Efficient Video Filtering of MPEG-7 Streams

Fabrizio Falchi, Claudio Gennaro, Pasquale Savino
ISTI - CNR, Pisa, Italy
{fabrizio.falchi, claudio.gennaro, pasquale.savino}@isti.cnr.it

Abstract

The production of digital multimedia content is continuously increasing. With the advent of the digital cable, satellite and terrestrial televisions, thousands of channels are practically available for users. Moreover, multimedia is widely used in many professional applications, such as video program production, video surveillance, and e-learning. Consequently, the use of techniques for video retrieval as well as video filtering is becoming of crucial importance. The adoption of the MPEG-7 standard is a significant step forward in simplifying the video retrieval and filtering. However, the performance issue can be relevant if the retrieval must be accomplished in real time, as in some applications such as the video surveillance or video filtering in general. Moreover, the number of streams to search, the number of queries, and the computational complexity of the feature similarity measure can heavily affect the effectiveness of such real time filtering applications.

In this paper we present the *Pivoted Stream*, a novel approach for efficient filtering of a video stream by using the MPEG-7 descriptors. Our proposal exploits the properties of the metric spaces, in order to reduce the computational load of the filtering receiver.

1 Introduction

Wide access to large information collections is of great importance in many aspects of everyday life: enormous quantities of information in different forms and of different types (text, images, audio, video, etc.) are being produced and archived. For this reason, a significant effort has been spent in studying and developing techniques that support an effective and efficient retrieval of multimedia data. Among other types of information, Audio/Video can be considered today as a primarily mean of communication, due to its richness in informative content and to its appeal. This implies that the development of techniques supporting the retrieval of Audio/Video documents is of primarily importance to enable the access to the general public as well as to professional users of a significant asset of today life. This

process will gain more impetus from the adoption of standards to represent video content (e.g. MPEG-7 [3, 12]). The retrieval process is based on a simple schema: users specify their request needs (e.g. a set of keywords or a sample image) that are translated into a system query; the items in the archive are compared with the user's query, in order to determine if they are relevant for the user's request. In order to process this type of query, it is necessary to determine a set of properties of the objects stored in the archive (usually called features) and a similarity measure to compare queries and archive objects. Video features can be described by using the MPEG-7 standard. In case the similarity measure is *metric*, many possible approaches to create indexes can be adopted. These indexes allow to improve the efficiency of the retrieval process, by comparing the query only with a limited number of objects in the archive [9].

Another important application scenario similar to the one described so far, is the one where information is not stored in an archive, but is flowing to the users continuously in a stream (or more streams). This happens, for example, with information delivered by news agencies or with broadcasted TV programs [11] [14], or in a surveillance system. In this case, an enormous quantity of information arrives to the users that are interested only to a very limited part of it: the process of selecting only the significant information is called *information filtering*. This process can be applied, in general, to describe a variety of processes involving the delivery of information to people who need it: however, the filtering of Audio/Video documents, which is the topic of this paper, is particularly challenging due to the complexity of performing the selection of video material in real-time, possibly from a large number of video streams.

The filtering process is based on a simple schema, which has many similarities with the retrieval process described so far:

- Users specify their information needs that are translated into a system query filter; in many cases a user, or a group of users with similar interests, may specify several filters.
- Data are analyzed in order to extract a set of features that describe their characteristics.
- The items in the stream are compared with the user's query filter in order to determine if they are relevant to the user's request. The item is delivered to the user in case it passes the filter, i.e. if its similarity with the filter is higher than a predefined threshold.

The main difference with information retrieval is due to the difficulty in creating access data structures that can speed-up the filtering process, since a pre-processing of the entire stream is not feasible in real time.

The efficiency problems that derive from the difficulties of using any access data structure, become significant if filtering is performed on Audio/Video data:

- The number of TV channels that are broadcasted is continuously increasing as it is increasing the number of channels that any single user can receive. The selection of appropriate programs and information becomes a problem for personal use as well as for professional uses.

- Video is frequently used for many different applications that go beyond the pure TV broadcasting, e.g. video surveillance, e-learning, etc. All these applications may require efficient real-time filtering of relevant events.

- Similarity matching of queries and Audio/Video data is in general more complex than for other data types, due to the fact that Audio/Video data have a complex structure and that different types of information can be used to represent Audio/Video. For example, we may use the single video frames (or a selection of the video frames), or we may use the audio track (a transcript of the speech, information about sound, etc.), we may use information about the video structure (a video can be seen as composed of shots and scenes), we may use metadata manually associated to the video, or combinations of these features (plus many others).

The complexity of the filtering process is linear with the number of streams, the number of filters and the number of features used to represent the video data. Moreover, the entire process must be performed in real-time, so that we can arrive to a situation where most of the processing power is dedicated to filtering. This paper goes toward a solution of this problem, by proposing a novel approach to Audio/Video filtering that makes use of simple additional information sent together with the video. This allows to avoid the comparison of the filter with video features for many non-matching videos or video components that will not pass the filter in any case. The proposed approach requires that the measure of the similarity between the filter and the video representative is *metric*, and it is based on the use of the well known technique of *pivots*. As it will be shown in the section dedicated to the experimental evaluation of the method, it is possible to obtain – depending on the filter threshold – an efficiency improvement of more than one order of magnitude.

