminating, but harder to compute, require more storage, and are more complex to compare with each other. They are well suited for object recognition and image retrieval tasks. SURF features are popular because they are faster and less complex.

Local information captured by SIFT or SURF features are more descriptive, but because of the high dimensionality of each feature vector and the large number of features detected on a frame, the computational complexity of the distance matching is greatly increased.

Roth et al. [13] use the robustness of local descriptors in their copy detection algorithm, but they reduce the high dimensionality of the resultant vectors in the following way. They take every frame of the source video and divide it horizontally and vertically by four. This forms 16 regions for each frame. They then find local points of interest (SURF features) in the frame. Typical feature extraction algorithms will store for each point of interest an n-dimensional vector describing the feature. The authors of this paper instead choose to simply count the number of features identified in each region and use this as the metric for copy detection. They create a database of videos and store for each video, the pre-processed SURF count for each frame. To determine whether a query video is a copy, they perform the same re-

gion by region SURF count for each frame in the query video and compare this to each source video. The comparison metric is the normalized sum of the differences in SURF counts for each region in the frame. Two frames are considered to be a match if this normalized sum is below some threshold. A copy is reported if the longest running subsequence of matches is above another threshold. One problem with this approach is its memory requirement. It needs to compare every frame in the query against every frame of the reference and store a score for each comparison in a table for later analysis. This requires $O(NM)$ memory for a query of M frames and a reference of N frames. Similarly, the running time is $O(NM)$. Our algorithm requires $O(N+M)$ memory and its running time is always better at $O(NM - M^2)$.

3. SYSTEM OVERVIEW AND PROPOSED ALGORITHM

In this section, we first present an overview of the proposed spatio-temporal video copy detection algorithm. Then, we present the details. Finally, we analyze the time complexity of the proposed algorithm.

3.1 Overview

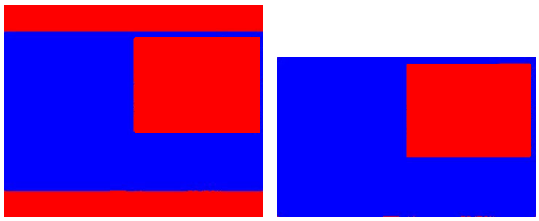
We illustrate in Figure 1 the common architecture for video copy detection systems. We define a *reference video* as the original video that we want to locate copies of. Signatures of reference videos are created and stored in a database offline. A *query video* is tested to see whether it is a copy by testing it against the database of reference videos using the proposed video copy detection algorithm.

Our algorithm improves the spatial signature of [13] by adding a temporal component using ordinal analysis. An ordinal number refers to the position or rank in a sorted list. The temporal component is calculated by ranking the spatial SURF count regions along the time line. While the actual counts may vary from frame to frame under various transformations, it is assumed that the spatial ranking of the blocks within the frames will change relatively slowly within a video shot. This enables our algorithm to subsample video sequences aggressively. That is, our algorithm compares only sample frames from the query video to reference video. This sub-sampling yields substantial saving in the storage space needed in the database for the signatures as well as significantly reducing the running time of the search algorithm since we are comparing fewer frames.

As detailed in the following section, there are several steps in the algorithm. However, the main idea of the proposed algorithm is to create combined and simple signatures that capture the spatial and temporal features of videos. Combined signatures improve the accuracy of the video copy detection process. In addition, these signatures are easy to compute and compare. This makes the proposed algorithm more computationally efficient than previous ones in the literature.

Figure 3 shows a simple example of how we create a signature for a video clip of 4 frames. The spatial part of the signature is computed by dividing each frame into regions. We detect local SURF (Speeded Up Robust Feature) features in each region. Then, we count the number of SURF features found within each region. These counts of SURF features in the regions of a frame represent the spatial part of the signature for that frame. The matrix S shows the

(a) Video with letter-box transformation and PiP



(b) Mask of static pixels (c) Mask with Borders removed

Figure 2: The mask for a combination of PIP and letter-box transformations.

spatial part of the signature in Figure 3. To add the temporal information to the signature, we sort the number of counts in each region along the time line and assign an ordinal value based on its temporal rank. In Figure 3, each row of the matrix λ shows the temporal part of the signature for one region. Notice that the element $\lambda_{i,j}$ represents the rank (or order) of region i in frame j. For example, region 1 in frame 1 has the smallest number of SURF features in all four frames, and thus has a rank of 4 in the λ matrix.

The final combined signature of a video clip (matrix λ) will have two parts: spatial information from partitioning the frames into regions and temporal information by ranking each region along the time-line. To detect copies, signatures can be compared using any similarity metric; we use the L1 distance in our implementation.

3.2 Algorithm Details

The pseudo code of the proposed algorithm for video copy detection is outlined in Algorithm 1. The algorithm runs once for each query video. It creates a signature for this query video. Then, it compares this signature against precomputed signatures of the reference videos. The algorithm has several steps, which are described in the following.

1. Remove static borders, letter-box and pillar-box effects. This preprocessing step is done by examining how the pixels change throughout the clip. Pixels with very low variance are likely edit effects added to the video. These effects include borders, logos, pattern insertion as well as letter-box and pillar-box effects from resizing.

