Optimization Algorithms for the Selection of Key Frame Sequences of Variable Length

Tiecheng Liu and John R. Kender

Department of Computer Science
Columbia University
New York, NY 10027
{tliu,jrk}@cs.columbia.edu

Abstract. This paper presents a novel optimization-based approach for video key frame selection. We define key frames to be a temporally ordered subsequence of the original video sequence, and the optimal k key frames are the subsequence of length k that optimizes an energy function we define on all subsequences. These optimal key subsequences form a hierarchy, with one such subsequence for every k less than the length of the video n, and this hierarchy can be retrieved all at once using a dynamic programming process with polynomial ($O(n^3)$) computation time. To further reduce computation, an approximate solution based on a greedy algorithm can compute the key frame hierarchy in $O(n \cdot log(n))$. We also present a hybrid method, which flexibly captures the virtues of both approaches. Our empirical comparisons between the optimal and greedy solutions indicate their results are very close. We show that the greedy algorithm is more appropriate for video streaming and network applications where compression ratios may change dynamically, and provide a method to compute the appropriate times to advance through key frames during video playback of the compressed stream. Additionally, we exploit the results of the greedy algorithm to devise an interactive video content browser. To quantify our algorithms' effectiveness, we propose a new evaluation measure, called "well-distributed" key frames. Our experimental results on several videos show that both the optimal and the greedy algorithms outperform several popular existing algorithms in terms of summarization quality, computational time, and guaranteed convergence.

1 Introduction

Key frames are usually defined to be an unordered subset of video frames that represent the visual content of a video. They are of great importance in video indexing, summarization, content retrieval, and browsing [1,4]. There are at least three main categories of existing video key frame selection approaches [9,11]. In the first, video key frame selection is based on video segmentation, and is often used on highly edited commercial videos [14]. But because the results of this approach depends on the accuracy of video segmentation [2], it is not quite suitable for semi-edited (e.g., instructional) videos, unedited (e.g., home)

A. Heyden et al. (Eds.): ECCV 2002, LNCS 2353, pp. 403–417, 2002.

videos, or extended single-shot (e.g., surveillance) videos. The second approach uses clustering techniques based on a definition of "far enough" frames [7,15, 16,17]. But an inherent problem in this approach is the choosing of appropriate thresholds. Although adaptive clustering methods can manipulate the threshold to produce a pre-designed number of key frames, this iterative searching process makes these methods computational expensive. The third approach [3] converts a key frame selection problem to a problem of searching for the minimum cover of a set of key frames, based on the definition of a semi-Hausdorff distance function. But this search can be shown to be NP-hard, and the $O(n^2)$ greedy algorithm approximation to it is computationally expensive. Additionally, in both this and the previous approaches, the frames are chosen without regard to their temporal order, although such temporal relations may be very important in video summarization, streaming and compression.

1.1 Two Levels of Frame Sampling

We note that prior research on key frames does not distinguish much between two significant levels of frame sampling, each with different applications. Sparse sampling, the selecting of about one frame per shot or per scene, is usually used in video content summarization and indexing. However, dense sampling, which chooses much more than one frame per shot or per scene, is more useful for video streaming, particularly in network environments where frame-dropping decisions are made in real time according to dynamic changes of network bandwidths or user requirements. Most previous work [1,6,9,10,11] appears to concentrate on sparse video frame sampling. Although some other work, for example the minimum-cover method [3], can be applied to dense sampling, generally their complexity make them unsuitable in use for extended or real-time videos.

Thus we note that for a video key frame selection algorithm, the following features are desirable: 1. A unified approach for both levels of key frame selection, with application for both video summarization and streaming. 2. The results of key frame selection should also be easily applied to video indexing and interactive content retrieval applications [10]. 3. For tractability with extended videos, the computation complexity should be $O(n)$ or $O(n \cdot log(n))$ in terms of the number of video frames.

