Content-Based Video Copy Detection: PRISMA at TRECVID 2010

Juan Manuel Barrios and Benjamin Bustos
PRISMA Research Group, Department of Computer Science, University of Chile
Blanco Encalada 2120, Santiago, Chile
{jbarrios,bebustos}@dcc.uchile.cl

Abstract

We present PRISMA's Video Copy Detection system (P-VCD). The system is based on visual-only global descriptors, weighted combinations of distances, a pivot-based index structure, and a novel approximated search and voting algorithm for copy localization.

We submitted four Runs to TRECVID 2010 CCD task:
`PRISMA.m.balanced.ehdNgryhst`: a combination of edge histogram and gray histogram.
`PRISMA.m.balanced.ehdNclrhst`: a combination of edge histogram and color histogram.
`PRISMA.m.nofa.ehdNgryhst`: a combination of edge histogram and gray histogram.
`PRISMA.m.nofa.ehdNghT10`: a combination of edge histogram and gray histogram with a different threshold.

P-VCD's results show that the combination of edge histogram and gray histogram is slightly better than edge histogram and color histogram. These results were positioned above the median, and considering just video-only Runs, were the bests positioned for Balanced and Nofa profile. These results also show that our pivot-based index enables to discard 99.9% of distance evaluations and still have good effectiveness, and that global descriptors can achieve competitive results with TRECVID transformations.

1 Introduction

Content-Based Video Copy Detection (CBVCD) consists in detecting and retrieving videos that are copies of known original videos. The detection method only makes use of visual and audio content, avoiding to embed watermarks into original videos. Joly et al. [8] propose a definition of copy based on a subjective notion of *tolerated transformations*. A tolerated transformation is a function that creates a new version of a document where the original document "remains recognizable". Let T be a set of tolerated transformations, and u and v be two documents, v will be a copy of u if $\exists t \in T,\ t(u) = v$.

TRECVID 2010 CCD evaluation considered 56 audio+visual transformations. Reference video collection was composed of 11,524 video files with a total extension of 425 hours. Query video collection was composed of 11,256 audio+visual queries (201 base queries $\times$ 8 visual transformations $\times$ 7 audio transformations) with a total extension of 224 hours.

In this paper we present P-VCD, a CBVCD system based on visual-only global descriptors, weighted combinations of distances, approximated searches with a pivot-based index structure, and a voting algorithm for copy localization. The rest of the paper is structured as follows: Section 2 reviews the current work for CBVCD, Section 3 shows our system in details, Section 4 reviews our system's results in TRECVID 2010, and finally Section 5 gives some conclusions and future work.

2 Related Work

CBVCD systems relies on two different tasks. On the one hand, *Content description* task consists in calculating representative descriptors for a video sequence. Descriptors can be either *global* or *local*. Global descriptors represent the content of a whole frame, as in [6] and [9]. Local descriptors represent the neighborhood of interest points in a frame, the most used for CBVCD are SIFT [13] and SURF [1].

On the other hand, *Similarity search* task corresponds to the algorithm for finding objects in an indexed collection that match the query. Three different approaches are used for CBVCD: continuous, discrete, and probabilistic. First, the continuous approach corresponds to the traditional search method in metric spaces. Given a query video and a distance function, it performs a range or a nearest neighbor search, selecting the closest objects to the query. Kim et al. [9] use a global descriptor and perform a range search linearly without any indexing structure. Gupta et al. [5] showed good results for audio-based CBVCD in TRECVID 2009 performing a linear nearest neighbor search implemented on GPU. Second, the discrete approach represents each descriptor with a value taken from a fixed

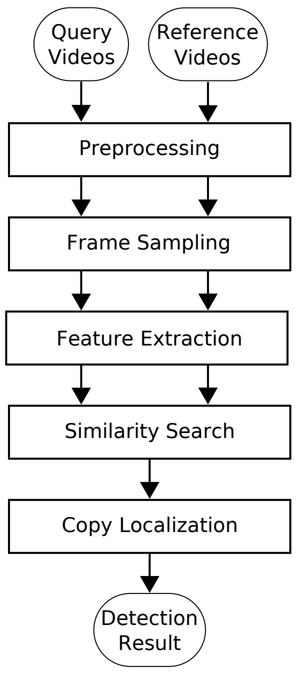

Figure 1. Tasks in P-VCD System.

set. A system implementing this approach showed good effectiveness and efficiency in TRECVID 2008 [4]. The Glocal descriptor [12] divides the descriptor space in blocks and quantizes local descriptors into those blocks creating a frame global descriptor from its local descriptors, then uses an inverted index for searching similar frames. Third, the probabilistic approach performs a probability-based approximated search. Joly et al. [8] partition the space with a Hilbert space filling curve and estimate a distribution of descriptors for each block. Poullot et al. [16] replace the Hilbert's curves with Z-order curves maintaining the probability-based search method. Law-To et al. [10] define a voting algorithm based on a geometric model and track interest points trajectories detecting copies even for complex transformations (like replacement of background or insertion of characters in a scene).

3 P-VCD System

P-VCD system uses only global visual information for detecting copies, thus only processes the 1,608 visual-only query videos (201 $\times$ 8). It is divided in five tasks: Preprocessing, Frame Sampling, Feature Extraction, Similarity Search, and Copy Localization (see Figure 1).

3.1 Preprocessing

This task tries to minimize the effect of visual transformations by filtering frames and applying inverse transformations. Each video enters to a transformation chain, producing one or more derived videos that are used by the next tasks.