Algorithm 1: The Proposed Algorithm

Input: Db_{ref}: Reference videos signature Database
Input: V_{query} Query video
Input: *Threshold:* Threshold distance
Input: H_r, V_r: Number of horizontal and vertical regions
Output: **True** if the query is a copy, **False** otherwise
Output: Location of best match in the reference video

foreach *Signature,* Sig_{ref} *in* Db_{ref} **do**
  $M \leftarrow$ Number of frames in the reference video
  $N \leftarrow$ Number of frames in in the query video
  $R \leftarrow \mathrm{H}_r * \mathrm{V}_r$: Number of regions in each signature
  $offset \leftarrow 0$
  $minDist \leftarrow \infty$
  foreach *Frame in* V_{query} **do**
    Crop the frame to remove static borders, letter-box and pillar-box effects
    Divide into a grid of H_r x V_r regions
    Calculate static pixels' percentage in each region
    Count the number or SURF interest points in each region and rank each region temporally
    $Sig_{query} \leftarrow \{r_0, r_1, ..., r_R\}$, where r_i is the ranking vector of the M frames of region i
  end
  for $i \leftarrow 1$ **to** $N - M$ **do**
    foreach *Usable region in the grid* **do**
      $regionDist \leftarrow \frac{1}{M} \sum_{k=1}^{M} |rank(Sig_{ref}(k + offset) - Sig_{query}(k)|$
    end
    $dist \leftarrow \frac{1}{R} \sum_{k=1}^{R} regionDist(k)$
    $offset \leftarrow offset + 1$
    if $dist < minDist$ **then**
      $minDist \leftarrow dist$
      $minOffset \leftarrow offset$
    end
  end
  if $minDist \leq Threshold$ **then**
    Output: True
    Output: $minOffset$
  end
  else Output: False
end

The variance is calculated using the formula of Mark Hoemmen [16] on each pixel. The gray-scale value of pixel x in frame i is x_i.

$$M_k = \begin{cases} x_1, & k = 1 \\ M_{k-1} + \frac{x_k - M_k - 1}{k}, & k = 2, ..., n \end{cases}$$

$$Q_k = \begin{cases} 0, & k = 1 \\ Q_{k-1} + \frac{(k-1)(x_k - M_k - 1)^2}{k}, & k = 2, ..., n \end{cases} \quad (1)$$

Once we get to the n^{th} frame and have calculated Q_n, the variance is simply Q_n/n. Figure 2(a) is an example of a video with both a letter-box and a picture-in-picture transformation. Its mask, shown in Figure 2(b), is used to remove borders on the outside of the video. The red regions are pixels whose variance is below threshold. If all pixels in a row (or column) on the outside border have a variance below threshold, they are removed from the image. The process is repeated until a row (or column) is encountered where at least one pixel shows variance above the threshold. The result is an image which is cropped of borders in which the pixels do not vary. The sub-image corresponding to the size of the cropped mask in Figure 2(c) is used for further processing. This will remove any pillar-box effects, letter-box effects, and borders from cropping or shifting.

2. Divide the video into regions. This is a configurable parameter. We can set the number of regions by specifying the number of vertical and horizontal partitions. For example, specifying 3 horizontal and 2 vertical partitions would divide the frame into a grid with 3 rows and 2 columns for a total of 6 regions.

3. Calculate the percentage of static pixels in each region. Each region is examined for static pixels. The presence of static pixels within the cropped image can indicate the presence of an image, some text, a logo, or a background pattern superimposed onto the video. If a significant number of pixels within a region are masked then too much of the area may be occluded to get useful information. If this is the case, the region can be turned off. The distance will be calculated based on the remaining regions. In Figure 2(c), we can detect the presence of a picture in picture transformation shown in red. If the percentage of red pixels in a region is too high, then that region is not used in the distance calculation.

4. Count the number of SURF interest points in each region. The SURF features for the frame are extracted. Each interest point is described using a 64-dimensional vector, but we are only interested in the location of the interest point in the frame. We determine which region the interest point belongs to and increment the count for that region.

5. Create the temporal signature. The temporal aspect of the signature is obtained by sorting the SURF feature counts in each region along the time-line. The frame with the most SURF interest points is assigned an ordinal value of 1. The frame with the next highest number of interest points is assigned a value of 2, and so on. Figure 3 provides an example of the signature creation process. The end result is a matrix where each row contains the ranking vector of a particular region.

   More formally, for a video consisting of M frames and L regions, each region t_i would result in an M-dimension vector, $s_i = (f_{i,1}, f_{i,2}, ..., f_{i,M})$, where $f_{i,k}$ is the number of SURF features counted in region i of frame k. The matrix $S_i = (s_1, s_2, ..., s_L)$ is used to produce the ranking matrix, λ=($\lambda_1, \lambda_2, ..., \lambda_L$). Each λ_i = $(r_1^i, r_2^i, ..., r_L^i)$, where r_k^i is the rank of the i^{th} region of frame k.

   For a video with M frames and L regions, the signature for the video consists of an LxM matrix.

6. Calculate the distance between two signatures. The distance between a reference video and a query video

is based on the L1 distance between them. Our general approach will be this. The number of frames in the reference video is N and the number of frames in the query video is M, where $N \geq M$. We divide each video into L regions.

We adopt a sliding window approach. First, we calculate the distance between the query video and the first M frames of the reference video. We then slide our window of M frames over one frame and find the distance between the query video and M frames in the reference video starting at the second frame. We keep track of the minimum distance and the frame offset, p, for which this occurred as we continue sliding our window. Once we reach the end of the reference video, the best match occurs at the minimum distance.