The paper provides a brief description of the applicative scenario in the next section. Section 3 is dedicated to the description of the technique whose performance is evaluated in Section 5. This section also contains an analysis of the system parameters to be used in order to obtain the best performance. Section 4 contains a description of a filtering algorithm when several descriptors are used

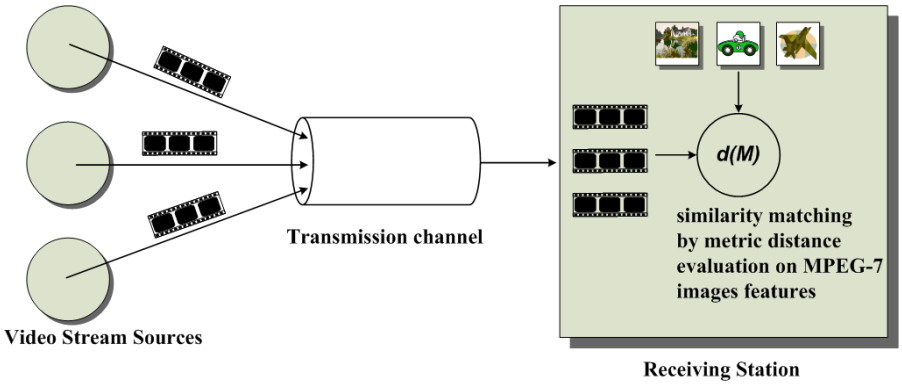

Figure 1: The scenario.

at the same time. Section 6 contains the conclusions and proposals for further research.

2 The Scenario

As described above, in our scenario we assume to have a certain number of virtual sources (for instance channels of a digital tv broadcaster), each one sending a video stream associated to an MPEG-7 stream that contains the description of the video stream. Moreover, these streams are sent through a generic transmission channel (see Figure 1) to the receiver stations (e.g., set–top–boxes, workstations, etc.). In general, any description of the video content can be considered: it can be based on a description of the entire video, e.g. based on metadata such as title, author, etc; it can be based on a description of the video shots or scenes; it can be based on a number of video representative frames. In this paper we assume that the MPEG-7 stream contains a number of *visual descriptors* (image features such as Scalable Color and Color Structure) for a subset of the frames of the video. We call these frames, *Selected Frames* (*S-Frames*). For simplicity, we will select and analyze a frame over 5 frames; this corresponds, in the PAL television, to 5 frames per second. However, the technique is not limited to use this simple method for the selection of the S-Frames: more semantically meaningful selection techniques can be adopted in real application settings, such as automatic selection of key-frames representing video shots or scenes. We call *Selection Frame Rate* (s_f) the number of S-Frames expressed in frames per second. Each frame is described by using the MPEG-7 visual descriptors, which are generated by the source station.

In this scenario a query filter is an image whose representation is obtained by using the same MPEG-7 descriptors. A comparison between the filter and each S-Frame is performed by using the metric similarity measure associated to the descriptor. This is the task of the receiver station, which has to match the queries (i.e., images submitted by the user), with the S-Frames of all the streams received. In case an S-Frame passes the filter, all successive frames until next S-Frame are shown to the user.

In order to filter the stream without any supporting index structure either on the video stream or on the queries, we should use the brute force algorithm, which requires the comparison of all S-Frames with the query filter. However, this solution overloads the receiver station, and reduces the number of channels we can filter simultaneously. In fact, let n_q, n_s, and n_d be the number of the query filters, the number of sources, and the number of MPEG-7 descriptors, respectively. The total time spent by the receiver station for the similarity evaluation is given by $T_c = n_q n_s n_d t_c$; where t_c is the average similarity computation time for the descriptors. If we have to elaborate the similarity for each S-Frame received, we have that the processor utilization of the receiver station is given by

$$U_c = T_c s_f = n_q n_s n_d t_c s_f \tag{1}$$

For instance, assuming $t_c = 10\mu sec$, $n_q = 50$, $n_d = 4$, $n_s = 100$, we obtain $T_c = 0.2sec$; with $s_f = 5frames/sec$ we obtain $U_c = 100\%$, which means that the processor is fully loaded by the filtering elaboration.

In order to improve the filtering capabilities of the receiver, two approaches can be adopted: (i) when the number of filters is very large, they can be organized at the receiver station by using an appropriate access data structure, or (ii) at the sender station, a pre-computation of the similarity between some S-Frames can reduce the number of similarity computations made at the receiver station between the query and the S-Frame. The first solution does not require any further computation on the sender station, but it involves the implementation of an index structure of the queries, to efficiently match S-Frames against several queries. This approach has been exploited in the field of textual data [13]; the extension to our application scenario can be straightforward by using an access structure for measuring the similarity between S-Frames [5, 8, 9]. Nevertheless, this solution has the drawback that is worthwhile only when the number of queries is large. With the second solution, we exploit the computational power of the sender station, by adding to the MPEG-7 stream some pre–computed similarity information, in order to increase the performance of the receiver station. Note that, the overhead for generating this pre-computed information is minimal because its computation is negligible compared with the one for the MPEG-7 stream production.

The scope of this research is to analyze the performance behavior of this second approach, which, to the best of our knowledge, has never been explored.

3 Metric Space and Pivot Filtering

The standard approach for exploring modern data repositories, such as multimedia databases, and to filter relevant data from a continuous multimedia stream, is based on the use of features extracted from complex information objects. Features are typically defined in high dimensional vector spaces or even generic metric spaces where pairs of elements can only be compared by *distance functions*. From a formal point of view, the mathematical notion of *metric space* provides a useful formalization of similarity or nearness.

The necessary starting point to implement a similarity search algorithm is to consider a *measurable distance* (*dissimilarity*), which in turn allows objects to be ranked according to their distance with respect to a given reference (target or query) object. Similarity queries are defined by a query object and a constraint on the ranked list of data objects, which is typically specified as a distance threshold or a number of required objects. In order to speed up retrieval in large collections of metric data, access methods have been developed. An excellent survey of methods for vector spaces can be found in [10], while a comprehensive list of techniques for generic metric spaces is analyzed in [7].