Figure 2 resumes the transformation chain for query videos and reference videos. Five transformation were implemented:

- **Skip Irrelevant Frames**. Processes the whole video and marks a frame to be skipped if it is Plain or Outlier. A frame is plain when the variance of intensity pixels or the difference between minimum and maximum intensity pixels is smaller than a threshold. A frame f_i is an outlier if the previous frame f_{i-1} and next frame f_{i+1} are similar between them and frame f_i is very different to f_{i-1} and f_{i+1}. We used a function $d(x, y)$ that scales both frames x and y to 20×15 and sums differences for every intensity pixel, then f_i is an outlier when both $d(f_{i-1}, f_i)$ and $d(f_i, f_{i+1})$ are greater than a threshold and $d(f_{i-1}, f_{i+1})$ is smaller than a threshold. A skipped frame is replaced with previous frame in video, or with next non-plain frame if it is the first frame in video.
- **Remove Black Borders**. Calculates the median and variance of intensity for each pixel of every frame in a video, then removes rows and columns from borders which medians and variances are smaller than a threshold.
- **Flip**. Applies a vertical mirroring to every frame.
- **PIP Detection**. First, it applies a Laplacian kernel to every frame and it calculates the median for every pixel of processed frames, obtaining a *mean-edge frame*. Second, it detects corner candidates by applying different masks to the mean-edge frame (top-left, top-right, bottom-left, and bottom-right corner masks). Third, it searches for rectangles by joining one of each four corner types, and it calculates a score. A picture-in-picture (PIP) is detected when the rectangle with best score is greater than a threshold. Finally, PIP is reverted by creating two new query videos: foreground video (each frame cropped to the detected rectangle) and background video (detected rectangle filled with black pixels).
- **Camcording Detection**. Applies a Laplacian kernel to each frame and detects lines with a Hough Transform. Four types of lines are detected (top, bottom, left, and right margins) by restricting position and slope. Then, calculates the mean and variance of the four lines for every frame. The camcording is detected when the number of lines detected is greater than a threshold. The movement in camcording is

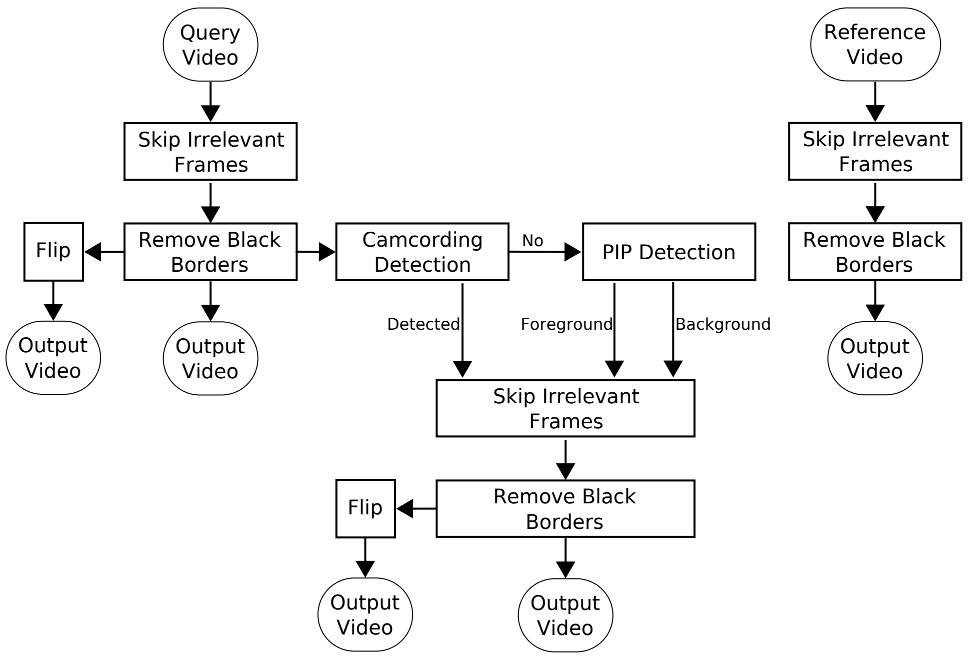

Figure 2. Preprocessing task for query videos and reference videos. Output videos are used by next tasks.

reduced by using a content-relative reference point. Finally, camcording is reverted by mapping detected quadrilateral into a 4:3 rectangle.

Our implementation for PIP and camcording detection are far from perfect as they have a precision of about 50% and a false alarm rate of about 50%, thus original queries are kept with new queries. New queries are treated as independent queries up to the copy localization task, where results are combined with original queries. As a result of this task, the number of query videos used by next tasks increased from 1,608 to 5,378.

3.2 Frame Sampling Task

This task divides each video in groups of similar consecutive frames. In the following, we will refer to these groups as *video chunks*. Thus, every query video and reference video is partitioned into video chunks. Note that a video shot is a serie of interrelated consecutive frames taken contiguously by a single camera and representing a continuous action in time and space [7]. We do not divide videos into shots, because first and last frames of a shot may be very different. A keyframe is the frame which can represent the salient content of a shot [18]. We do not extract keyframes, because we use a whole group of frames for feature extraction instead of representative frames.

For creating video chunks, we compare two frames by calculating the maximum difference between intensity of pixels after scaling both frames to 20×15. For efficiency and for avoiding noisy frames, we extracted three frames per second. Then, we compared each extracted frame with the first frame of the current chunk. When the difference is smaller than a threshold, the frame is added to the current video chunk, and when it is larger than the threshold, the current chunk is finished and a new video chunk is started with this frame.

The 11,524 reference videos (containing 39,463,431 frames) produced 3,967,815 reference video chunks. The 5,378 query videos from processing task (containing 11,526,076 frames) produced 990,246 query video chunks.