If λ_i is the ranking vector of the i^{th} region, the distance between a query video V_q and a reference video V_r is calculated as:

$$D(V_q, V_r) = \underset{p}{\text{argmin}}(\ D(V_q, V_r^p)\ ), \tag{2}$$

where p is the frame offset in the reference video which achieved this minimum and represents the location of the best match between the query video and the reference video, and $D(V_q, V_r^p)$ is given by:

$$D(V_q, V_r^p) = \frac{1}{L}\sum_{i=1}^{L} d^p(\lambda_q^i, \lambda_r^i),\ where$$

$$d^p(\lambda_q^k, \lambda_r^i) = \frac{1}{C(M)}\sum_{j=1}^{M} |\lambda_q^k(j) - \lambda_r^j(k)(p+j)| \tag{3}$$

$C(M)$ is a normalizing factor which is a function of the size of the query. It represents the maximum possible distance between the reference video and the query video. This maximum distance occurs when the ranking of the reference video is exactly opposite to that of the query. There are two cases based on whether M is even or odd. The case when M is even is illustrated in Figure 4. It is twiece the sum of the first $M/2$ odd integers. Similarly, when M is odd, $C(M)$ is twice the sum of the first $(M-1)/2$ even integers. Each of these sequences can be computed directly as follows:

$$C(M) = \begin{cases} \left(\frac{M}{2}\right)^2, & M\ even \\ \left(\lfloor\frac{M}{2}\rfloor\right)\left(\lfloor\frac{M}{2}\rfloor + 1\right), & M\ odd \end{cases} \tag{4}$$

7. Decide if the query video is a copy of the reference video. If the minimum distance between the query video and the reference video at offset p is below a threshold, then it is likely that the query video is a copy of the reference video. In this case, we report that a copy has been located starting in frame p of the reference video.

3.3 Algorithm Analysis

We analyze the time complexity of the proposed algorithm in this subsection. We compute the time needed to process a query video of M frames when it is compared against a reference video of N frames. The algorithm starts by finding the distance between the set of frames from the query video and the first M frames in the reference video. It then shifts the comparison window and finds the distance between the query set and the reference set starting at the second frame of the reference set. The algorithm continues for a total of $N - M + 1$ calculations. For every calculation, it compares the M frames in the query with M frames of the reference video for a total of $(N - M + 1)M$ comparisons. The total running time is thus $O(NM - M^2 + M)$. As the ratio of the number of frames in the query video to the number of frames in the reference video M/N increases, the proposed algorithm will run faster, achieving its best running time as M/N approaches 1. In this case it becomes $O(M)$ which means the proposed algorithm would be ideal for finding full length videos. Small queries also run very fast. Taking the derivative of the running time and setting it equal to zero, we see that that the worst running time is achieved when $M \approx \frac{N}{2}$. Around this region our algorithm has a running time of $O(M^2)$ and will greatly benefit from search space reduction techniques.

Thus, our algorithm has a worst-case running time similar to running times of other algorithms [2] [5] [7] [12] which use a sliding window approach. However, for full-length videos and short clips, the running time of our algorithm is much smaller than previous ones as it needs only $O(M)$ steps.

In addition, our algorithm has a better running time than algorithms that compare frames pairwise such as [14] [13] [17] [6]. These algorithms find the distance of all possible pairs between the query set and the reference set. Since each

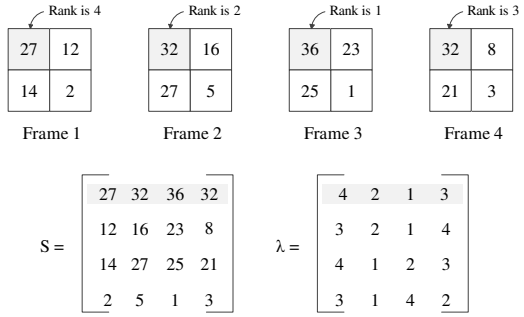


Figure 3: Building the ranking matrix. Here, each frame is divided into a 2x2 grid. The number of SURF features are counted for each area of the grid to produce the matrix S. The ranking matrix, λ, is built as follows: each row stores the ordinal rank of the corresponding frame over the length of the video sequence.

1	2	3	4	5	6	7	8	9	10
10	9	8	7	6	5	4	3	2	1
9	7	5	3	1	1	3	5	7	9

Figure 4: The top two rows show the ranking vectors between a query and a reference video where the vectors are exactly opposite to each other. Here, M=10 is even and the normalization factor, $C(M)$, is simply twice the sum of the first $M/2$ odd integers.

frame of the query set must be compared to each frame in the reference set, the running time is always $O(MN)$. Note that $N \geq M$. The proposed method will always be faster than these algorithms, since we are always subtracting M^2 from the running time of $O(NM)$. This reduction in running time is substantial especially as the ratio M/N approaches 1.

Furthermore, as shown in the evaluation section, the proposed algorithm is quite robust to sub-sampling, i.e., the accuracy of the returned results by our algorithm does not decrease substantially when the algorithm compares only a subset of the frames in the query and reference videos instead of comparing all frames. This enables our algorithm to substantially decrease its running time, especially when used in large-scale video copy detection systems.

4. EXPERIMENTAL EVALUATION

We have implemented the proposed spatio-temporal video copy detection algorithm, and we rigorously evaluated it using many real videos prepared for video copy detection by TRECVID [15]. We start by describing our experimental setup in the following section. Then, we present the results of our different experiments. We then compare our algorithm versus several recent algorithms in literature.

