A Simple but Effective Approach to Video Copy Detection

Gerhard Roth, Robert Laganière, Patrick Lambert, Ilias Lakhmiri, and Tarik Janati
gerhardroth@rogers.com, laganier@site.uottawa.ca, patrick.lambert@univ-savoie.fr
University of Ottawa, School of Information Technology and Engineering
Ottawa, K1N 6N5, Canada

Abstract

Video copy detection is an important task with many applications, especially since detecting copies is an alternative to watermarking. In this paper we describe a simple, but efficient approach that is easy to parallelize, works well, and has low storage requirements. We represent each video frame by a count of the number of SURF interest points in each of 4 by 4 quadrants, a total of 16 bytes per frame. This representation is tolerant of the typical transformations that exist in video, but is still computationally efficient and compact. The approach was tested on the TRECVID copy detection task, for which approximately 15 different groups submitted a solution. Performance was among the best for localization, and was approximately equal to the median with regards to the false positive/negative rate. However, performance varies significantly with the video transformation. We believe that the change in gamma, and decrease in video quality transformations are the most common in practice. For these two transformations our method works well.

1. Introduction

A copy is a segment of a video derived from another video, usually by means of various transformations. Detecting copies is an important and new topic that provides an alternative to watermarking for copyright control and other applications. There are many possible solutions to this problem, so it is necessary to provide an evaluation framework. Below we list some features that we believe should be part of any successful copy detection system.

Every copy detection system will execute an indexing stage to extract search information from a video database, and then the search stage will use this information to match against a given video query. Since the indexing stage is done off-line, it can be complex and time consuming. However, given the rapidly increasing amount of video data that is being created, the amount of index information saved per video database entry should be as small as possible. By contrast to indexing, the search process should be as fast as possible since it is done on-line. In practice a copy detection system will need to run on parallel hardware, such as multiple CPUs or even GPUs. Therefore the search process should be scalable and easy to parallelize. In summary, a system that satisfies these criteria is what we call efficient. It goes without saying that any copy detection system should perform well and be effective, but how can this be quantified? To be effective the system should be tolerant to the typical transformations that occur in video, such as the quantization due to rencoding, or the insertion of station logos and identifiers. It is not an easy task for a copy detection system to be both efficient and effective.

Keeping these two criteria in mind, we will look at the current set of solutions to the problem. In general, they can be divided into two groups; those that use global features and those that use local features. Typical global features, such as moments, or image statistics, can be extracted quickly and efficiently, and they provide compact index information [5, 6]. While they are also efficient in terms of search, they are not discriminatory enough to provide good performance since they cannot deal with many common video transformations. Given the success of local features such as SIFT in image matching and search, it is natural to attempt to use these features in video copy detection [4, 3]. Their extraction is usually slow, which is not a problem, since this is done off-line. Local features are indeed discriminatory enough to be effective because of their invariance to different image transformations. This means they are also tolerant to many types of video transformations (such as quantization), but not to all global transformations (such as image flip). However, they require a considerable amount of storage per image frame, since there is a high dimensional descriptor associated with each local feature point. This means that systems based on local features are typically not efficient, because of the size of the index file, and the difficulty of parallelizing the search process.

2. Local Index Feature Count

Looking at the past solutions we have come to the conclusion that a combination of local and global features is the best way to solve the problem. In our approach we first extract a set of SURF feature points in each video frame in the database. The SURF algorithm is a modern scale invariant feature detector [2], similar to the well known SIFT features. The SURF features extracted from a typical single video frame are shown in Figure 1 (marked by a cross). Now as is usual for such local features, there is a high dimensional feature vector associated with each of these feature points (most often, 64 integers per interest point). We do not use these feature vectors since this would be too much information to store, and a feature-based comparison across all frames would be prohibitive in terms of computational load. Instead our descriptor for a single frame is simply a count of the number of SURF points in each of sixteen (four by four) equal quadrants. Since there are rarely more than 255 SURF features in a quadrant, we store only sixteen bytes for each video frame (in the rare cases where more than 255 features are detected, we fix the value at 255). Experiments with different image partitioning methods showed that this sixteen byte descriptor is surprisingly discriminatory and therefore tolerant to different video transformations. One reason is the actual number of SURF feature points changes with the characteristics of the image, another is that the descriptors for a number of consecutive image frames must agree for a valid match. Finally, the SURF features themselves have a certain invariance to image transformations due to their nature.