In this section, we first give the definitions for metric distance functions and similarity range query. We explain some terminology and the assumptions necessary for the second part of the section, where a general model of the application is presented.

3.1 Metric spaces

A convenient way to assess the similarity between two objects is to apply metric functions to decide the closeness of the objects as a distance, which can be seen as a measure of the objects *dis-similarity*. A *metric space* $\mathcal{M} = (\mathcal{D}, d)$ is defined by a domain of objects (elements, points) $\mathcal{D}$ and a total (distance) function d. For any distinct objects $x, y, z \in \mathcal{D}$, the distance must satisfy the following properties:

$d(x, x) = 0$	*reflexivity*
$d(x, y) > 0$	*strict positiveness*
$d(x, y) = d(y, x)$	*symmetry*
$d(x, y) \leq d(x, z) + d(z, y)$	*triangle inequality*

Given a data–set of metric objects $X \subset \mathcal{D}$, for a query object $q \in \mathcal{D}$, the most important similarity query is the *range query*, which retrieves all elements within distance r to q, that is the set $\{x \in X | d(q,x) \leq r\}$.

3.2 Pivot-Based Filtering

One important characteristic of distance based index structures used for data search is that, depending on specific data and a distance function, the performance can be either I/O or CPU bound and the strategy of indexes based on distances should be able to minimize both the cost components. The type of access structure that we will use for filtering requires to access a little number of stored objects (the queries, the pivots, and the current S-Frames), thus the cost is only CPU bound: the efficiency of the filtering is obtained by reducing the number of similarity measures between the query object and the target objects.

The basic idea of the pivot-based algorithms is to exploit the knowledge of a set of pre–computed distances between one object of the data-set, called pivot, and all the objects of the data-set. Using the triangle inequality, sometimes we can avoid the distance computation between the query object q and the object x of the data-set. Formally, given the pivot p, a generic object x, and a query object q, the triangle inequality corresponds to the two following statements

$$d(q,p) \leq d(x,p) + d(x,q) \quad \Rightarrow \quad d(q,p) - d(x,p) \leq d(x,q)$$

$$d(x,p) \leq d(q,p) + d(x,q) \quad \Rightarrow \quad d(x,p) - d(q,p) \leq d(x,q);$$

which can be synthetized by the following inequality

$$|d(x,p) - d(q,p)| \leq d(x,q).$$

An improvement of the efficiency can be obtained by avoiding the evaluation of the distance between the query and the object $d(x,q)$, which is on the right side of the above inequality. For this purpose we can exploit, the term on the left side, which represents a lower bound of the unknown distance $d(x,q)$. If we indicate this lower bound distance as $\underline{d}(x,q)$,i.e.

$$\underline{d}(x,q) = |d(x,p) - d(q,p)| \tag{2}$$

we have that when

$$\underline{d}(x,q) > r \quad (\text{exclusion test}), \tag{3}$$

we can exclude that $d(x,q) \leq r$, i.e. x does not belong to the result set of the query q. We refer to such a check as *exclusion test*. Here, we assume that the distances

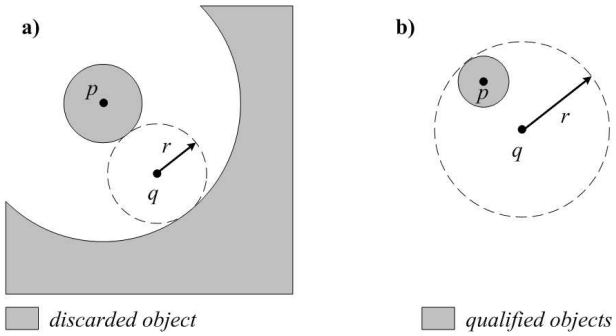

Figure 2: Example of pivots behavior.

$d(q,p)$ and $d(x,p)$ are pre–computed, thus the object x can be excluded from the result set without actually evaluating the distance $d(x,q)$. Figure 2a illustrates this principle of the pivoting technique: provided that the distance between any object and p is known, the gray area represents the region of objects x that do not belong to the query result. When this test fails, it is possible to exploit again the triangular inequality for a second check, which is

$$d(x,q) \leq d(x,p) + d(q,p).$$

In this case, we obtain an upper bound of the distance $d(x,q)$, which we designate as $\overline{d}(x,q)$, i.e.

$$\overline{d}(x,q) = d(x,p) + d(q,p). \tag{4}$$

When

$$\overline{d}(x,q) \leq r \quad (\text{inclusion test}), \tag{5}$$

we are sure that $d(x,q) \leq r$, i.e., x belongs to the result set, without the need of evaluating $d(x,q)$. We refer to such a check as *inclusion test*. Figure 2b illustrates this second idea. In this case the gray circle represents the region of object which do not need to be evaluated and belong to the result set. Eventually, if both Conditions (3) and (5) are not satisfied we have to evaluate the distance $d(x,q)$. For more details about the pivoted-based algorithms, see for example [6].