4.1 Experimental Setup

Video Dataset and Transformations. The videos used in the evaluation come from the TRECVID 2009 dataset [15]. These videos were provided by the Netherlands Institute for Sound and Vision. The videos vary in size from 50 seconds to over 1.5 hours. They are stored in MPEG-1 format at 352 x 288 pixels. Altogether, there are 399 videos totaling 107 GB of data. This amounts to over 180 hours of content and over 13 million frames. The videos are encoded at a frame rate of 25 frames per second. The dataset contains a wide variety of video sequences including newscasts, documentaries, interviews, educational programming, and sporting events.

We call the 399 videos *reference* videos. They represent the original videos from which we want to detect copies. We create *query* videos as follows. We randomly choose seven videos from the reference videos. Since users usually copy (and post online) clips of different sizes of original videos, we choose segments of different sizes and starting at random locations. Segment sizes range from few frames to full videos, depending on the experiment being conducted. When copied, video clips can be changed either intentionally or as a side effect of the recording and copying process. Intentional modifications on video clips can be used to avoid detection, insert ads into videos, reduce bit rates, and/or change a video's resolutions. On the other hand, recording and copying equipment can introduce noise and blurring, record videos in different resolutions, change video contrast, and/or change the gamma shift parameter. We refer to these video changes as *video transformations*. Therefore, query videos are created from clips of different sizes, which can be subjected to one or more transformations.

To perform thorough evaluation of our algorithm, we consider query videos that are subjected to many video transformations. We consider videos that are subjected to individual transformations as well as those subjected to multiple transformations at the same time. In particular, we applied the following individual transformations:

Transformations		
2 pixel blur	contrast +20%	letter-box
5% crop	10% noise	gamma shift 0.8
10% noise	text insertion	3 pixel Blur
scale 120%	horizontal flip	rotate 5^o
camcording $15^o - 15^o$	pillar-box	contrast -20%
shift (20,10)	rotate 10^o	horizontal flip
camcording $20^o - 20^o$	gamma shift 1.3	contrast -30%
scale 75%	5% crop	shift (50,30)
text insertion	stretch	letter-box
20% noise	horizontal flip	5% crop

Table 1: Combined transformations applied to create query videos.

- Gaussian blur of 3 pixels in radius
- Gamma shift to 0.5
- Gamma shift to 1.6
- Image rotation by 10^o
- Image shift by 50 horizontally and 40 pixels vertically
- Image crop of 20 pixels from all sides
- Addition of random noise to 20% of the pixels
- Addition of text
- Image height scaled by 75% creating both a horizontal stretch and a letter-box effect
- Camera angle adjusted by 20^o horizontally and 20^o vertically
- Image resize to 50% of its original size
- Image resize to 200% of its original size
- Contrast decreased 30%
- Contrast increased 30%
- Zoomed to 125%
- Flipped along vertical axis centered horizontally on the video

Including the untransformed clips, the above transformations resulted in 119 query videos, which we used in the experiments. The magnitude of each transformation is chosen based on the maximum used in TRECVID's evaluation criterion for that transformation.

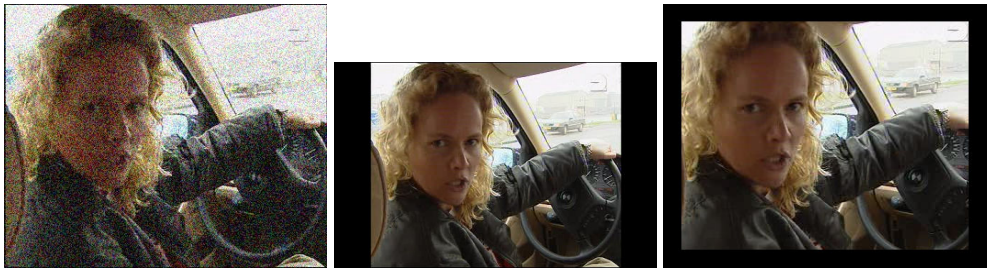
For multiple transformations, we combined 3 different transformations in various configurations. We created 10 different combinations, which are summarized in Table 1 along with their parameters, and applied them on each of the randomly chosen 7 videos. This adds a total of 70 transformed videos to the query videos.

We used the Adobe Premier Pro tool to choose video segments and to apply various transformations on them. To visually illustrate the meaning of the various video transformations and help the reader understand their impact on videos, we present an example in Figure 5, which we created using the Adobe Premier Pro tool.

(a) Original Image (b) Picture in picture (c) Camcording

(d) Text/logo insertion (e) Gamma decreased (f) Gamma increased

(g) Blur (h) Rotate (i) Contrast decreased



(j) Addition of noise (k) Pillar box (l) Crop

(m) Shift (n) Flip (o) Letter-box with stretch

Figure 5: Illustration of common transformations that can be applied on copied videos.

Performance Metrics. We use two main metrics to assess the performance of the proposed algorithm: precision and recall, which are defined in the following two equations.

$$precision = \frac{number\ of\ correctly\ identified\ copies}{total\ number\ of\ reported\ copies}. \quad (5)$$

$$recall = \frac{number\ of\ correctly\ identified\ copies}{actual\ number\ of\ copies}. \quad (6)$$

In addition, we measure the running time of the algorithm, which is an important metric because in practice video copy detection algorithms are expected to process thousands of videos in a short period (typically, video copy detection systems crawl online video websites every few hours to download and check newly posted videos for copies). **Experiments Conducted.** We conduct the following experiments, which are detailed in later sections:

- Base case : evaluate the proposed algorithm with query videos that have no transformations.
- Robustness against individual transformations: evaluate the proposed algorithm with query videos that were subjected to individual transformations.
- Robustness against multiple transformations: evaluate the proposed algorithm with query videos that were subjected to combined transformations.
- Computational complexity and sub-sampling: measure the running time of the proposed algorithm and the tradeoff between sub-sampling, running time, precision and recall.
- Comparison: compare against recent algorithms in the literature.