To summarize, we use a combination of local (SURF) and global (16 counts) features, which is both compact and discriminatory. This descriptor is inherently invariant to resizing (up to a certain extent) and can be made invariant to flipping and mirroring just by changing the order of the count vector. SURF features are also quite resistant to image deterioration and coding artefacts. For each video database entry we create an associated index file of size sixteen bytes times the number of frames. This is a very compact representation as Gigs of video data can be represented by just a few Mb of feature count data. For example, if there are 25 frames per second, then one hour of video data with 90,000 frames requires only 1.4 MBytes of index information. A video index file is created in an off-line, pre-processing phase for every video file in the database, at approximately three to five times the video frame rate.

2.1 Video Similarity Measurement

Assume we have a database of videos that have been preprocessed in this fashion to create a set of index files, and that we are given a query video. Once we have computed

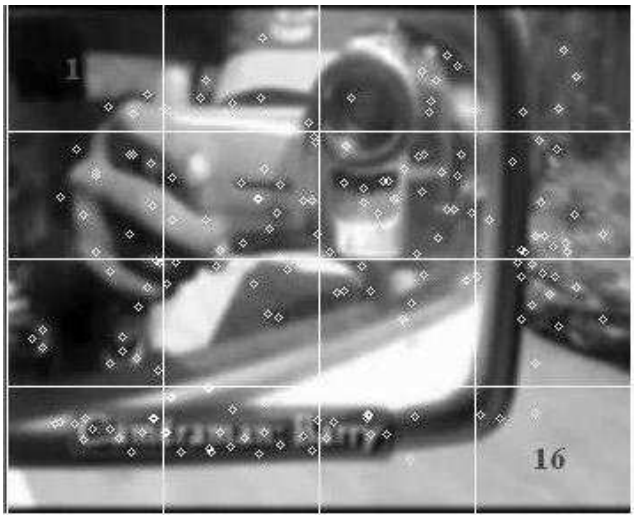

Figure 1. The features (SURF points) in a single frame. Our descriptor is simply the number of feature points in each quadrant [3,7,...,3]

.

the index feature counts for the query video by the same preprocessing, the question is how to find the best match with the feature counts of the video database. This requires us to compare the SURF counts of a video query to the SURF counts for each entry in the video database. Now to actually compare a video query to a database entry we must compare their sequences of SURF feature counts for all possible matching video frames. This is shown diagrammatically in Figure 2 for a given video database entry. First we build a cross table where we compute the normalized $L1$ difference between the vector of 16 SURF feature counts for every query frame and every frame in the database video. When comparing two feature count vectors the normalization process divides their $L1$ difference by the sum of the feature counts for the two vectors, and scales the result to be between 0 and 255. If there are K frames in the query video, and N video frames in the database video, then this takes $O(KN)$ time. However, we subsample the database frames by a factor of at least ten (since typically $N >> K$), so the time to compute each normalized difference is very small. It is also clear that the process of creating this crosstable of normalized differences of the SURF feature counts can be easily parallelized.

We use this crosstable to find the longest matching frame subsequence between the query video and database video. We must search for matching subsequences because it is not necessary that an entire query be contained in the database video, a match for any sub-part of the query video is acceptable. To do this we first threshold each of these difference

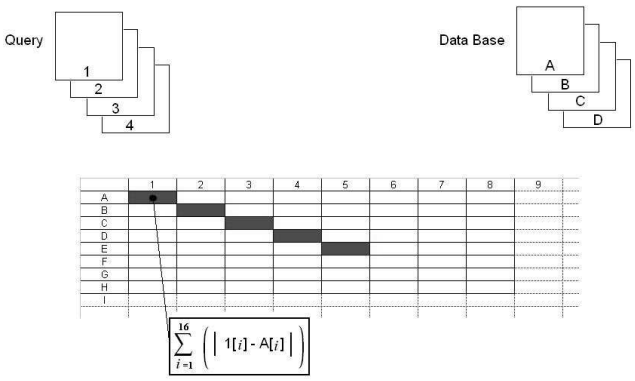

Figure 2. Measuring video similarity. Each frame of the query is compared with each frame of the input video. A cross-table of comparison of the vector of SURF counts is created; the similarity measure corresponds to the longest diagonal with a distance score less than a given threshold.