1.2 Overview of Our Approach

In this paper, we provide an optimization approach for video key frame selection. We define an energy function on the selected key frames. Considering key frames as a temporally ordered subsequence of the original video sequence, the key frame selection problem can then be cast as a global optimization problem, solvable by dynamic programming. In this paper, the sum of all distances of temporally adjacent key frames is used as the energy criterion; we show that this maximizes the cover of the visual content of a video, reduces sampling redundancy, and avoids the selection of troublesome frames such as those within dissolves. If a video is considered as a weighted directed acylic graph in which every frame is

represented by a node, the weight of an edge is the measured visual distance between two frames, and the direction is strictly limited to the temporal order of frames, then to select k optimal key frames is to find the longest path with k nodes in the graph.

Based on the criterion defined above, we provide an optimal solution for key frame selection to compute the entire key frame hierarchy, for all k, $1 \leq k < n$, based on dynamic programming with $O(n^3)$ computation time. To further reduce computation complexity, we present a greedy algorithm based on a level-to-level, decreasing k, optimization process with almost linear computation complexity. We present experimental results of comparisons that indicate the performance of the greedy algorithm is quite similar to that of optimal algorithm. We show how the results of our key frame selection via the greedy algorithm can be used in video indexing and interactive content retrieval. Extensive experiments on key frame selection show that either of our approaches, whether optimal or greedy, retrieve better key frames than existing key frame selection algorithms.

2 Key Frame Selection as an Optimization Problem

First, we introduce some terms for video key frame selection. For a n frame video $f_1, f_2, \cdots, f_n$, let F be the sequence of all frames $F = \{f_1, f_2, \cdots, f_n\}$. There is a distance $\| \cdot \|$ defined on F. This distance can be any user-defined metric; in this paper, we use the histogram difference based on the L^1 norm.

Definitions. $F_k^i = \{f_{i_1}, f_{i_2}, \cdots, f_{i_k}\}$ is a k key frame selection of F if $F_k^i \subset F$ and $1 \leq i_1 < i_2 < \cdots < i_k \leq n$, where i represents a selection method, $F_k^i \neq F_k^j$ if $i \neq j$. In other words, F_k^i is a k key frame selection of F if F_k^i is a strict temporal subsequence of F with k elements.

We introduce an energy function $f(\alpha, \beta)$, where α is a subsequence (key frame selection) of the longer sequence β. Assume F_k^p is a k key frame selection of F. F_k^p is an optimal k key frame selection of F if it maximizes the energy function for all k key frame selections:

$$f(F_k^p, F) = \max_i \{f(F_k^i, F)\}$$

Let $S_k^* = \max_i \{f(F_k^i, F)\}$ be the optimal value of the energy function of all k key frame selections, F_k^p is an optimal key frame sequence of length k if and only if $f(F_k^p, F) = S_k^*$.

Using the definitions above, the video key frame selection problem has now been cast as a subsequence optimization problem. Further, in this paper, we define the energy function as:

$$f(F_k^i, F) = \sum_{p=1}^{k-1} \| f_{i_p} - f_{i_{p+1}} \|$$

This definition explicitly acknowledges the great perceptual impact that occurs between temporally adjacent key frames, and deliberately rewards the selection of those key frames which are maximally distant from their immediate temporal predecessors.

3 Optimal Solution Based on Dynamic Programming

3.1 Optimal Algorithm

In searching for the optimal k key frames, one straightforward solution is to investigate all possible k subsequences of the original video. The number of all such subsequences is the binomial coefficient $C_n^{\ k}$, which is not polynomial in terms of n. However, we can reduce the computation complexity to polynomial time by using a dynamic programming approach. The k key frame selection problem can be converted to a k-step decision problem solvable by dynamic programming, by exploiting an "optimal substructure" [5] existing within the key frame selection.