4.2 Base Case Evaluation

The purpose of this experiment is to test the effectiveness of the proposed algorithm on copied video clips that do not have any transformations. This scenario happens, for example, when a clip is directly taken from a digital video stored a on DVD or hard disk. We create 7 query videos, each has 750 frames (30 sec), which is the minimum query length used by TRECVID. Each query video is compared against all 399 reference videos. We vary the distance threshold between 0.0 and 1.0 and compute the precision and recall for each case. We plot the results in Figure 6. The figure shows that the proposed algorithm can achieve 100% precision and 100% recall using a threshold value in the range of 0.23 to 0.28.

4.3 Robustness against Individual Transformations

In this experiment, we apply individual transformations on query videos. This is to test the robustness of the proposed algorithm against video changes that usually occur in practice when videos are copied. We apply all transformations mentioned in Section 4.1, and illustrated in Figure 5, one by one on all query videos. That is, we repeat the experiment 16 times, and in each repetition, we vary the distance threshold between 0.0 and 1.0 and compute the precision and recall for each case. Then, we compute the average

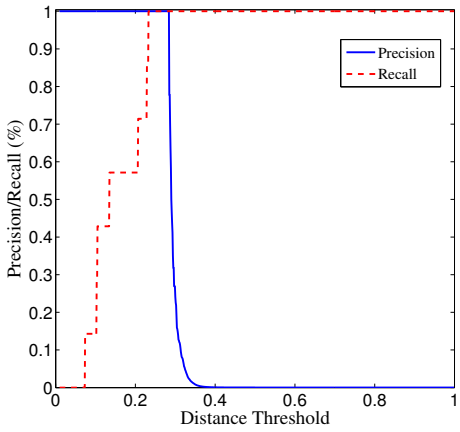


Figure 6: Performance of the proposed algorithm on videos with no transformations.

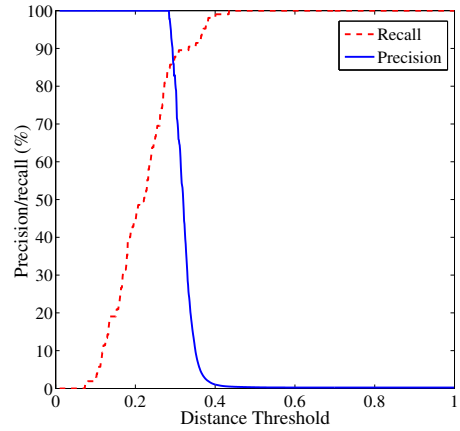
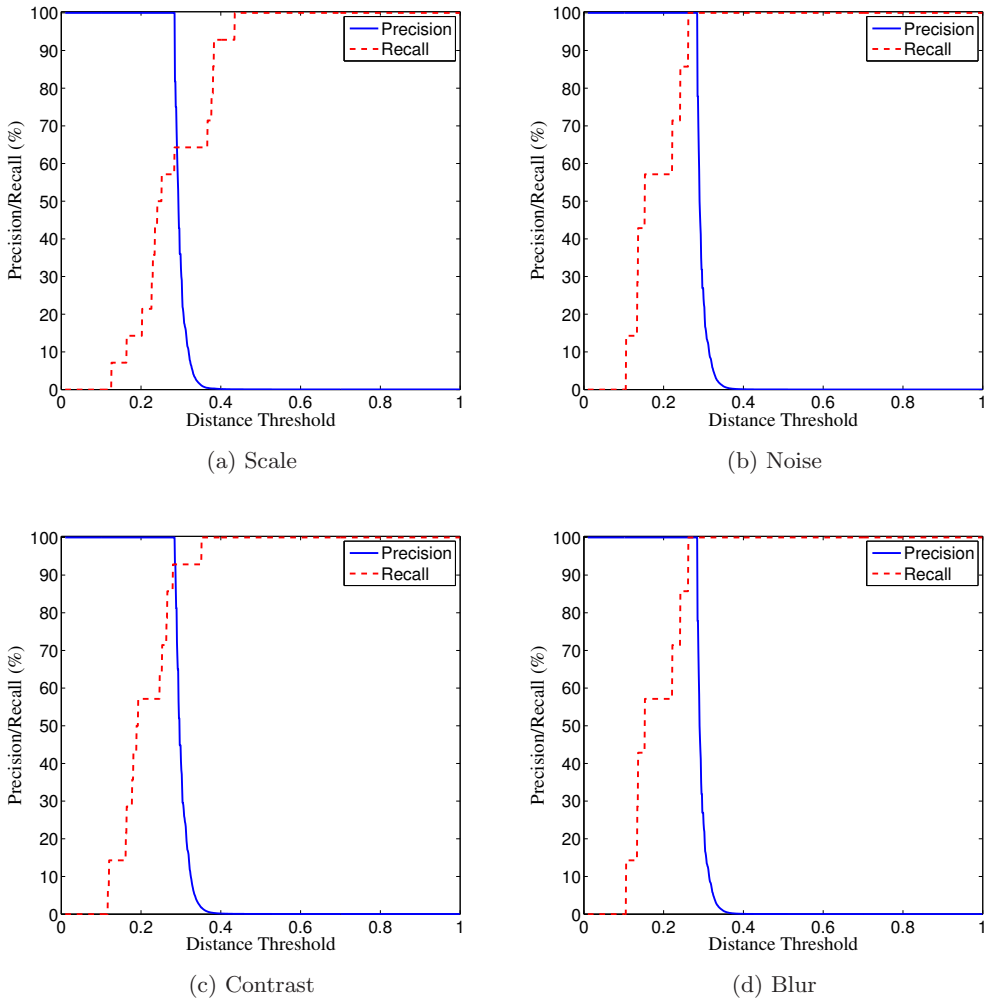