3.3 The Pivoted Stream

We can now describe in detail the Pivoted Stream method, that uses the pivot technique to improve the efficiency of video filtering. In Section 2 we mentioned that each MPEG-7 stream is elaborated at the source station and n_d visual descriptors

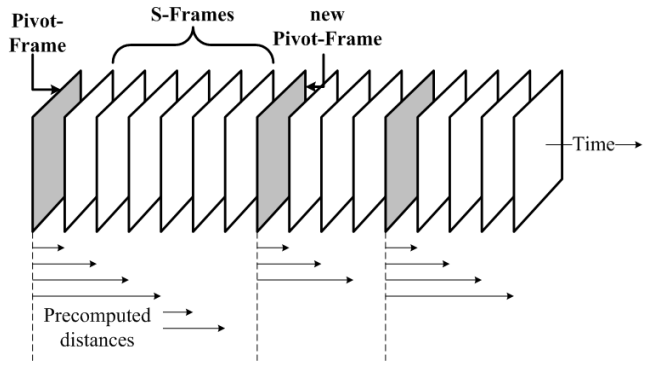

Figure 3: Illustration of the Pivoted Stream principle.

are extracted from each S-Frame, in order to represent its content. In general, a generic descriptor i is supported by the functions D_i and d_i. The former is used for extracting the feature from the image, and the latter for evaluating the distance (or dissimilarity) between two the features. For instance, given two image x and y, the distance can be evaluated by applying d_i on the corresponding features: $d_i(D_i(x), D_i(y))$. However, for the sake of simplicity we use the notation $d_i(x, y)$ for indicating the distance between the feature extracted by the descriptor D_i.

Throughout the paper, we use the symbols f and q to indicate an S-Frame (of the video stream), and the query image, respectively. Moreover, we suppose that the receiver station maintains a set of n_q queries, $q_1, \ldots, q_{n_q}$.

The principle of the Pivoted stream is to elect one of the S-Frames f as a *Pivot-Frame* p, and to pre–compute (at the source station) the distances between the current pivot frame p and the successive S-Frames f. Figure 3 illustrates this principle. At the beginning the first S-Frame of the stream is set as a Pivot-Frame p_i of the $i-th$ descriptor; for any successive S-Frame f, we evaluate the distances $d_i(p_i, f)$. These distance measures are attached to each S-Frame f together with their MPEG-7 descriptors. In this way, during the filtering phase, the receiver can exploit these pre–computed distances in order to skip some distance evaluations between the S-Frames and the queries, as explained in the previous section. Periodically, new Pivot-Frames are introduced in the stream so that the distance between the S-Frame and the pivot is sufficiently small. Note that, in general, we have n_d distinct pivots, one for each descriptor. In order to decide if an S-Frame should become the Pivot–Frame p_i of the i-th descriptor, we define n_d independent thresholds s_i; if for a given S-Frame f $d_i(p, f) > s_i$, the frame f is elected as a new Pivot-Frame p and the distances of the following S-Frames are evaluated with

```
for j = 1 to n_q
    for i = 1 to n_d
        # exclusion test:
        if d_i(q_j, f) > r_i then continue;
        else
            # inclusion test:
            if not d̄_i(q_j, f) ≤ r_i then
                # evaluate the distance:
                if d_i(q_j, f) > r_i then continue;
            end if
        end if
        f ↦ R_{i,j};
    end for
end for
```

Figure 4: Pivot Filtering Algorithm for Range Queries.

respect to the new Pivot-Frame.

It is worth observing that, having n_d pivots does not imply a significant overhead of the data transmission. Indeed, this technique only requires to send the distance measured between the S-Frame and the pivot; if this distance is zero, the S-Frame is considered as a new pivot. In general, an S-Frame can be simultaneously the pivot of many descriptors.

The next subsection describes the algorithms used by the receiving station to exploit the knowledge of the pre–computed distances $d_i(p_i, f)$, in order to implement range queries.

3.3.1 Range Queries

We assume that a search radius r_i is associated to each descriptor i. Figure 4 reports the algorithm for the similarity range queries. During a generic interval of time, the receiver station filters the source stream and produces the result sets $R_{i,j}$ (for each descriptor d_i and query q_j) formally defined as $R_{i,j} = \{\forall f \text{ received} \mid \ d_i(f, q_j) \leq r_i\}$. The basic idea of the algorithm is straightforward: first it checks the two pivot tests explained in the previous section. If both of them fail, we must evaluate directly the distance $d_i(q_j, f)$, in order to verify if it is smaller than r_i; in this case f is added to the current set $R_{i,j}$. In general, at any time, $R_{i,j}$ contains the range query result set, for the the descriptor i and query q_j. Note that, the **continue** statement force the inner loop to jump to the evaluation of the next descriptor i.

It is worth observing that, the choice of the threshold s_i is important; however, the receiver does not need to know it. If s_i is too small, many S-Frames become

Pivot-Frames while if s_i it too large we may have a strong performance degradation. An evaluation of the relationship between the filtering performance and s_i is given in the next section.

4 Combining Descriptors

In order to improve the effectiveness of the filter, it is possible to combine several visual descriptors. Indeed, the similarity between visual descriptors, is used to emulate human perception of frame similarity. Thus, the use of several visual descriptors, should improve the quality of the similarity measure: for example, by comparing two images according to their color distribution and the shape of the objects they contain, should provide more effective results that the use of a single descriptor.

Although several types of feature combination can be used, we will use the simplest, the linear combination, which provided good retrieval effectiveness in our image search engine of the MILOS system [4].

Given a query q, and an S-Frame f, we define the new combined distance d as

$$\begin{aligned} d(f,q) &= w_1 * d_1(f,q) + \ldots + w_{n_d} d_{n_d}(f,q) \\ &= \sum_{i=1}^{n_d} w_i d_i(f,q) \end{aligned} \tag{6}$$

where w_i is the weight assigned to visual descriptor i. Note that the new distance d is still metric. In fact, it is easy to see that the axioms of Section 3 are valid for the combined distance d if they are valid for each distance d_i.

The idea of using the knowledge of pre–computed distances for obtaining two bounds of the metric distance can be exploited also in case of combined descriptors. Moreover, in this case we can enhance this technique by avoiding some distance evaluation between the pivots and the query (besides the one between the query and the frame). In practice, the technique consists of calculating the bounds $\overline{d}$ and $\underline{d}$ incrementally, adding the contribution of the each descriptor one bye one, as explained below.