	A	B	C	D	E	F
0	125	307	243	124	311	245
1	**56**	121	212	125	325	215
2	107	**61**	154	115	299	184
3	245	130	**58**	321	287	183
4	102	177	236	125	145	243
5	225	317	253	115	**43**	322

Table 1. Measuring video similarity. Each frame of the query is compared with each frame of the input video. A cross-table of comparison of the vector of SURF counts is created; the similarity measure corresponds to the longest diagonal with a distance score less than a given threshold. Here with a threshold of 70, a copy is found between query frame 1-5 and database video frame B-F; note the non-matching frames 4,D (possible dropped frames) accepted in the diagonal sequence.

scores in the crosstable to transform this table into a binary form. We found that the exact value of this threshold is not too critical in the detection of a similar video segment, and we use the same threshold value for all our experiments. Then as shown in Figure 1 we find the longest non-zero diagonal sequence in the thresholded crosstable, which is the longest sub-sequence match between the query video and that particular database video. However, as shown in Table 1, we allow for dropped frames and outlier frames by skipping over a small number of matches in the diagonal sequence which do not pass the threshold test. In this way a small number of badly matched frames does not cause an otherwise long matching video sequence to be ignored.

The best match produces a length (number of matched frames) and an average frame difference for each overlapping frame (the normalized $L1$ difference of the vector of 16 SURF counts). The match length along with the starting and ending frames in both the query and match video are also known. It is possible to use only the match length as the decision criteria to reject or accept a match. However, this does not deal with the fact that some good matches have a very short match length, but also have a very low average frame difference. To deal with this we boost the match length artificially when the average frame difference for a match is low. This process is explained in more detail in the Experimental Results section.

2.2 Dealing with Video Transformations

A key issue in copy detection is in dealing with the transformations that occur in video as it is transmitted and stored. Obvious transformations are quantization and the embedding of logos (called insertion of patterns). One of the first systematic approaches to the performance evaluation of copy detection was the Muscle database [1]. Our first approach to copy detection used the PACT representation [8]. However, it did not perform well on the standard transformations in the Muscle database. We mention this because the PACT representation has shown itself discriminatory enough to be successful in place recognition tasks, but it did not succeed in dealing with the typical copy detection transformations. For this reason, we came up with our new representation, the count of SURF features, whose performance in copy detection was much superior.

Since copy detection is an important task, NIST has added it to the TRECVID family of performance evaluation databases [7]. The Muscle database [1] was the basis of the current TRECVID copy detection database, which is a much expanded version. Since matching untransformed video is relatively simple, the TRECVID copy detection task takes each query video and transforms it in a number of different ways. The TRECVID transformations include picture-in-picture, insertion of patterns, rencoding, change in gamma, decreasing the quality, post production transformations, and combination of everything. We show a number of these transformations in Figure 3. The decrease in quality, and post-production transformations are actually a combination of the individual transformations, of which up to 3 can be applied simultaneously. The combination of everything transformation allows for up to 5 of the other possible transformations simultaneously. The idea is to simulate all the changes that might occur in a query video when it

is broadcast, since the transformed version is what must be matched to the database. It is clear that the most difficult task is dealing with these different types of transformations, especially since a query video can have more than one transformation applied to it.

Figure 3. A number of transformations: Picture in Picture, pattern insertion, rencoding, and compression

The transformations which simply modify the image without inserting or deleting any content (such as blur, rencoding, gamma noise) are dealt with automatically by the invariance of the SURF feature point detection process, so they do not require any other specific processing. However, transformations that actually change the video by inserting or detecting part of each frame must be dealt with explicitly. This is done by a preprocessing step which is applied to every query in which a mask image is calculated. This mask image consists of the thresholded variance for each frame pixel over the entire video. The image regions in the mask file that have low variance are often not part of the video. For example, they may be inserted patterns (such as a logo) or even black areas (due to a shift, crop or letterbox transformation). The process of creating the mask image is shown in Figure 4, where this image is displayed for a given query video. The SURF counts for image quadrants which are masked are then adjusted during the search process so that the masked regions are ignored. Different types of mask images for text insertion, crop and shift are shown in Figure 5. This process is applied to each video query, to produce a mask image that is used to adjust the SURF counts during the search process. Of course, if there are no masked pixels then there is no impact on the search.

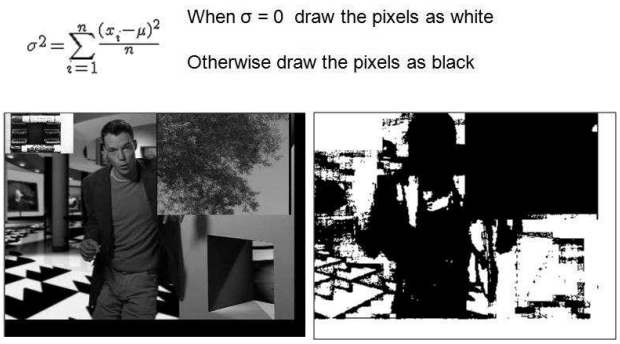

Figure 4. The mask creation process for a single query video.