Optimal substructure. We show that an optimal substructure exists, using a proof by contradiction. Let $F^i_{k,p} = \{f_{i_1}, f_{i_2}, \cdots, f_{i_{k-1}}, f_p\}$ represent k key frames selected using selection indices i under the condition of last key frame being f_p; it is obvious $p \geq k$ for any $F^i_{k,p}$. $F^i_{k,p}$ is called a conditional k key frame selection, and $S^i_{k,p} = f(F^i_{k,p}, F)$. Let $S^*_{k,p}$ be the optimal energy value of k key frames with that last key frame f_p,

$$S^*_{k,p} = \max_i \{f(F^i_{k,p}, F)\}$$

$F^*_{k,p}$ is an optimal k key frame selection with the last key frame f_p when $f(F^*_{k,p}, F) = S^*_{k,p}$.

Now assume that $F^j_{k+1,p}$ is an optimal $k+1$ key frame selection with the last frame f_p. Then F^j_{k,j_k} must also be an optimal k key frame selection with last frame f_{j_k}, since otherwise there must exist another F^t_{k,j_k} with $S^t_{k,j_k} > S^j_{k,j_k}$. Let $F^t_{k+1,p} = \{f_{t_1}, f_{t_2}, \cdots, f_{t_{k-1}}, f_{j_k}, f_p\}$. $F^t_{k+1,p}$ has energy value

$$S^t_{k+1,p} = \parallel f_p - f_{j_k} \parallel + S^t_{k,j_k} > \parallel f_p - f_{j_k} \parallel + S^j_{k,p} = S^j_{k+1,p}$$

But this contradicts with the assumption that $F^j_{k+1,p}$ is an optimal $k+1$ key frame selection with last key frame f_p. So the optimal key frame selection problem has an optimal substructure: any subsequence of the optimal key frame selection must also be an optimal key frame selection. Thus we have

$$S^*_{k+1,p} = \max_{k \leq m < p} \{\parallel f_p - f_m \parallel + S^*_{k,m}\}$$

Dynamic programming. From the equation above, we propose a dynamic programming approach for optimal key frame selection. According to our definition of the energy function, $S^i_{1,m} = 0$ for any m and i. To select k optimal key frames from the original video, as shown in Figure 1, the first key frame can be selected from $\{f_1, f_2, \cdots, f_{n-k+1}\}$, and $S^*_{1,m} = 0$ for $1 \leq m \leq n-k+1$, the second key frame can be selected from the sequence $\{f_2, f_3, \cdots, f_{n-k+2}\}$, etc. We update the optimal values for the second step (or "level") in Figure 1:

$$S^*_{2,p} = \max_{1 \leq m < p} \{\parallel f_p - f_m \parallel + S^*_{1,m}\}$$

where $2 \leq p \leq n-k+2$.

So after every step, we calculate the optimal values of the conditional key frame selection, in the following way. Assume we have already calculated the qth level and derived optimal values $S^*_{q,m}$ for $q \leq m \leq n-k+q$. For the next step, the $q+1$-th key frame can be chosen from $\{f_{q+1}, f_{q+2}, \cdots, f_{n-k+q+1}\}$. We compute all conditional values for level $q+1$:

$$S^*_{q+1,p} = \max_{q \leq m < p} \{\parallel f_p - f_m \parallel + S^*_{q,m}\}$$

where $q+1 \leq p \leq n-k+q+1$. Finally, we get the optimal values for the k-th step $S^*_{k,p}$, $k \leq p \leq n$. Then the optimal value of energy function is

$$S^*_k = \max_{k \leq p \leq n} S^*_{k,p}$$

Suppose the last key frame is the frame f_t, $S^*_{k,t} = S^*_k$. Starting from the last key frame, we can follow back-pointers to the pre-stored prior optimal subsequences, and the optimal k key frames can be retrieved.