3.3 Feature Extraction Task

This task calculates one or more global descriptors for representing a whole video chunk. In our experiments, three global descriptors were extracted for each frame:

- **Edge Histogram**. Based on MPEG-7 descriptor [14], captures the spatial distribution of edges in a frame. Unlike the original definition, we use 10 orientations and we quantize each histogram bin uniformly into a 8-bit value, resulting a vector of 160 dimensions.
- **Gray Histogram**. Converts a frame to gray scale, divides it into 3×3 blocks, and for each block calculates a 20 bins histogram. Each histogram bin is uniformly quantized into a 8-bit value, resulting a vector of 180 dimensions.

- **Color Histogram**. Divides a frame into 2×2 blocks and for each block calculates a histogram of 16 bins for each R, G, and B channel. Each histogram bin is uniformly quantized into a 8-bit value, resulting a vector of 192 dimensions.

Finally, we defined the global descriptor for a video chunk as the average of the global descriptor for each of its frames [15]. Thus, each video chunk is represented by three descriptors: the Average Edge Histogram (AEH), the Average Gray Histogram (AGH), and the Average Color Histogram (ACH) (160, 180 and 192 bytes, respectively). The following table shows the space used by reference video chunks and query video chunks:

	reference	**query**
AEH	605 MB	151 MB
AGH	681 MB	170 MB
ACH	727 MB	181 MB

3.4 Similarity Search Task

Let $\mathcal{Q}$ be the set of query video chunks and $\mathcal{R}$ be the set of reference query chunks. The similarity search tasks takes each query video chunk $Q \in \mathcal{Q}$ and determines the most similar reference video chunks $R \in \mathcal{R}$. For measuring the degree of similarity between two video chunks a temporal distance function is defined. The search is an approximated similarity KNN+range search (i.e., approximated search of the K closest objects to a query object inside a range threshold τ).

3.4.1 Distance Function

Let m be the number of descriptors extracted for a video chunk, and let dim_i and γ_i be dimensionality and a similarity function for i^{th} descriptor, $i \in \{1, ..., m\}$. In our experiments we defined γ_i as L_1 distance (Manhattan) for AEH, AGH and ACH descriptors:

$$\gamma_1(\vec{x}, \vec{y}) = \sum_{j=1}^{dim_i} |x_j - y_j|$$

Let $desc_i(Q)$ be the i^{th} descriptor for video chunk Q, we defined similarity between video chunks Q and R as a weighted combination between its descriptors:

$$\delta(Q, R) = \sum_{i=1}^{m} w_i \cdot \gamma_i(desc_i(Q), desc_i(R))$$

We selected weights $w_1, .., w_m$ using a novel technique that depends on γ_i and δ distance histograms. A distance histogram of a function d is constructed by sampling many pairs of objects $o_1, o_2 \in \mathcal{R}$, evaluating distances $d(o_1, o_2)$, and accumulating them into a histogram [3].

We set initials weights by first calculating distance histograms for γ_i functions. Then, we selected each w_i independently to a value that normalizes to 1 the distance that covers a fraction $\alpha \in (0, 1]$ of pairs on γ_i distance histogram. α should be fixed to a near zero value due that we need smaller distances to be comparable (in our experiments we fixed $\alpha = 0.0001$).

However, depending on visual descriptors that are being combined, sometimes δ is biased to some γ_i. The α-normalization is not accurate when the growth between zero and α of γ_i histograms are too different. Then, we developed a novel technique for correcting weights. Given a distance histogram, the intrinsic dimensionality $\rho = \frac{\mu^2}{2\sigma^2}$ quantifies how hard is to search on that metric space [3]. In our experiments, we realized that a combination of weights that implies a higher ρ for δ is less biased that a combination with a smaller ρ. Then, our correction method is to seek for a local maximum of ρ given a initial set of weights $\{w_1, ..., w_m\}$ from the α-normalization. For maximization, Newton-Raphson method can be used, however we used a simpler approach that iteratively replaces one w_i with $w_i \pm \epsilon$ if that change increases ρ, and ends when every weight have been tested and none was updated.

We defined a temporal distance function between two video chunks as the average similarity between $2W + 1$ consecutive chunks:

$$DIST(Q_s, R_t) = \frac{1}{2W+1} \sum_{d=-W}^{W} \delta(Q_{s+d}, R_{t+d}) \quad (1)$$

3.4.2 Pivot-based Index

A naive approach for similarity search would requiere to evaluate $DIST$ function $|\mathcal{Q}| \times |\mathcal{R}|$ times, i.e. 990,246 $\cdot$ 3,967,815 distance evaluations, where each evaluation requieres at least $160 + 180$ operations. This naive approach would take several months on a desktop computer.

A function d is a *metric* when it satisfies the properties of reflexivity ($d(x, y) = 0$ iff $x = y$); non-negativity ($d(x, y) > 0$ iff $x \neq y$); symmetry ($d(x, y) = d(y, x)$); and triangle inequality ($d(x, y) + d(y, z) \geq d(x, z)$). If every γ_i complies with metric properties (Manhattan is a metric), then δ and $DIST$ also complies with metric properties. Let $\mathcal{P} \subseteq \mathcal{R}$ be a set of video chunks from reference videos, the lower bound function $LB_{\mathcal{P}}$ is defined as:

$$LB_{\mathcal{P}}(Q, R) = \max_{P \in \mathcal{P}} \{|DIST(P, Q) - DIST(P, R)|\} \quad (2)$$

Each object $P \in \mathcal{P}$ is called a *pivot*. Note that if $DIST(P, x)$ is precalculated $\forall x \in \mathcal{Q} \cup \mathcal{R}$, then with just $|\mathcal{P}|$ operations is possible to evaluate $LB_{\mathcal{P}}$. Because $DIST$ satisfies metric properties, then for every pair of video chunks Q and R:

$$DIST(Q, R) \geq LB_{\mathcal{P}}(Q, R) \quad \forall \mathcal{P} \subseteq \mathcal{R} \tag{3}$$

The index structure consists of a $|\mathcal{P}| \times |\mathcal{R}|$ table with distances from each pivot to every reference video chunk. Additionally, for efficient calculation of $DIST$, each video chunk has a reference to the previous and next video chunks.