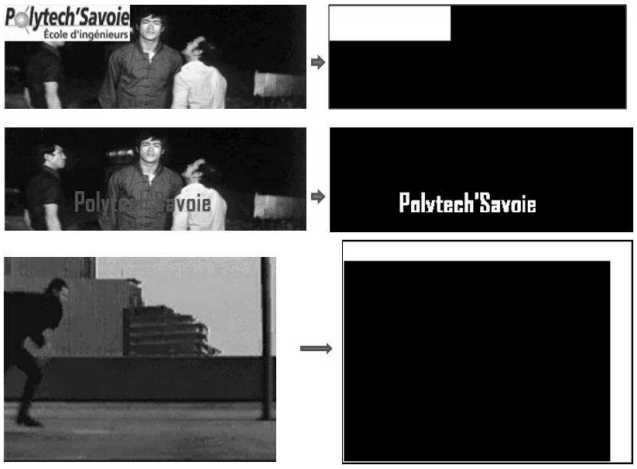

Figure 5. The masks for image and text insertion, along with a shift.

The pre-processing necessary to deal with an inserted video is more complex; here we attempt to detect whether the query video is imbedded inside another video. If so, we

need to extract the query video and process it separately. In practice, the success rate for dealing with insertion of video is very low. It should be noted that this is different than the insertion of still pictures or patterns, which can be dealt with by the mask image. For this reason, in our experiments we do not show any results for the inserted video transformation since we consider it to be uncommon. One of the more difficult of the simple transformations is an image flip (sometimes called a mirror flip). The method used to deal with an image flip (a post production transformation) is shown in Figure 6. With our representation we can simply mirror the SURF count descriptor for each video frame of the query. There is also specific processing for the ratio and shift transformations (members of the decrease in quality and post-production transformations), since these require an adjustment of the region sizes when comparing the image descriptors. An important transformation that is a member of the decrease in quality set is dropping frames, which simulates error in the transmission or storage of video. As we have mentioned in our section on Video Similarity Measurement, this is dealt with implicitly by the matching process which skips over a small number of frames in a longer matching video subsequence that do not pass the match threshold test.

Figure 6. The processing for dealing with image flips

3. Experimental Results

We will now describe our results when processing the TRECVID database for a certain set of transformations. However, first we will revisit the issue of using both the match length and average match value as the decision criteria. The match length is the number of matched frames, and the average match value is the normalized average frame difference between the SURF counts over that match. The difficulty is that simply using the number of matched frames to decide on whether there is a match is sometimes not sufficient. This is because for a certain percentage of good matches the matching frame count is low, so using only the match length as an evaluation criteria would eliminate many true positives. However, we hypothesized that in many of these cases the average match value was also low. In other words, there are sometimes short sequences that are very good matches. To validate this hypothesis, we computed the match length, and the average match value for a small number of videos (approximately 100). These were then visually classified as a positive or negative detection, as shown in Figure 7. In this figure the triangles are the incorrect matches (false positives) and the rectangles the correct matches (true positives) as obtained by manual inspection. It is clear that our hypothesis is correct, and that simply using the match length (which would be a vertical line) would eliminate many true positives. Possibly a function such as the exponential curve drawn in the figure could be used as the decision criteria, but for now we use a simpler approach.

We would like our decision threshold to be a function of both the match length and the average match value score. Our current approach is to simply boost the match length artificially when the average match value is low. Specifically, if the average match value is less than 40, the match length is boosted by 500, if it is 60 or higher there is no boost, and between 40 and 60 the match length is boosted proportionally. What this means is that matches that are shorter in length will be accepted if they have a lower average match value score. In our experiments this simple boosted scoring method performed better than non-boosted approach for every TRECVID transformation.

We now provide more details on the TRECVID methodology. TRECVID provides a test collection of about 500 hours of video and a non-test collection of about 200 hours, which have no overlap. From the test collection a set of about 140 queries are randomly selected, with a length of between one/half to five minutes. About half of these queries are embedded in a randomly selected video from the non-test collection which is of similar length. Finally, about 60 videos of length one/half to five minutes are randomly selected from the non-test collection, which by definition do not occur in the test collection. So one third of the queries have no match, one third are a complete match, and one third are a partial match.