Figure 7: Performance of the proposed algorithm on videos with individual transformations: Average results across all transformations.

values across all experiments for the corresponding values of threshold. We plot the average precision and recall values in Figure 7. The figure shows that at threshold of 0.28, the proposed algorithm can still achieve 100% precision with 77% recall. The intersection point of the precision and recall curves occurs at a threshold of 0.30 where both recall and precision have a relatively high value of 85%, given that copied clips of the videos have been significantly changed from the original ones. Lowering the threshold gives greater precision, but more copies may go undetected. Raising the threshold catches more copies, but false positives could be reported.

In Figure 8, we plot four samples of the precision-recall curves that we obtained for individual transformations. This figure shows that some transformations affect the precision and recall values more than others. For example, the blur and contrast transformations have less impact on the precision and recall curves than the noise and scale transformations.



(a) Scale (b) Noise

(c) Contrast (d) Blur

Figure 8: Performance of the proposed algorithm on videos with individual transformations: sample results for individual transformations.

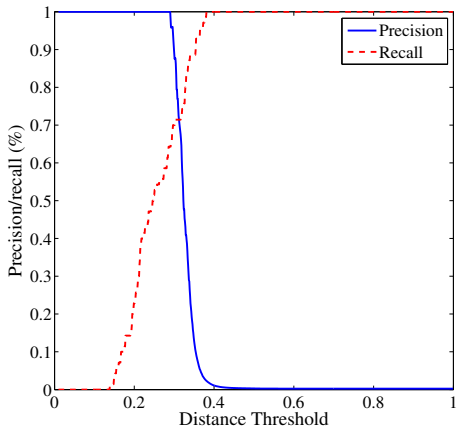


Figure 9: Performance of the proposed algorithm on videos with multiple transformations.

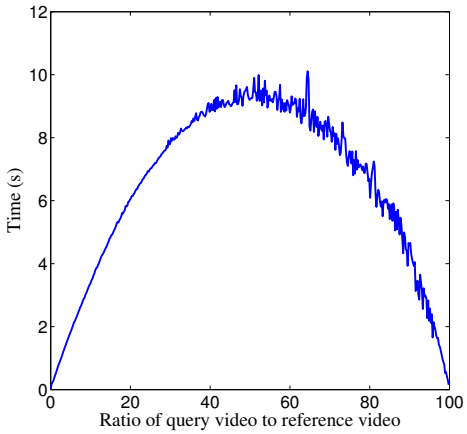


Figure 10: Running time as a function of the ratio of query to reference video length.

4.4 Robustness against Multiple Transformations

In this experiment, we evaluate how well our system is able to detect copied video clips altered by applying multiple transformations on them at the same time. We apply the 10 combined transformations mentioned in Table 1 on all query videos. We repeat the experiment 10 times, once for each combined transformation. In each experiment, we vary the distance threshold between 0.0 and 1.0 and compute the precision and recall for each case.

Then, we compute the average values across all experiments for the corresponding values of threshold. We plot the average precision and recall values in Figure 9. As the figure shows, the best recall for 100% precision is 64%, which is achieved using a threshold of 0.29. The intersection point of the precision and recall curves occurs at a threshold of 0.31. At this point, the precision and recall values are 71%. We note see that the recall rate for the same precision is lower than in the previous case where individual transformations are applied. This is expected, since the more the video is transformed, the less it resembles the original, and the greater the distance between them.

In summary, the results in this subsection and the previous two show that the proposed video copy detection algorithm achieves high precision and recall values and is robust to common transformations performed on copied videos.

4.5 Time Complexity and Sub-sampling

In this subsection, we measure the running time of the proposed algorithm and study the effect of the query length on the running time. We also show that the running time of our algorithm can further be reduced by applying it only on a subset of the video frames without substantially decreasing the precision and recall values achieved by the algorithm.

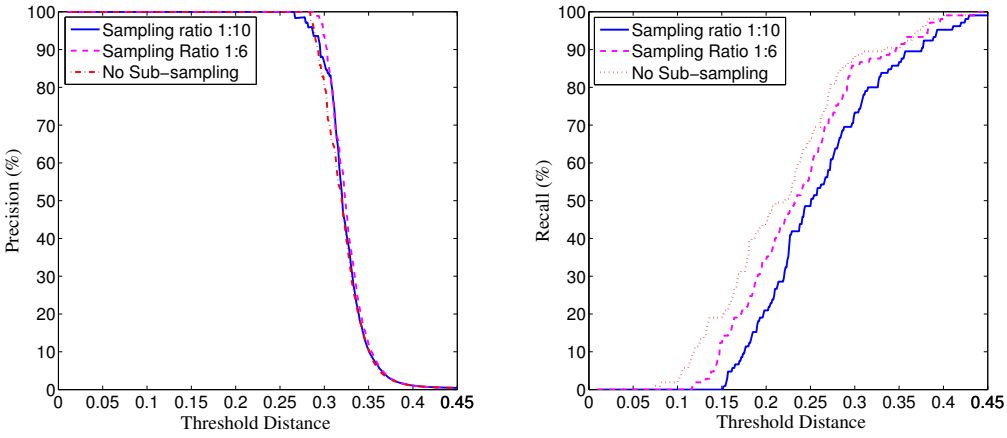
In the first experiment, we focus on one long reference video which has a length of 15 minutes. We create query videos with increasing sizes, starting from a length of 50 frames (2 sec) with increments of 50 frames. We compare the query videos against the reference video, one by one, and measure the total processing time for each query. We do not sub-sample frames in this experiment. We plot the results in Figure 10, where the x-axis shows the ratio of the length of the query video to the length of the reference video, and the y-axis shows the time in seconds. The figure first shows that the proposed algorithm is fairly efficient as it terminates in a few seconds in the worst case, which happens when the query length is about 50% of the length of the reference video (which is 15 minutes). That is, examining a 7.5-minute query video (11,250 frames) against a 15-minute (22,500 frames) took less than 10 seconds on a commodity PC with unoptimized implementation of our algorithm and without sub-sampling frames.