3.4.3 Approximated Search with Pivots

Given a query video chunk $Q \in \mathcal{Q}$ the first step is to calculate $DIST(P, Q) \forall P \in \mathcal{P}$, and then perform a KNN+range search. Algorithm 1 shows a classic algorithm for pivot-based KNN+range search. It uses Equation 3 to evaluate $DIST$ only when the lower bound is lesser than both the range τ and the K^{th} candidate distance.

Algorithm 1: Classic Pivot KNN+range search.

```
Input: Q query video chunk, $\mathcal{R}$ reference video
       chunks, $K$ number of NNs, $\tau$ threshold for
       range search, $\mathcal{P}$ set of pivots.
Output: List of $K$ nearest neighbors to Q

NNs ← new priority queue
foreach R ∈ $\mathcal{R}$ do
    lb ← $LB_{\mathcal{P}}$(Q, R)                         // see Equation 2
    if lb < $\tau$ and ( size of NNs < $K$ or lb < max
    distance in NNs ) then                      // see Equation 3
        dist ← $DIST$(Q, R)                       // see Equation 1
        if dist < $\tau$ then
            add R to NNs with distance dist
            if size of NNs > $K$ then
                remove max distance object from NNs
return NNs
```

However, in our experiments Algorithm 1 was not fast enough. We tested with different values for τ, K, and $\mathcal{P}$ but the results were not satisfying. Then, we developed a novel technique for approximated searches.

While performing our experiments we realized that the lower bound for the nearest neighbor was usually between the lowest lower bounds. With that property in mind, we developed Algorithm 2 that uses $LB_{\mathcal{P}}$ as an estimator for actual distance: evaluates $LB_{\mathcal{P}}$ for every object, discards objects with $LB_{\mathcal{P}}$ greater that threshold τ, selects the T lowest $LB_{\mathcal{P}}$ values and just for them evaluates $DIST$. Finally, between the T evaluated distances, selects the K nearest neighbors that are lesser than τ. This is an approximated search because there is not guarantee that actual NN's lower bound will be between the T lowest lower bounds.

Algorithm 2: Approximated KNN+range search.

```
Input: Q query video chunk, $\mathcal{R}$ reference video
       chunks, $K$ number of NNs, $\tau$ threshold for
       range search, $\mathcal{P}$ set of pivots, $T$ number of
       lower bounds.
Output: List of approximated $K$ nearest neighbors to
        Q

MinLbs ← new priority queue
foreach R ∈ $\mathcal{R}$ do
    lb ← $LB_{\mathcal{P}}$(Q, R)                         // see Equation 2
    if lb < $\tau$ then
        add R to MinLbs with distance lb
        if size of MinLbs > $T$ then
            remove max distance object from MinLbs
NNs ← new priority queue
foreach R ∈ MinLbs do
    dist ← $DIST$(Q, R)                           // see Equation 1
    if dist < $\tau$ then
        add R to NNs with distance dist
        if size of NNs > $K$ then
            remove max distance object from NNs
return NNs
```

The key difference between classic search and our approximated search is that while Algorithm 1 uses a $LB_{\mathcal{P}}$ value as a lower bound for discarding objects with a high $DIST$, Algorithm 2 compares $LB_{\mathcal{P}}$ values between them, assuming that a low/high $LB_{\mathcal{P}}$ value implies a low/high $DIST$ value, thus $LB_{\mathcal{P}}$ is used just as a cheap $DIST$ estimator.

Algorithm 2 needs that $LB_{\mathcal{P}}$ be a good estimator for $DIST$. Tightness between $LB_{\mathcal{P}}$ and $DIST$ depends on the size and quality of $\mathcal{P}$. Given two sets of pivots $\mathcal{P}_1 \subset \mathcal{P}_2$ then $LB_{\mathcal{P}_1}(a, b) \leq LB_{\mathcal{P}_2}(a, b) \forall a, b$. Then, selecting more pivots usually implies better approximations, but also implies more operations. With a better approximation, however, a smaller T is necessary for selecting actual NNs. Thus, there is a tradeoff between the $|\mathcal{P}|$, T and approximation. Note that when T tends to $|\mathcal{R}|$ the result of Algorithm 2 tends to Algorithm 1 independently of $\mathcal{P}$.

3.4.4 Pivot selection and evaluation

Pivot selection is critical for Algorithm 2. A naive approach for selecting pivots is to select objects randomly in $\mathcal{R}$. However, a key property for good pivot selection is that pivots should be far away between each other [17]. Sparse Spatial Selection (SSS) [2] uses this property for selecting pivots incrementally. Algorithm 3 shows our implementation for SSS that first randomizes $\mathcal{R}$ and then selects sparse pivots.