Given a query video the copy detection task should find the sub-part of that video which matches a section of any of the test videos. The fact that one third of the query videos do not have any match in the test collection makes the problem more difficult, and means that the copy detection process must also return “no-match” in certain cases. If a match is

found, the copy detection process should specify the match location and length in the query video, and in the test video collection. It should be pointed that we cannot assume that there is at most one match to the query video in the test video collection, since this test video collection many contain duplicates. It is therefore important for the copy detection process to find all the matches for a given query video, if they exist, and not just the best match.

In TRECVID the search is made more difficult by taking each of the original 200 query videos and transforming each query by the 7 different transformations. We show the results when these transformations are picture-in-picture, insertion of patterns, strong rencoding, change of gamma, decrease of quality, post production alterations, and combination of everything. This produces a total of 1400 queries that are to be matched. This is the final set of queries used in the video only search component of TRECVID. While TRECVID has an audio component to the search, and a combined audio/video component, in this paper we will discuss only the video component of the search process.

There are a number of performance measures used in TRECVID, one of which is localization error. Given that a copy has been detected, how accurately is the position and length of the query localized in the test video? Our video matching approach performs well in terms of the localization error, which is a consequence of the construction of the algorithm. In fact, it is one of the best TRECVID submissions in terms of the localization error. Since a frame is represented with only 16 bytes, it is possible to compare each frame of the query with each frame of the video database, so if we get a match we know the location very accurately.

The next performance measure used in TRECVID is the speed of the copy detection which is difficult to evaluate in an objective fashion. Our system is slower than the median TRECVID submission in terms of execution speed, and takes a few minutes to search 400 hours of video for a typical query. However, any practical copy detection system will need to run on parallel hardware. The fastest approaches in TRECVID use complex data structures (such as K-D trees) to speed up their search process, but the use of these global data structures makes parallelization more difficult. The reason is that once such a data structure has been created for a given set of videos in the test database then this is an implicit level of granularity for any parallelization. If the copy detection process then needs to be redistributed to a different number of processors, then this global data structure must be recomputed. By contrast, our search algorithm is very simple, and operates independently for every pair of query/test videos that must be processed. So it is very easy to run this search process in a distributed fashion, and also to distribute the index creation process.

The last performance measure used in TRECVID is the false positive/negative rate. In fact, TRECIVD uses various profiles that quantify the cost of a false positive relative to a false negative. The details of these different profiles are discussed in [7]. In this paper we will use the results from the TRECVID video only evaluation (where the ground truth is known), but we will display these results using the standard response (ROC) curve. In this type of 2D plot the X axis is the probability of a false positive (saying the query is a copy when it is not), and the Y axis is the probability of a true positive (saying the query is a copy when it truly is). Every point in this 2D plot represents an operating point, which is the false positive/true positive probability of detection that results for using a particular decision threshold to classify the queries.

In our case the decision threshold is the boosted matched length, which is used to classify a query as a copy, or not. For every set of query/test video pairs the copy detection process produces a decision value. If this decision value is above a given threshold, then this query is labelled as being a copy that exists in that particular test video. Every point on the ROC curve represents the set of query results for a given value of the decision threshold. The copy detection process is free to use any decision process it deems appropriate, but it is expected that the higher the decision value for a given query/test video pair, the more likely it is that the query is a copy. The ROC curve shows the system performance clearly, and leaves the actual decision threshold that will be used open. The closer the ROC curve to a step function, the better the performance. For copy detection increasing this threshold makes it more likely that there are fewer false alarms but also decreases the chance of a true positive, that is of finding a copy. The value of the decision threshold that is appropriate is therefore always a trade-off.

In Figure 8 we see the ROC curve for transformation 6, which is a decrease in quality plotted with and without boosting. Here we see the effect of using a decision threshold which is a combination of match length, and average match value, relative to a threshold that uses only match length. From this graph we see that the using both criteria produces a curve which is higher on the graph, which means that the performance using boosting is superior. This superiority was clear over all the TRECVID transformations, so from now we will show only boosted results.

In Figure 9 we see the ROC curve for all seven transformations. The results vary, but it should be kept in mind that many of these transformations are very extreme, and unlikely to occur in practice. In our opinion the most likely transformations to occur in practice are number five (change in gamma), and especially number six (decrease in quality) that was shown in the previous graph. As video is transferred and rencoded it often decreases in quality so we will focus our discussions on these transformations. We note that for these two transformations it is easy to find a decision threshold that results in a 70 percent detection rate, and

about a 20 percent false alarm rate. While this may not seem like good performance, in practice the decrease in quality transformations used in TRECVID are more dramatic than what is found in most applications. So these performance curves are actually a lower bound on the performance of our system, which is more than sufficient in practice.