The lower bound $\underline{d}$ can be defined generalizing Equation (2) as follows

$$\begin{aligned} \underline{d}(f,q) &= w_1 * |d(f,p_1) - d(q,p_1)| + \ldots + w_{n_d} |d(f,p_{n_d}) - d(q,p_{n_d})| \\ &= \sum_{i=1}^{n_d} w_i |d(f,p_i) - d(q,p_i)| \end{aligned} \tag{7}$$

Similarly, the upper bound $\overline{d}$ in Equation (4) can be generalized as follows

$$\begin{aligned}\overline{d}(f,q) &= w_1 * (d(f,p_1) + d(q,p_1)) + w_{n_d}(d(f,p_{n_d}) + d(q,p_{n_d})) \\ &= \sum_{i=1}^{n_d} w_i(d(f,p_i) + d(q,p_i)). \end{aligned} \quad (8)$$

Observe that these bounds require the evaluation of the distances between the query q and the current pivot p_i. In particular, during the evaluation of the sum of the lower bound $\underline{d}$ we can exploit the "partial sum" by testing if it is greater than the radius r. If the test is positive we can avoid its complete evaluation since in this case we are sure that the query does not belong to the result set.

Figure 5 reports the filtering algorithm for combined descriptors. In order to simplify the description of the algorithm, some simple optimizations of the code – implemented in the version used for the experiments – are not included in the algorithm shown in Figure 5.

After the initialization phase, the algorithm can be seen as composed of two inner **for** loops inside the outer **for** loop, which iterates over the evaluation of the n_q queries. The first inner loop evaluates $\overline{d}$ and $\underline{d}$. If the evaluation of these bounds is not able to satisfy the current query q_j, we proceed by computing the real distance $d(f, q_j)$ (i.e., the second second inner **for** loop). However, also in this case we can proceed incrementally – we continue with a second level of refinement of the bounds $\overline{d}$ and $\underline{d}$, by substituting each term of the summation with the real distance $d_i(q_j, f)$, and exploit them for the query evaluation.

Let us now examine the algorithm in more detail. For what concerns the first inner **for** if the partial sum of $\underline{d}$ is greater than r we can skip the current query q_j (since it does not qualify) and we can process the next query j. For this purpose the **break** statement exits the inner **for** loop and the **continue** statement jumps to the next query. However, if this does not happen and we reach the end of the **for** loop, we can check if the upper bound $\overline{d}$ is smaller than r; if so, the query qualifies and we can jump to the evaluation of the next query j. In case also the upper bound fails, we must proceed to the second part of the algorithm.

Before continuing, we must explain the role of the matrix $o'_{i,j}$. Each row of the matrix maintains the order of the evaluation of the distance between the descriptors of the query q_j and the pivots. For instance, if $o'_{3,j} = 2$ we will evaluate the descriptor 2 as third, for the elaboration of the query q_j. In this way, we dynamically adapt the algorithm by attempting to evaluate first the descriptors that produce greater values of the term $w_i * |d_i(p_i, q_j) - d_i(p_i, f)|$, increasing the probability that $\underline{d} > r$ before the end of the loop. For this reason, the swap statement exchanges the order of the evaluation of the two last descriptors if the first one produces a small

for $i = 1$ **to** n_d
  for $j = 1$ **to** n_q
    $o'_{i,j} = o''_{i,j} = i$;
  end for
end for
for $j = 1$ **to** n_q
  $\overline{d} = 0$;
  $\underline{d} = 0$;
  for $m = 1$ **to** n_d
    $i = o'_{m,j}$; # take the descriptor index
    if $w_i > 0$ **then**
      $min\_curr = w_i * |d_i(p_i, q_j) - d_i(p_i, f)|$;
      $\overline{d} = \overline{d} + w_i * (d_i(p_i, q_j) + d_i(p_i, f))$;
      $\underline{d} = \underline{d} + min\_curr$;
      if $min\_curr > last\_min$ **and** $m > 1$ **then**
        $swap(o'_{m,j}, o'_{m-1,j})$;
      else
        $last\_min = min\_curr$;
      end if
      if $\underline{d} > r$ **then break**;
    end if
  end for
  if $\underline{d} > r$ **then continue**;
  if $\overline{d} \leq r$ **then**
    $f \mapsto R_{i,j}$;
    continue;
  end if
  for $m = 1$ **to** n_d
    $i = o''_{m,j}$; # take the descriptor index
    if $w_i > 0$ **then**
      $diff\_curr = w_i * (d_i(q_j, f) - |d_i(p_i, q_j)$
        $-d_i(p_i, f)|)$;
      $\overline{d} = \overline{d} + w_i * (d_i(q_j, f) - d_i(p_i, q_j) - d_i(p_i, f))$;
      $\underline{d} = \underline{d} + diff\_curr$;
      if $diff\_curr > last\_diff$ **and** $m > 1$ **then**
        $swap(o''_{m,j}, o''_{m-1,j})$;
      else
        $last\_diff = diff\_curr$;
      end if
      if $\underline{d} > r$ **or** $\overline{d} \leq r$ **then break**;
    end if
  end for
  if $\overline{d} \leq r$ **then** $f \mapsto R_{i,j}$;
end for

Figure 5: Pivot Filtering Algorithm for Combined Descriptors.

contribution. After an initial transient period, the rows $o'_{i,j}$ will contain the best order of evaluation for the descriptors of all queries, giving higher chances that the lower bound $\underline{d}$ succeeds.