Because $LB_{\mathcal{P}}$ is a lower bound of $DIST$, the higher

Algorithm 3: Pivot selection, based on SSS [2].

```
Input: $\mathcal{R}$ set of reference video chunks, treshold
       minimum distance between pivots.
Output: $\mathcal{P}$ set of pivots
$(A_0, ..., A_{|\mathcal{R}|}) \leftarrow$ randomize objects in $\mathcal{R}$
$\mathcal{P} \leftarrow \{A_0\}$
foreach $A_i \in (A_1, ..., A_{|\mathcal{R}|})$ do
    if $\forall\ p \in \mathcal{P},\ \ DIST(p,\ A_i) \geq treshold$ then
        $\mathcal{P} \leftarrow \mathcal{P} \cup \{A_i\}$
return $\mathcal{P}$
```

the value of $LB_{\mathcal{P}}$ the tighter to $DIST$. Then, the evaluation algorithm samples many pairs of objects Q and R, calculates $\mu_{\mathcal{P}}$ that is the average value of $LB_{\mathcal{P}}(Q, R)$, and selects the set of pivots $\mathcal{P}$ that has the higher $\mu_{\mathcal{P}}$. Because the pivot selection depends on randomization of $\mathcal{R}$, we selected k sets of pivots, we compared their respective $\mu_{\mathcal{P}}$, and we kept the set of pivots with higher $\mu_{\mathcal{P}}$.

3.5 Copy Localization

For a query video Q with video chunks $\{Q_1, ..., Q_t\}$, the input to this task is a set $\{S_1, ..., S_t\}$ where $\forall i \in \{1, ..., t\}\ S_i \subset \mathcal{R}$, S_i is a list with the nearest neighbors for query video chunk Q_i, and $0 \leq |S_i| \leq K$. The output of this task is a list of copy detections candidates, where each copy detection is composed of the query video segment (start/end time), the reference video offset, and a copy detection score. An offset is the time than needs to be added to query video segment times for getting reference video segment times, i.e., *query start + offset = reference start* and *query end + offset = reference end*.

We designed a voting algorithm for copy localization. First, it compares every Q_i with its neighbors in S_i recollecting every video reference candidate and minimum and maximum offsets. Then, for every reference video and offset interval a copy detection is performed with Algorithm 4.

For each reference video and offset candidate, Algorithm 4 returns query video segment (start/end) and a copy detection score. The score is the sum of votes of supporting reference video chunks. Each supporting video chunk and its vote is calculated with Algorithm 5. $MatchVote$ is the value for a supporting vote (in our implementation is 1), that is weighted according to the relevance of distance and position of the voter chunk. The relevance of a distance is a value near 0 when distance is τ, and is 1 when distance is 0, the relevance of a position is a value near 0 when voter is the K^{th}, and is 1 when is the 1^{st}, in-between values should be assigned between those limits. $MissCost$ is the cost when there is not a reference chunk supporting the detection,

Algorithm 4: Copy detection algorithm.

```
Input: $\{Q_1, ..., Q_t\}$ query video, $\{S_1, ..., S_t\}$ nearest
       neighbors, V reference video candidate, Offset
       offset interval candidate.
Output: start, end and score for copy detection in
        video V with offset Offset
(start, end, score) $\leftarrow$ (null, null, 0)
(cstart, cend, cscore) $\leftarrow$ (null, null, 0)
foreach $S_i \in \{S_1, ..., S_t\}$ do
    (voter, vote) $\leftarrow$ CalculateVote($Q_i$, $S_i$, V, Offset)
    cscore $\leftarrow$ cscore + vote
    if cscore < 0 then
        (cstart, cend, cscore) $\leftarrow$ (null, null, 0)
    else if vote > 0 then
        if cstart is null then
            cstart $\leftarrow$ voter
        cend $\leftarrow$ voter
        if cscore > score then
            (start, end, score) $\leftarrow$ (cstart, cend, cscore)
return (start, end, score)
```

this value should be a negative value for penalizing detections with discontinuities (in our implementation is -.1).

As a final step, this task eliminates eventual overlaps between detections (by keeping the detection with higher score), joins detections for videos that belongs to the same original query video (created by preprocessing task), and reports detections with best scores.

We set the decision thresholds by inspecting the density of detection scores. For Nofa profile, we tested two different thresholds (`nofa.ehdNgryhst` a threshold of 7 votes, and `nofa.ehdNghT10` a threshold of 10 votes). However, as shown in next section, both thresholds resulted to be too low.

Algorithm 5: *CalculateVote* function.

```
Input: $Q_i$ query video chunk, $S_i$ list of nearest
       neighbors for $Q_i$, V reference video candidate,
       Offset offset interval candidate.
Output: voter best matching chunk, vote score of
        match.
(voter, vote) $\leftarrow$ (null, $MissCost$)
foreach $(R_k,$ distance$) \in S_i$ do
    if $R_k \in$ V and offset($Q_i$,$R_k$) $\in$ Offset then
        v $\leftarrow$ $MatchVote$
                $\times$ relevance of distance between $[0, \tau]$
                $\times$ relevance of $k$ between $[1, K]$
        if v > vote then
            (voter, vote) $\leftarrow$ ($R_k$, v)
return (voter, vote)
```

4 Results Analysis

Visual transformations in TRECVID 2010 are: T1: simulated camcording; T2: PIP original video in foreground; T3: insertions of pattern; T4: strong reencoding; T5: change of gamma; T6: 3 transformations between blur, change of gamma, frame dropping, contrast, compression, ratio, and white noise; T8: 3 transformations between crop, shift, contrast, caption, flip, insertion of pattern, and PIP original video in background; T10: random combination of 3 previous transformations.