4. Conclusion

In terms of performance our method ranked in the middle of the TRECIVD submissions for copy detection. However, on the more common transformations (gamma and image quality change) it performs well. In terms of the localization our approach is one of the best, as would be expected from a system that compares the video sequences for overlap. More importantly, we believe that this method satisfies many of our design goals.

We are working on a number of improvements; implementing the process on GPU hardware, having a dynamic frame subsampling strategy, improving the decision scoring by a better combination of match length and average match value, and using simpler features (such as Harris or Fast corners).

References

[1] http://www-rocq.inria.fr/imedia/civr-bench/data.html.

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

[3] M. Hertier and S. Foucher. Video copy detection using latent aspect modeling over sift matches. In *CRIM Notebook Paper - TRECVID*, 2008.

[4] Y. Ke and R. Sukthankar. An efficient parts-based near-duplicate and sub-image retrieval system. In *IEEE Transactions on Audio, Speech and Language Processing*, volume 16, pages 396–407, 2008.

[5] C. Kim and B. Vasudev. Spatiotemporal sequence matching for efficient video copy detection. In *IEEE Trans. Circuits Syst. Video Technology*, volume 15, pages 127–132, 2005.

[6] A. Kimura and K. Kashino. A quick search method for audio signals based on a piecewise linear representation of feature trajectories. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 869–876, 2004.

[7] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press.

[8] J. Wu and M. Rehg. Where am i:place instance and category recognition using spatial pact. In *Computer Vision and Pattern Recognition*, 2008.