The second part of the algorithm takes advantage of the computation already performed; it proceeds by substituting the terms of the two summations ($\overline{d}$ and $\underline{d}$) with $w_i d_i(q_j, f)$, the weighted distances between the current query and the frame f. This is obtained by iterating again over the descriptors. During the iteration we will have more and more precise values of the bounds, which at the end will converge to the real combined distance, i.e. $\overline{d} \equiv \underline{d} \equiv d(q_j, f)$. The hope is to terminate the algorithm before the complete evaluation of the (second **break** statement). Again, we try to keep the best order in the evaluation of the descriptors by using a second set of vectors $o''(j)$. However, in this case we must evaluate the descriptors which produce higher values of the difference between the real distance $d_i(q_j, f)$ and its surrogate $|d_i(p_i, q_j) - d_i(p_i, f)|$.

5 Performance Evaluation

The experimental evaluation has been performed by using real broadcasted programs: in particular, we used 90 minutes of the Rai Uno TV channel (the main Italian public television channel) taken from 7:45pm to 9:15pm. Queries are given by frames taken from the stream itself (this allows us to test the worst case performance, because the probability that the pivot test fails is higher for queries closer to the S-Frames). In the experiments we used four different queries that represent specific aspects of the video stream, namely (i) the beginning of advertising, (ii) the beginning of news, (iii) a frame taken from a soccer play, and (iv) a frame taken from a quiz show.

Three MPEG-7 visual descriptors have been tested – Scalable Color (SC), Color Structure (CS), and Edge Histogram (EH) – by using the MPEG-7 XM Reference Software for their extraction [2]. The similarity computation was based on the distance measures proposed in the XM Reference software, of the MPEG group. The descriptors are vectors and their distances are metric. The following parameters have been used for each descriptor: for SC we used 64 coefficients with 0 bitplanes discarded; for CS, we used 64 coefficients; no parameter was required by the EH descriptor (it uses 80 coefficients).

As already described, the system is able to filter the S-Frames by using range queries, i.e. all S-Frames whose distance from the query filter is lower than a certain value r are selected. The brute force algorithm requires the comparison of all S-Frames with the query filter. The use of the pivots can reduce the total number of distance computations. This number is equal to the number of similarity mea-

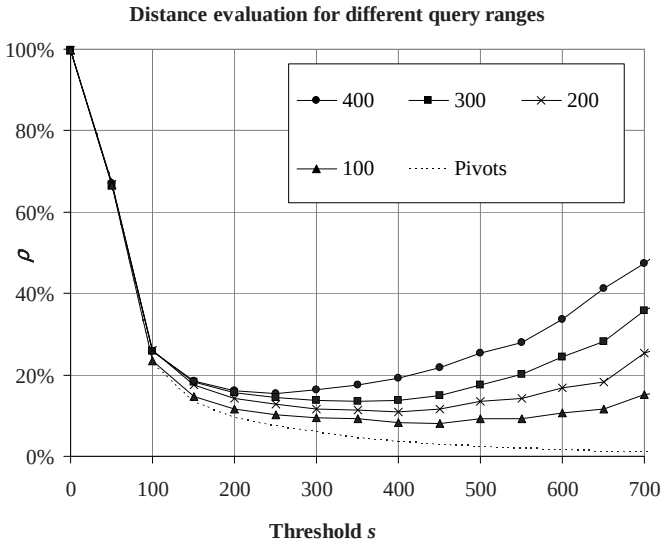

Figure 6: Distance computation cost ρ vs the threshold s, for the quiz query. The Scalable Color is used and graphs are shown at different query ranges (100, 200, 300, and 400), as well as when only the pivots are used.

sures computed between the queries and the pivots plus the number of similarity measures between the queries and the S-Frames, in case the pivot test fails. All experiments will measure the percentage of distance similarity computations needed when the pivot method is used, with respect to the number of distance computations needed when the brute force algorithm is adopted. Let T be an arbitrary period of time, we denote this distance computation cost as

$$\rho_{i,j}(T) = \frac{n_{i,j}}{T\ s_f}.$$

where $n_{i,j}$ is the number of distance computations during the time T for the descriptor d_i and a given query q_j ($0 < \rho_{i,j}(T) \leq 1$). The objective is to obtain a value of ρ as small as possible ($\rightarrow 0$), it we want to minimize the number of distance computations.

In general, the total number of distance similarities to be evaluated depends on (i) the query range r_i and (ii) the threshold that the server uses to choose the Pivot-Frames s_i. The evaluation reported in the following analyzes the improvement of performance ρ that can be obtained by using the pivot method by varying these two parameters.

5.1 Sensitivity of the performance with respect to the threshold

In Figure 6, ρ is reported as a function of the threshold s_i for different query ranges. The Scalable Color (SC) descriptor is used as image feature. The figure shows, in dotted line, the percentage of distance computations due only to the pivot evaluations. This curve represents a lower bound of computation for the Pivoted Stream algorithm, since the distance between the pivots and the query must always be calculated. Obviously, when the value of the threshold increases, this contribution decreases, since the frequency of the pivots diminishes.

The other four graphics in solid line represent ρ for distinct range queries as function of the threshold. As we can see, in general, for a range query with $r > 0$, the number of distance evaluations decreases with the threshold until a certain point, after which it grows again. This behavior can be explained by observing that, when the threshold is small, the number of distances is dominated by the distance evaluation $d(p, q)$ between the pivot and the query (which follows the dotted line). As the threshold grows, the Pivoted Stream algorithm works progressively better. However, after a certain point (related to the search radius r), when the threshold becomes too high, the number of pivot faults increases, and the total number of distance computations grows. In other words, for a given search radius there is a threshold for which the number of distance evaluations is minimum.