The second point shown by the results in Figure 10 is that the running of the proposed algorithm is minimal when the ratio of the query video length to the reference video length is either small or close to one. The measurements obtained during this experiment confirm the theoretical analysis of the time complexity presented in Section 3.3.

Although the running time of the proposed algorithm is small, we aim to further shorten it in order to support large-scale video copy detection systems with reasonable computing hardware resources. We accomplish this by reducing the search space when comparing a query video versus a reference video by comparing only a subset of the frames. We refer to this as sub-sampling of the video frames. This will speed up the search phase of our algorithm. In order for this to be effective, the algorithm must deliver close precision and recall values using sub-sampling as in the case without sub-sampling.

We repeat the experiments presented in Sections 4.2 and 4.3, but we use different sampling ratios. A sampling ratio of 1:X means that we consider every Xth frame for comparison, instead of considering each single frame. We conduct three experiments with sampling ratios of 1:10, 1:6, and 1:1 (no sampling). In each one of them, we vary the distance threshold from 0.0 to 1.0 and we measure the total running time for processing all queries and the precision and recall values achieved.

The total running time for processing all 119 queries are about 16 hours, 46 minutes, and 18 minutes for sampling ratios of 1:1, 1:6, and 1:10, respectively. These values clearly show the substantial saving in running times that can be



(a) Precision (b) Recall

Figure 11: Precision and recall graphs for different sampling ratios.

achieved by sub-sampling. For example, a speed up factor of more than 20 (=16*60/46) can be achieved with a modest sampling ratio of 1:6. We note that the running time is not affected by the distance threshold value used.

Next, we show the impact of sub-sampling on the accuracy of the results produced by the algorithm. We plot the precision and recall values in Figure 11 for the considered three sampling ratios and for a range of distance threshold values. The results in this figure show that there is a very small impact on precision and recall, especially for the sampling ratio 1:6. That is, the substantial saving in running time (factor of 20+) comes at almost negligible reduction in the accuracy of the results.

In summary, the experiments in this subsection show that the proposed algorithm is computationally efficient and it offers administrators of video copy detection systems a control knob (the sampling ratio) to further reduce the running time at a small loss in accuracy.

4.6 Comparison against other Algorithms

In this subsection, we compare our proposed algorithm against others in the literature. We first note that this comparison is a bit tricky, since previous algorithms usually use different datasets and sometimes even different evaluation criteria. In addition, there are typically several parameters in each algorithm that need to be fine tuned. We take a conservative approach in conducting this comparison: we contrast our results to the best results reported in the previous papers by their authors. This is conservative as authors typically optimize their algorithms to yield the best performance.

We start with the system proposed by Chen et al. [3]. This system was evaluated using only few simple transformations, namely: Change in contrast $\pm$ 25%, resize to 80% and 120%, Gaussian blur of radius 2, and letter-box and pillar-box. The authors compared their system against the systems in [7], [8], and [9]. They showed that their system performs the best among all compared systems. In contrast, our work handles many and more complex transformations and achieves better results than the system proposed by Chen et al. [3]. For example, at 90% precision, our algorithm achieved an overall recall rate of 82% for the 16 different types of transformations evaluated. The system of Chen et al. [3] had a recall rate under 80% for the 90% precision, and that is achieved only with 4 simple transformations. Since our system outperforms the one in [3], we can conclude that it also outperforms the ones in [7], [8], and [9].

The algorithm proposed by Wu et al. [17] achieves 100% recall and precision. Since this algorithm is designed for video retrieval, queries prepared by the authors did not have any transformations applied to them. Li et al. [11] achieve 100% recall with 100% precision by evaluating only simple transformations applied singly. They evaluated blurring, cropping, flipping and resizing. They set the levels of each transformation low which made them easier to detect. Our algorithm is also able to obtain both 100% recall and precision on untransformed videos and high precision and recall values on videos with many individual and combined transformations, which is more realistic.