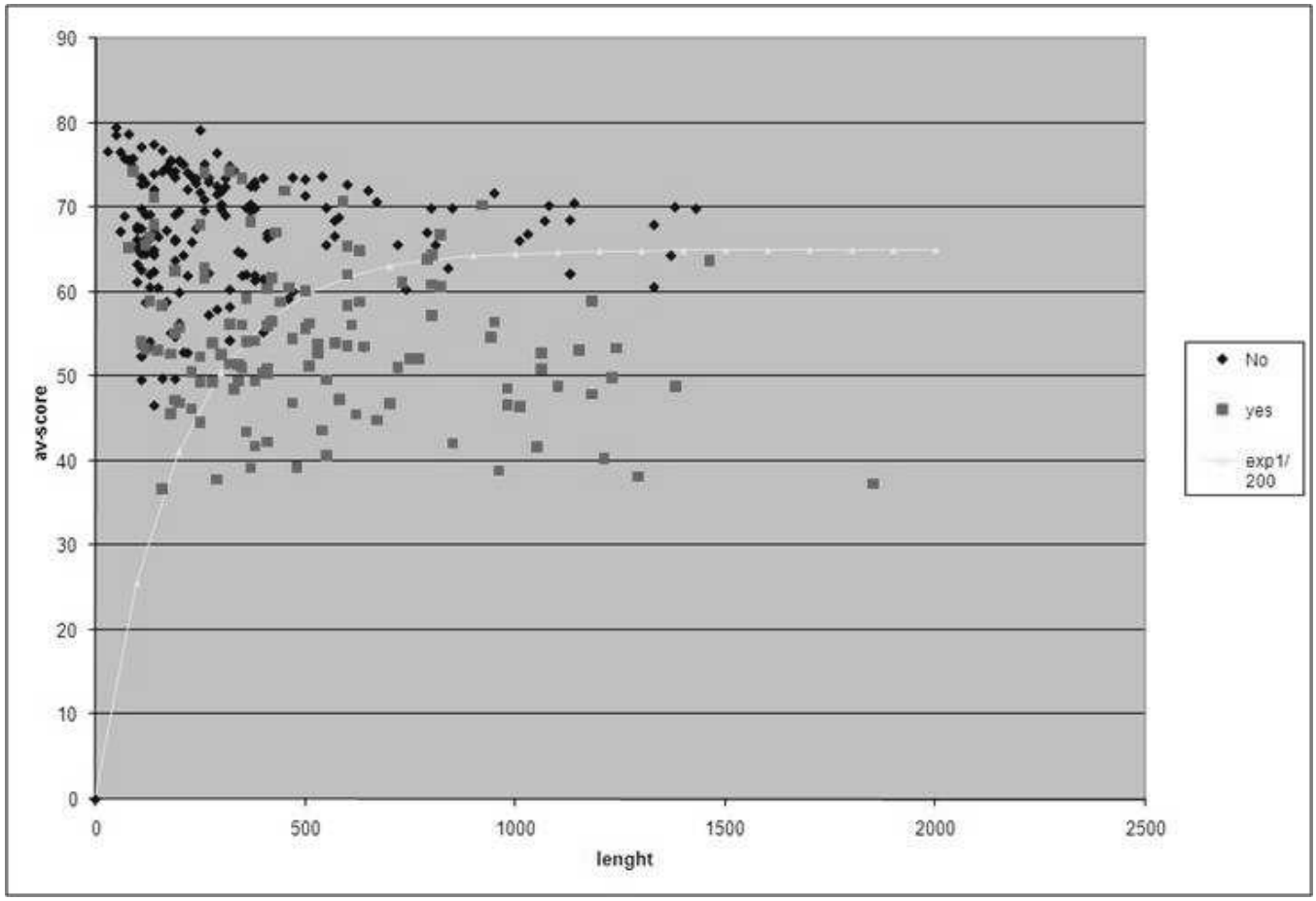

Figure 7. The thresholding criteria, based on match length and average match score.

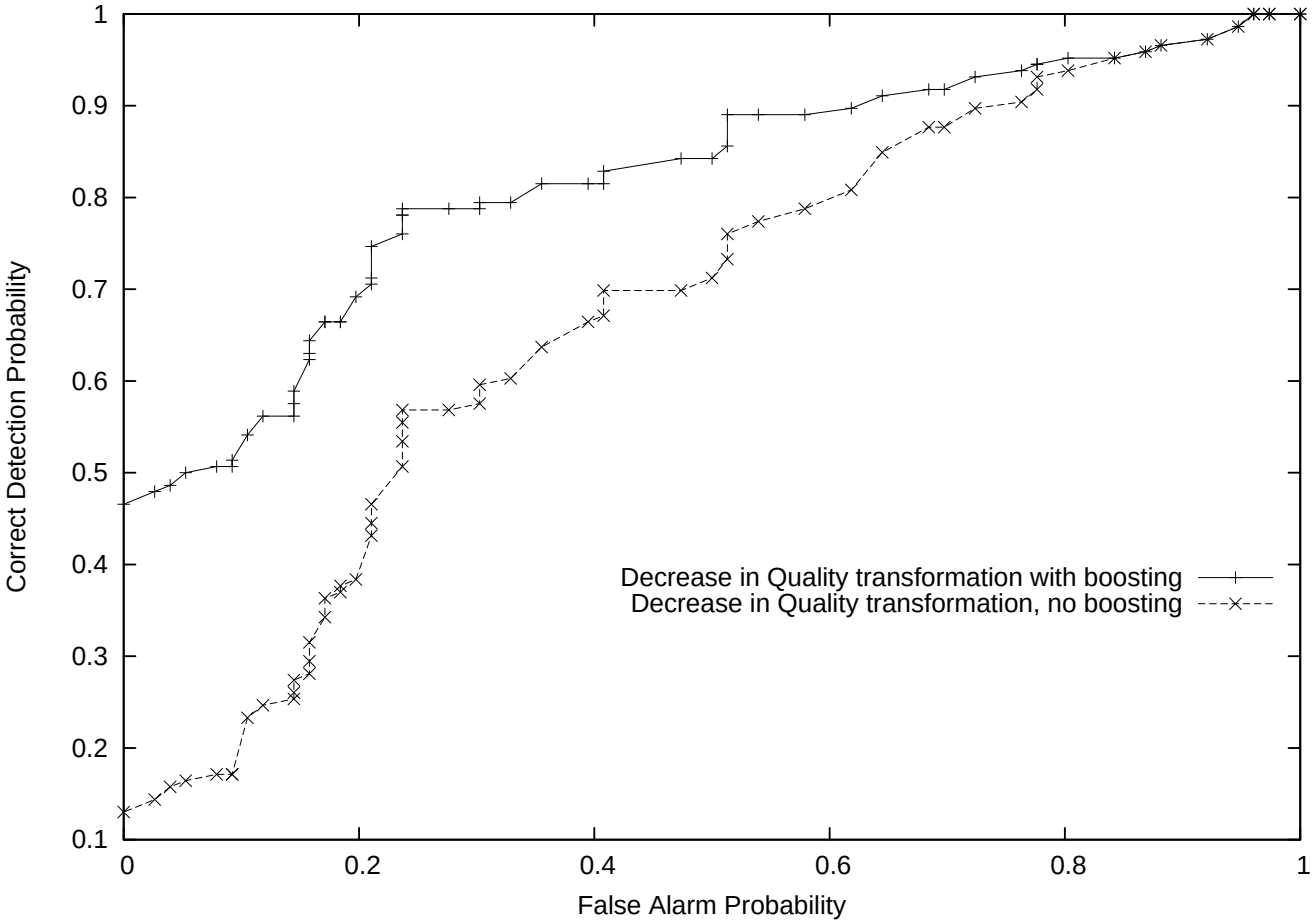

Figure 8. The ROC curve for decrease in quality transformation, with and without boosting.