The evaluation of a submitted Run relies on three measures:

- **NDCR**: Measures the effectiveness of the detection. The closer to zero the better the effectiveness of the Run. For each profile, a trivial NDCR of 1.0 can be obtained by submitting an empty Run, thus a good result should not be greater that 1.0.
- **F1**: Measures the accuracy in localization after a copy has been detected correctly. The closer to 1.0 the better the accuracy.
- **Mean processing time**: Measures the efficiency for processing queries.

These three measures are calculated at a submitted decision threshold. Additionally, Optimal NDCR and Optimal F1 are calculated by cutting the Run at the optimal decision score. TRECVID calculates these measures separately for each transformation, and for easier comparing purposes we include the average result for all transformations.

The submitted decision threshold for our four Runs resulted to be too low. This is because we lacked of training data to support an accurate selection of a decision threshold. Then, the actual NDCR for our four submissions were too high (much greater than 1.0). Due to this, our results analysis is based on Optimal NDCR and Optimal F1 rather than NDCR and F1 at the submitted threshold.

The evaluation considered 56 audio+visual transformations for the query videos (8 visual transformations and 7 audio transformations). Two profiles were evaluated: Balanced and No False Alarms (Nofa). 22 teams participated in the evaluation. Each team submitted 4 Runs, which resulted in 37 submissions for Nofa profile (with 14 visual-only Runs), and 41 submissions for Balanced profile (with 15 visual-only Runs). We stated that a Run is visual-only when its results for NDCR, F1 and processing time are identical for the 7 audio transformations in a same visual transformation (thus, its results are not influenced by changes in audio).

In our experiments, we tested two descriptor combination for δ function in Equation 1:

- AEH and AGH descriptors (`balanced.ehdNgryhst`, `nofa.ehdNgryhst`, and `nofa.ehdNghT10` Runs):

$$\begin{aligned}\delta(Q,R) = \quad & 0.069686411 \cdot L_1(\mathrm{AEH}(Q), \mathrm{AEH}(R)) \\ & + 0.090415913 \cdot L_1(\mathrm{AGH}(Q), \mathrm{AGH}(R))\end{aligned}$$

- AEH and ACH descriptors (`balanced.ehdNclrhst`):

$$\begin{aligned}\delta(Q,R) = \quad & 0.068073519 \cdot L_1(\mathrm{AEH}(Q), \mathrm{AEH}(R)) \\ & + 0.045144545 \cdot L_1(\mathrm{ACH}(Q), \mathrm{ACH}(R))\end{aligned}$$

Weights for these two combinations comes from the weight selection algorithm presented in section 3.4.1.

Tables 1, 2, and 3 and Figure 3 shows P-VCD system's results for Nofa and Balanced profile. Runs `nofa.ehdNgryhst` and `nofa.ehdNghT10` only differs on the submitted threshold, thus both have the same results for Optimal NDCR and Optimal F1. For both Runs, Optimal NDCR is better than the median for every transformation, its best results are achieved on T3 and T4 and its worst result on T5 and T6. The Optimal F1 value tend to be around the median, but it has its worst localization on T1.

Run `balanced.ehdNgryhst` had an Average Optimal NDCR of .597 (14th global rank and 1st visual-only rank) with an Average Optimal F1 of .820 (15th global rank and 2nd visual-only rank). Its best results are achieved on T2 and T3 and its worst results on T5.

Run `balanced.ehdNclrhst` had an Average Optimal NDCR of .658 (16th global rank and 3rd visual-only rank) with an Average Optimal F1 of .820 (16th global rank and 3rd visual-only rank). Its best results are achieved on T2 and T3 and its worst results on T6.

Mean Processing Time is higher than median in all Runs, in particular for camcording and PIP transformations, mainly because the preprocessing task created more query videos in those cases.

For all Runs, the parameters for Algorithm 2 were K=6, τ=6, $|\mathcal{P}| = 9$, and T=.001$|\mathcal{R}|$. We fixed these parameters by first deciding the amount of time that similarity search should take (we decided that the search should take no more than 24 hours total for all queries), then we tested with different values for T and $|\mathcal{P}|$ for complying with that restriction. In Equation 1, we set W=1 thus $DIST$ function needs more than 1,000 operations to be evaluated, but $LB_{\mathcal{P}}$ estimated it with just 9 operations, and actual $DIST$ is evaluated just on 0.1% times (3,967 evaluations for each query chunk). We made our tests on a Intel Q9400 CPU (2.66 GHz $\times$ 4 cores) with 4 GB RAM on a GNU/Linux 2.6.18.

In summary, the results for our submitted Runs were positioned above the median for Optimal NDCR and

Optimal F1, and considering just visual-only Runs, they were the bests positioned for Balanced and Nofa profile. The results for Balanced profile show that a combination of edge histogram and gray histogram is slightly better than edge histogram and color histogram. The results also show that our pivot-based approximation enables to discard 99.9% of $DIST$ evaluations and still have good effectiveness, and that global descriptors can achieve competitive results with TRECVID transformations

5 Conclusions

In general, we are satisfied with our results, especially considering that: this is our first participation in TRECVID, our video copy detection system uses rather simple global descriptors, we do not use any audio information, and we achieved this results on a standard desktop computer. It is well known that local descriptors should be used for complex transformations [11], however we have shown that, with a preprocessing task, an indexing structure, and a good approximation technique, global descriptors can achieve competitive results with TRECVID transformations and even beat many systems that use local descriptors and/or combine audio and visual information.

Another interesting property is that δ function in Equation 1 combines descriptors at the similarity search task, thus enabling a novel way for fusing audio and visual information. We plan to work on this issue in a future. Other issues we plan to address are: analyze the impact of approximated search parameters on detection result, test other distance functions, and test the fusion with local descriptors.