As shown in the dynamic programming process above, to compute k optimal key frames, an array of size of $k(n-k+1)$ is needed to store the optimal paths of subsequences for final retrieval, and another $n-k+1$ sized array is used to store the optimal values. Therefore, memory usage is $O(k \cdot (n-k))$. As for time, the frame distance computation is $1+2+\cdots+(n-k+1)$ for every step except the first step, and there are totally $k-1$ steps to compute, so the total computation is $O(k(n-k)^2)$. The entire key frame hierarchy of all such k can be computed in $O(n^3)$ time and $O(n^2)$ space using a similar process as shown in Figure 1.

3.2 Temporal Boundaries between Key Frames

The selected key frames can be used as a compressed version of the original video. When playing the compressed video, the appropriate time to advance between key frames should be properly calculated to ensure good quality of video playback. For example, suppose the two key frames $\{f_1, f_4\}$ are selected from a video $f_1 - f_2 - f_3 - f_4 - f_5$. When playing the compressed version of the video, there is a problem of when to display video frame f_4. The straightforward solution of displaying a key frame at its corresponding temporal position in the original video often incurs "content lag": if frame f_3 is closer in content

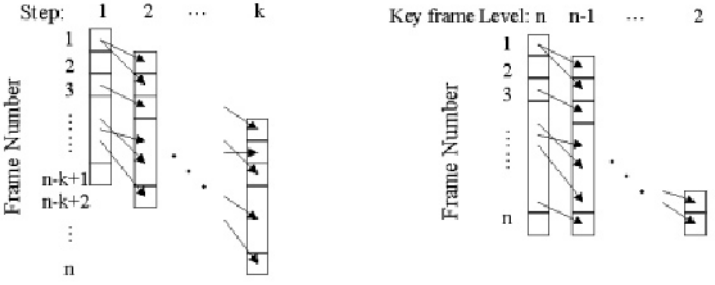

Fig. 1. Optimization methods based on dynamic programming. The left figure shows the process of select key frame sequence of length k, the right one shows the process of generating a key frame hierarchy.

to frame f_4, rather than playing $f_1 - f_1 - f_1 - f_4 - f_4$, the video sequence of $f_1 - f_1 - f_4 - f_4 - f_4$ would be more reasonable.

To address "content lag", we use a criterion of minimum sampling error to determine the most appropriate times to advance between key frames. Suppose the original video is $\{f_1, f_2, \cdots, f_n\}$ and the selected k key frames are $F_k^i = \{f_{i_1}, f_{i_2}, \cdots, f_{i_k}\}$. The time to start displaying key frame f_{i_p} should be between i_{p-1} and i_p. We choose the start time m such that m minimizes sampling error:

$$err(i_{p-1}, m, i_p) = \min_{i_{p-1}<t<i_p} \{err(i_{p-1}, t, i_p)\}$$

Therefore, $err(i_{p-1}, t, i_p)$ is defined as accumulated sampling error for displaying f_{i_p} at t between i_{p-1} and i_p:

$$err(i_{p-1}, t, i_p) = \sum_{j=i_{p-1}+1}^{t-1} \parallel f_{i_{p-1}} - f_j \parallel + \sum_{j=t}^{i_p-1} \parallel f_j - f_{i_p} \parallel$$

This computation can be done in linear time, as follows. First we calculate

$$err(i_{p-1}, i_{p-1}+1, i_p) = \sum_{m=i_{p-1}+1}^{i_p-1} \parallel f_m - f_{i_p} \parallel$$

Then we get $err(i_{p-1}, t, i_p)$ of all $i_{p-1} < t < i_p$ by an iterative process. After $err(i_{p-1}, t, i_p)$ is known, $err(i_{p-1}, t+1, i_p)$ can be derived as

$$err(i_{p-1}, t+1, i_p) = err(i_{p-1}, t, i_p) + \parallel f_{i_{p-1}} - f_{i_{p-1}+t} \parallel - \parallel f_{i_p} - f_{i_{p-1}+t} \parallel$$

The computation of all $err(i_{p-1}, t, i_p)$ $(i_{p-1} < t < i_p)$ takes $O(i_p - i_{p-1})$ time. Thus the complexity for computing all appropriate starting times for key frame playback is $O(n)$.