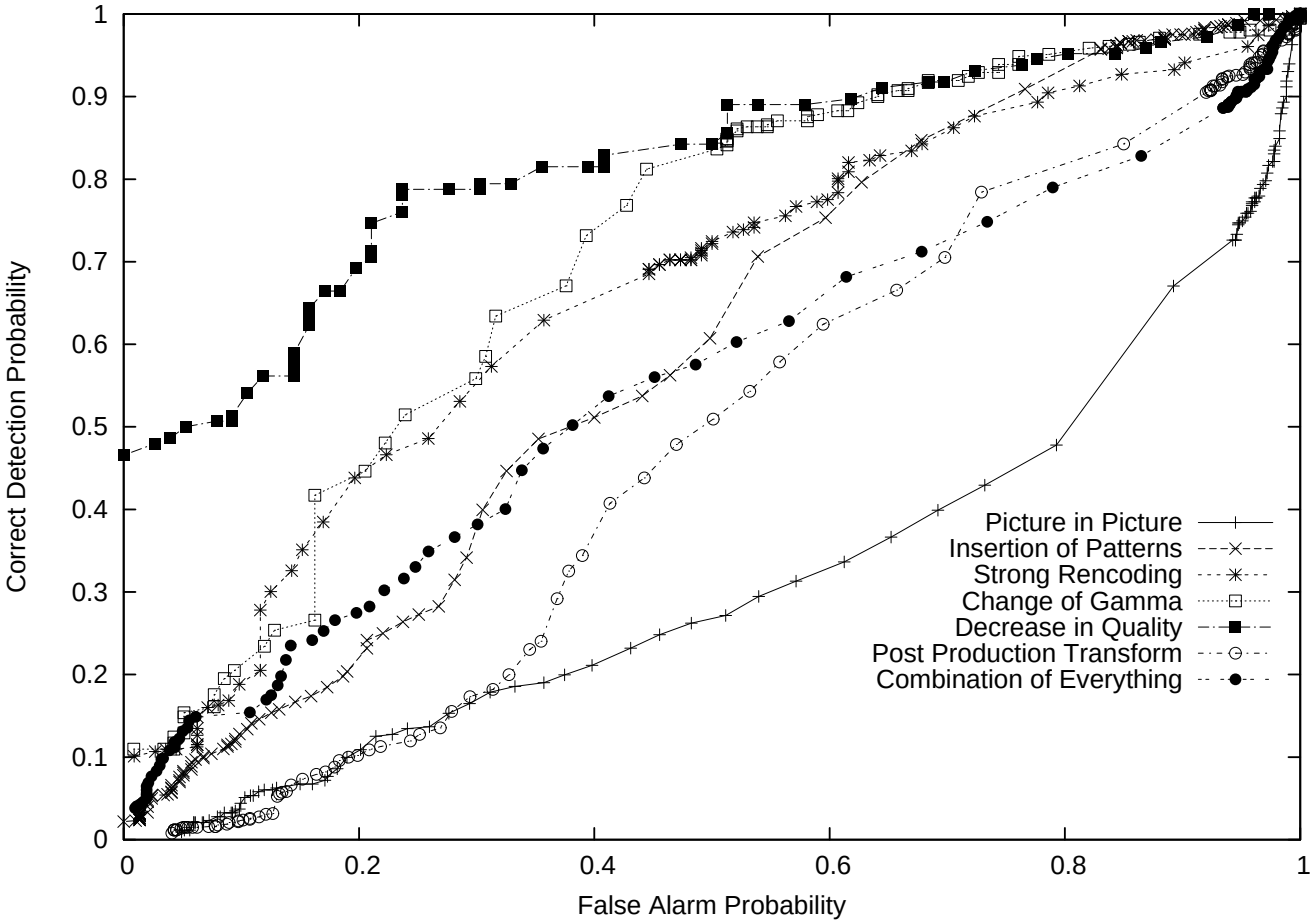

Figure 9. The boosted ROC curve for all seven transformations.