References

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[2] Benjamin Bustos, Oscar Pedreira, and Nieves Brisaboa. A dynamic pivot selection technique for similarity search. In *Proc. of the intl. workshop on Similarity Search and Applications (SISAP)*, pages 105–112. IEEE, 2008.

[3] E. Chávez, G. Navarro, R. Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[4] Matthijs Douze, Adrien Gaidon, Herve Jegou, Marcin Marszalek, and Cordelia Schmid. Inria Lear's video copy detection system. In *TRECVID*, 2008.

[5] Vishwa Gupta, Gilles Boulianne, and Patrick Cardinal. CRIM's content-based audio copy detection system for TRECVID 2009. In *Proc. of the intl. workshop on Content-Based Multimedia Indexing (CBMI)*, 2010.

[6] Arun Hampapur and Ruud Bolle. Comparison of distance measures for video copy detection. In *Proc. of the IEEE intl. conf. on Multimedia and Expo (ICME)*, pages 737–740. IEEE, 2001.

[7] Alan Hanjalic. Shot-boundary detection: unraveled and resolved? *IEEE Trans. on Circuits and Systems for Video Tech.*, 12(2):90–105, 2002.

[8] Alexis Joly, Olivier Buisson, and Carl Frélicot. Content-based copy retrieval using distortion-based probabilistic similarity search. *IEEE Trans. on Multimedia*, 9(2):293–306, 2007.

[9] Changick Kim and Bhaskaran Vasudev. Spatiotemporal sequence matching for efficient video copy detection. *IEEE Trans. on Circuits and Systems for Video Techn.*, 15(1):127–132, 2005.

[10] Julien Law-To, Olivier Buisson, Valerie Gouet-Brunet, and Nozha Boujemaa. Robust voting algorithm based on labels of behavior for video copy detection. In *Proc. of the intl. conf. on Multimedia (MM)*, pages 835–844. ACM, 2006.

[11] Julien Law-To, Li Chen, Alexis Joly, Ivan Laptev, Olivier Buisson, Valerie Gouet-Brunet, Nozha Boujemaa, and Fred Stentiford. Video copy detection: a comparative study. In *Proc. of the intl. conf. on Image and Video Retrieval (CIVR)*, pages 371–378. ACM, 2007.

[12] Duy-Dinh Le, Sebastien Poullot, Michel Crucianu, Xiaomeng Wu, Michael Nett, Michael E. Houle, and Shin'ichi Satoh. National institute of informatics, japan at TRECVID 2009. In *TRECVID*, 2009.

[13] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[14] B. S. Manjunath, Jens-Rainer Ohm, Vinod V. Vasudevan, and Akio Yamada. Color and texture descriptors. *IEEE Trans. on Circuits and Systems for Video Tech.*, 11(6):703–715, 2001.

[15] A. Mufit Ferman, S. Krishnamachari, A. Murat Tekalp, M. Abdel-Mottaleb, and R. Mehrotra. Group-of-frames/pictures color histogram descriptors for multimedia applications. In *Proc. of the intl. conf. on Image Processing (ICIP)*, pages 65–68. IEEE, 2000.

[16] Sébastien Poullot, Olivier Buisson, and Michel Crucianu. Z-grid-based probabilistic retrieval for scaling up content-based copy detection. In *Proc. of the intl. conf. on Image and Video Retrieval (CIVR)*, pages 348–355. ACM, 2007.

[17] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search: The Metric Space Approach (Advances in Database Systems)*. Springer, 2005.

[18] Yueting Zhuang, Yong Rui, and Thomas Huang. Video key frame extraction by unsupervised clustering and feedback adjustment. *Journal of Computer Science and Technology*, 14:283–287, 1999.

	T1	T2	T3	T4	T5	T6	T8	T10	Average T1–T10
Optimal NDCR	.977	.631	.269	.262	.769	.708	.562	.708	.611
Global Rank	12th	9th	8th	8th	15th	15th	8th	12th	10th of 37
Visual-Only Rank	1st	1st	1st	2nd	5th	5th	1st	2nd	1st of 14
Optimal F1	.484	.946	.878	.877	.753	.838	.931	.916	.828
Global Rank	21st	5th	19th	14th	24th	20th	12th	15th	14th of 37
Visual-Only Rank	5th	1st	7th	5th	8th	6th	5th	5th	1st of 14
Mean processing time [s.]	269	170	108	70	84	75	126	123	128
Global Rank	25th	24th	20th	14th	18th	16th	23th	23th	23th of 37
Visual-Only Rank	12th	12th	8th	5th	6th	7th	11th	11th	11th of 14

Table 1. Results for `nofa.ehdNgryhst` and `nofa.ehdNghT10` Runs. Global Rank is the position between the 37 submitted Runs for Nofa profile. Visual-Only Rank is the position between the 14 Runs that did not use audio information.

	T1	T2	T3	T4	T5	T6	T8	T10	Average T1–T10
Optimal NDCR	.977	.515	.269	.262	.777	.708	.562	.708	.597
Global Rank	19th	9th	9th	9th	20th	18th	14th	17th	14th of 41
Visual-Only Rank	2st	2st	1st	2nd	6th	5th	2st	4nd	1st of 15
Optimal F1	.484	.888	.878	.877	.747	.838	.931	.916	.820
Global Rank	23st	14th	17th	14th	24th	19th	11th	10th	17th of 41
Visual-Only Rank	5th	3st	7th	6th	8th	6th	6th	4th	3st of 15
Mean processing time [s.]	269	170	108	70	84	75	126	123	128
Global Rank	30th	29th	24th	19th	23th	21th	25th	27th	27th of 41
Visual-Only Rank	12th	13th	8th	6th	7th	8th	9th	11th	11th of 15

Table 2. Results for `balanced.ehdNgryhst` Run. Global Rank is the position between the 41 submitted Runs for Balanced profile. Visual-Only Rank is the position between the 15 Runs that did not use audio information.