5.2 Range queries

In Figure 7 we reported ρ for the four queries, as a function of the query range. Due to the lack of space we show only the performance of the SC descriptor. In the figures, we reported three different curves: (i) the *best threshold*, which uses for each value of the query range the best threshold value; (ii) a curve obtained for a fixed value of threshold equal to 250; (iii) the percentage of qualifying S-Frames. The first curve provides an indication of the variability of the performance with respect to the choice of the threshold. The most significant and frequently used range values are the intermediate ones, between 100 and 400. For these values, the differences between curves (i) and (ii) are not significant, which means that the optimal choice of the threshold value is not critical. In general, the performance improvements are higher for lower query ranges. This does not happen for some queries (Soccer query), where for range queries higher than a certain value, performance improves as the query range increases. This is due to the inclusion test (Equation (5)). Note that this advantage cannot be exploited if we need to rank the result set, since the inclusive check allows to avoid some distance computations $d(x, q)$, even if the object x qualifies.

In Figures 8 we report a synthetic view of performance improvement for the

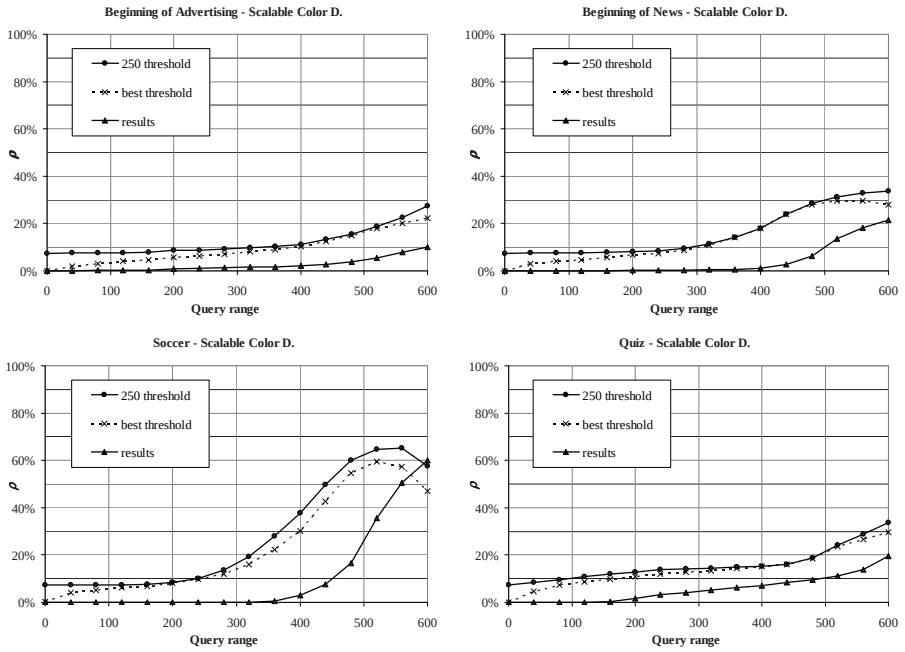

Figure 7: Distance computation cost ρ for different queries vs the query range. The Scalable Color descriptor is used.

four test queries, by using all the three feature descriptors for range queries. The thresholds used are 250 for SC, 3 for CS and 9.60 for Edge Histogram. Using these thresholds we obtained that 7% of the S-Frames become pivots. We used a radius 200 for SC, 3 for CS and 5 for EH except for the beginning of advertising query for which we used 50, 0.25 and 0.3. Indeed, for this specific query we almost needed an exact match, while for the other queries we only required the selection of similar images. This experimental results show that for all different queries and for all feature descriptors the Pivoted Stream method gives significant performance improvements, with the EH descriptor that gives the worst results for all four queries. It is worth mentioning that the radii of the range queries have been chosen in order to have a reasonable number of results, which are between 10 and 50 for all descriptors. Only for the Quiz Query the SC and CS descriptors give many more matches (several hundreds) due to the high color similarity between different S-Frames of the quiz show.

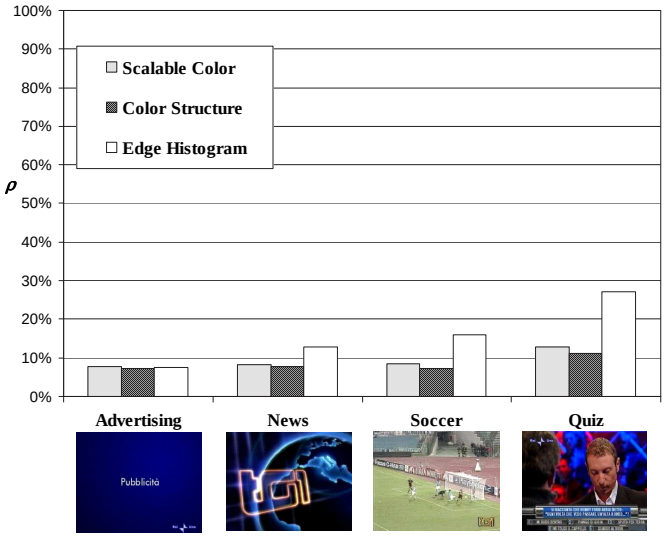

Figure 8: Distance computation cost ρ for different range queries and three different image descriptors.

5.3 Linear combination

The last set of experiments studies the performance of the system when we linearly combine the three visual descriptors (SC, CS, and EH) as explained in Section 4. For convenience, we have chosen the weights w_i to be the inverse of the radii of the range queries experiments. Consequently, assuming $r = 3$ for a combined range query, we have that the number of results are same order of magnitude of the single descriptor ones. In particular in our experiments we have chosen r in way that if a frame f passes all the three single descriptor filters, it passes also the combined descriptor filter. On the contrary, if the frame f passed the combined descriptor filter, it passes at least one of the single descriptor filter.