The system of Zhang et al. [18] is tested against some of the transformations evaluated in our system, but they do not consider zooming, camcording, shifting, cropping, or flipping. Their system tests small increases and decreases in playback speed and rotations of 90^o, 180^o, 270^o. Including 3 simple transformations of this type artificially boosts the results of the recall and precision metric. For most of the transformations, they do not indicate the level of transformation. This makes it difficult for an accurate comparison. In their testing, they attained 86% precision with 75% recall. These results are comparable with our system if the levels of transformations are similar. In addition, our algorithm is robust against multiple transformations, which they did not consider. Law-To et al. [10] achieved 95% precision with 90% recall using a query length of 60 seconds. That is twice as long as ours which was just 30 seconds. Doubling our query length would further improve our results. Our algorithm is an improvement of the system of Roth et al. [13]. However, our system has lower computational complexity. They tested their system more rigorously on queries which had combinations of several transformations to obtain 50% recall for the same precision of 86%. We expect that with

the same data our system would be much faster, particularly for full length queries.

Douze et al. [6] were unable to handle flipping. They were also unable to detect copies that were scaled to 30% of the original size. They got around these limitations by adding the signature of a flipped version and one resized by 50% to the database. Using SIFT features, they achieved close to perfect precision and recall for similar transformations as our system. However, the computational complexity of dealing with the high dimensional descriptors resulted in a running time too high to make this a useful system. Finally, we suspect that the algorithm of Chen and Stenfiford [3] may yield slightly better precision and recall values than ours, but at much higher computational cost. However, we could not confirm this because their video database and queries are not available to us.

5. CONCLUSIONS AND FUTURE WORK

We have presented a new algorithm for detecting video copies. The algorithm creates a signature to describe the content of a video. This signature has the following properties: (i) it has a spatial component from dividing the frame into regions, (ii) it has a temporal component by considering the ordinal ranking along the time line, (iii) it is robust to many common transformations, even at extreme limits, and (iv) it is compact, requiring just 16 bytes per frame. The proposed algorithm is computationally efficient. For example, our experimental results show that it can process a database of 13 million frames in under 20 seconds on a commodity workstation. We conducted an extensive experimental study to show that the proposed algorithm produces high accuracy. Specifically, the algorithm achieves 100% precision and 100% recall when the query videos are unmodified clips from the reference videos. Our results also show that the proposed algorithm is robust to many video transformations even when multiples transformations are applied at the same time.

The work in this paper can be extended in different directions. For example, the current implementation assumes that the query video is a copy of only 1 reference video. Some copied videos may contain content from many different sources. This is not a difficult obstacle to overcome. We can incorporate shot detection into this work to divide a complex query into a number of sub-queries based on shot boundaries. These sub-queries can be run independently on the current system.

Acknowledgments

This work is partially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada and the British Columbia Innovation Council (BCIC).

6. REFERENCES

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[2] D. Bhat and S. Nayar. Ordinal measures for image correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):415 –423, April 1998.

[3] L. Chen and F. W. M. Stentiford. Video sequence matching based on temporal ordinal measurement. *Pattern Recognition Letters*, 29:1824–1831, 2008.

[4] S. Chen, J. Wang, Y. Ouyang, B. Wang, Q. Tian, and H. Lu. Multi-level trajectory modeling for video copy detection. In *Proc. of the IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP'10)*, pages 2378 –2381, March 2010.

[5] C. Chiu and H. Wang. A novel video matching framework for copy detection. In *Proc. of the 21th IPPR Conference on Computer Vision, Graphics and Image Processing (CVGIP'2008)*, August 2008.

[6] M. Douze, H. Jegou, and C. Schmid. An image-based approach to video copy detection with spatio-temporal post-filtering. *IEEE Transactions on Multimedia*, 12(4):257 –266, June 2010.

[7] A. Hampapur, K. Hyun, and R. M. Bolle. Comparison of sequence matching techniques for video copy detection. In *Storage and Retrieval for Media Databases*, pages 194–201, 2002.

[8] X. Hua, X. Chen, and H. Zhang. Robust video signature based on ordinal measure. In *Proc. of the International Conference on Image Processing (ICIP'04)*, volume 1, pages 685 – 688 Vol. 1, October 2004.

[9] C. Kim and B. Vasudev. Spatiotemporal sequence matching for efficient video copy detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(1):127 – 132, January 2005.

[10] J. Law-To, O. Buisson, V. Gouet-Brunet, and N. Boujemaa. Robust voting algorithm based on labels of behavior for video copy detection. In *Proc. of the 14th annual ACM international conference on Multimedia (MULTIMEDIA'06)*.

[11] Z. Li and J. Chen. Efficient compressed domain video copy detection. pages 1 –4, August 2010.

[12] M. R. Naphade, M. M. Yeung, and B. Yeo. Novel scheme for fast and efficient video sequence matching using compact signatures. 3972(1):564–572, 1999.

[13] G. Roth, R. Laganière, P. Lambert, I. Lakhmiri, and T. Janati. A simple but effective approach to video copy detection. In *Proc. of the 2010 Canadian Conference on Computer and Robot Vision (CRV'10)*, pages 63–70, Washington, DC, 2010. IEEE Computer Society.

[14] K. Tasdemir and A. E. Cetin. Motion vector based features for content based video copy detection. *International Conference on Pattern Recognition*, 0:3134–3137, 2010.

[15] Trecvid website. `http://trecvid.nist.gov/`.

[16] Hoemmen's one pass variance algorithm. `http://www.eecs.berkeley.edu/~mhoemmen/cs194/Tutorials/variance.pdf`.

[17] Z. Wu, Q. Huang, and S.Jiang. Robust copy detection by mining temporal self-similarities. In *Proc. of the 2009 IEEE international conference on Multimedia and Expo (ICME'09)*, pages 554–557, Piscataway, NJ, 2009. IEEE Press.

[18] Z. Zhang and J. Zou. Compressed video copy detection based on edge analysis. pages 2497 –2501, June 2010.