	T1	T2	T3	T4	T5	T6	T8	T10	Average T1–T10
Optimal NDCR	.962	.454	.346	.608	.638	.808	.762	.685	.658
Global Rank	17th	7th	11th	16th	18th	19th	17th	16th	16th of 41
Visual-Only Rank	1st	1st	2st	3nd	5th	6th	4st	3nd	3st of 15
Optimal F1	.583	.893	.894	.809	.849	.767	.952	.813	.820
Global Rank	21st	11th	14th	22th	19th	24th	6th	21th	16th of 41
Visual-Only Rank	4th	2st	5th	8th	6th	8th	3th	7th	2st of 15
Mean processing time [s.]	276	179	112	70	86	76	130	130	132
Global Rank	31th	30th	26th	20th	24th	22th	28th	28th	28th of 41
Visual-Only Rank	13th	14th	10th	7th	8th	9th	12th	12th	12th of 15

Table 3. Results for `balanced.ehdNclrhst` Run. Global Rank is the position between the 41 submitted Runs for Balanced profile. Visual-Only Rank is the position between the 15 Runs that did not use audio information.

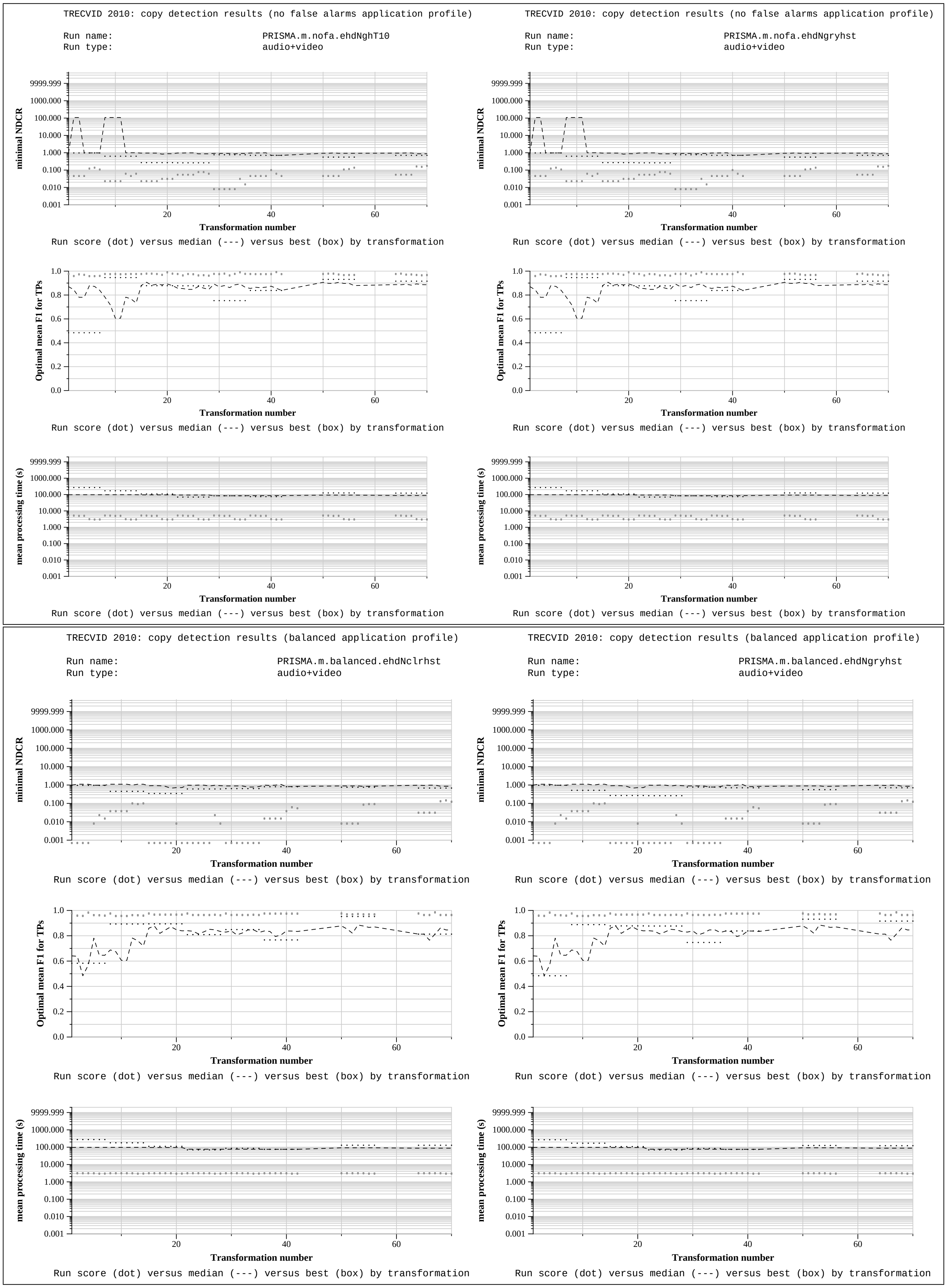

Figure 3. TRECVID results for our four submitted Runs: `nofa.ehdNgryhst`, `nofa.ehdNghT10`, `balanced.ehdNgryhst`, and `balanced.ehdNclrhst`.