Figure 9 shows the global result of the query with combined descriptors for range queries. In this case ρ represents the distance computation cost averaged over all the descriptors. In all the experiments, the advantage of the pivots is evident: ρ ranges from 2.48% of Advertising to 7.23% of Quiz. The number of results for the range queries is 21 for Advertising, 31 for News, 17 for Soccer, and 163 for Quiz.

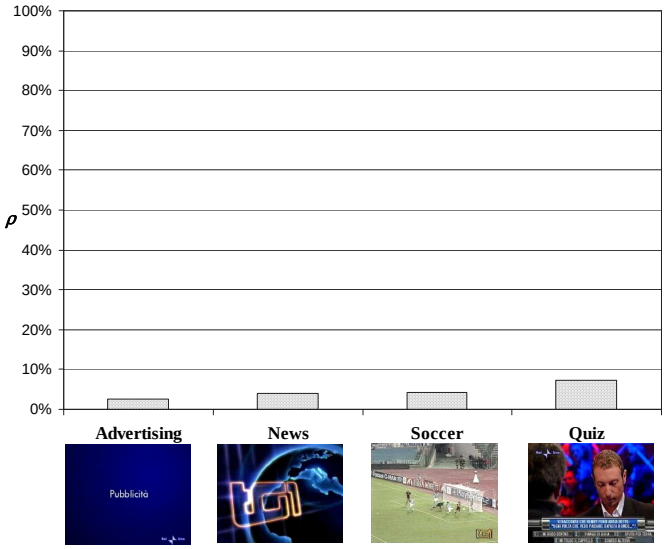

Figure 9: Distance computation cost ρ for various range queries using combined descriptors.

6 Conclusion

In this paper we presented a preliminary research study about a technique, the Pivoted Stream, which can also be used to improve the performance of filtering video streams. The experimental evaluation of the method provides very encouraging results, with performance improvements up to more than one order of magnitude with respect to the brute force algorithm which is usually adopted. We also proved the efficiency of the pivots for the more interesting and useful case of combined descriptors

There are many possible applications where the Pivoted stream technique could be useful: for instance, for statistics purpose, it could be necessary to compute how many times an advertising is transmitted; the method can be used to select the start time of a television programme, in order to begin its recording. Another possible use is in the field of property protection of videos or images distributed by news agencies like the Italian ANSA. These agencies have special agreements of use of the material sold to TV broadcaster. A filtering application as the one proposed in the paper could help the news agencies to discover illegal use of this material. Besides the television environment there can be other different applications such as the earth satellite surveillance (SAR), the police surveillance, etc. Moreover,

with the finalization of the MPEG-21 standard [1] it will be easy to introduce the pivoted stream in a real broadcast system.

We think that the proposed approach opens many possible future lines of investigation. In particular, we intend to explore the use of other features, such as the text (OCR captions, the subtitles, etc.) and audio descriptors.

Further performance improvements can be studied if the number of query filters is large: in this case it would possible to organize the query filters by using a specific access structure for metric spaces.

References

[1] Mpeg mds group, mpeg-21 multimedia framework, part 7: Digital item adaptation (final committee draft), iso/mpeg n5845, july 2003. http://www.chiariglione.org/mpeg/working_documents/mpeg-21/dia/dia_fcd.zip.

[2] Mpeg-7 reference software, October 2002. http://www.lis.e-technik.tu-muenchen.de/ research/bv/topics/mmdb/e_mpeg7.html.

[3] Mpeg requirements group, mpeg-7 overview, 2003. Doc. ISO/IEC JTC1/SC29/WG11N5525.

[4] G. Amato, C. Gennaro, P. Savino, and F. Rabitti. Milos: a multimedia content management system for digital library applications. In *Proceedings of the 8th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2004)*, volume 3232 of *Lecture Notes in Computer Science*, pages 14–25. Springer, September 2004.

[5] T. Bozkaya and Z. M. Özsoyoglu. Indexing large metric spaces for similarity search querie. *ACM TODS*, 24(3):361–404, 1999.

[6] E. Chávez, J. L. Marroquín, and G. Navarro. Fixed queries array: A fast and economical data structure for proximity searching. *Multimedia Tools and Applications*, 14(2):113–135, 2001.

[7] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. Technical Report TR/DCC-99-3, Dept. of Computer Science, Univ. of Chile, 1999. ftp://ftp.dcc.uchile.cl/pub/users/-gnavarro/survmetric.ps.gz, to appear in ACM Computing Surveys, 2001.

[8] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proc. of VLDB'97*, pages 426–435. Morgan Kaufmann, August 1997.

[9] V. Dohnal, C. Gennaro, P. Savino, and P. Zezula. D-index: Distance searching index for metric data sets. *Multimedia Tools and Applications*, 21(1):9–13, 2003.

[10] V. Gaede and O. Gunther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[11] S. Krishnamachari, A. Yamada, M. Abdel-Mottaleb, and E. Kasutani. Multimedia content filtering, browsing, and matching using mpeg-7 compact color descriptors. In *Proceedings of 4th Advances in Visual Information Systems, VISUAL 2000, Lyon, France*, volume 1929 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2000.

[12] B. S. Manjunath, P. Salembier, and T. Sikora, editors. *Introduction to MPEG-7: Multimedia Content Description Interface*. Wiley Uk, 2002.

[13] T. W. Yan and H. Garcia-Molina. Index structures for information filtering under the vector space model. pages 337–347. IEEE Computer Society, 1994.

[14] D. Zhong, R. Kumar, and S.-F. Chang. Real-time personalized sports video filtering and summarization. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 623–625. ACM Press, 2001.