4 Greedy Algorithm

4.1 Definition

As before, the optimal k key frame selection is defined to be the k frames from F that optimize the energy function $f(F_k^i, F)$. We now introduce a fast approximation to the dynamic programming approach. In contrast to the prior method, this proceeds in order of decreasing k.

Suppose we have already selected k key frames F'_k. To select $k-1$ key frames, instead of directly choosing them from original frame sequence F, we select the greedy $k-1$ key frames F'_{k-1} from F'_k to optimize the energy function. Selecting $k-1$ frames from F'_k means choosing one frame in F'_k to drop. Since there are k such choices, we have

$$f(F'_{k-1}, F'_k) = \max_i f(F^i_{k-1}, F'_k) = \max_{1 \le p \le k} f(F'_k \setminus \{f_{i_p}\}, F'_k)$$

Starting from the original n frames (the "nth level"), this level-to-level greedy algorithm retains for the next level those $n-1$ key frames that maximize $f(F^i_{n-1}, F)$. And in general, if it has determined a greedy k-level key frame selection F'_k, the greedy $k-1$-level key frames selection F'_{k-1} is determined solely from F'_k. This process continues until it reaches a user-designated level of key frames. Along the way, we can maintain a list of the frames ranked by frame "significance", which records the order in which they were discarded from level to level. The key frames left in the final level are considered to be the most significant frames, and the frame dropped from the n to $n-1$ level is considered as the least significant frame, as shown in Figure 5.

This greedy algorithm has three advantages. First, its computation complexity is $O(n \cdot log(n))$, which is much better than the $O(n^3)$ complexity of the dynamic programming solution. Secondly, it provides a hierarchy of key frames with a shorter "edit distance" between levels, as the key frames in a further level is a strict subset of the key frames in any prior level, which makes it more appropriate for interactive content retrieval and searching. Thirdly, the results of the greedy algorithm can easily accomodate the change of network bandwidths in streaming, as we will show in the section 5.

4.2 Computation of Greedy Algorithm

According to our definition of greedy key frames, the greedy algorithm chooses $k-1$ greedy key frame from k already selected key frames,

$$S'_{k-1} = \max_{1 \le p \le k} S^p_{k-1} = \max_{1 \le p \le k} f(F'_k \setminus \{f_{i_p}\}, F'_k)$$

Using the same definition of the energy function, for greedy key frames, we have

$$S^p_{k-1} = S'_k - V_{i_p}, 1 \le p \le k.$$

where V_{i_p} is called the "compensation value", a measure of change of total subsequence energy:

$$V_{i_p} = \begin{cases} \| f_{i_1} - f_{i_2} \|, & p = 1 \\ \| f_{i_k} - f_{i_{k-1}} \|, & p = k \\ \| f_{i_{p-1}} - f_{i_p} \| + \| f_{i_p} - f_{i_{p+1}} \| - \| f_{i_{p-1}} - f_{i_{p+1}} \|, & 1 < p < k \end{cases}$$

Every frame f_{i_p} corresponds to a compensation value V_{i_p}, which records the change of energy value if the frame is discarded. We maximize S^p_{k-1} by dropping the frame in $\{f_{i_1}, f_{i_2}, \cdots, f_{i_k}\}$ which has the minimum compensation value. As a consequence, monotonicity of energy values is maintained in the greedy key frame hierarchy.

In more detail, the bottom-up level-to-level optimization process starts from original n frames. First, we compute the compensation value for every frame. Then we find the frame with minimum compensation value, drop that frame and get the $n-1$ greedy key frames. Let

$$S'_n = \sum_{t=1}^{n-1} \| f_t - f_{t+1} \|$$

We update the optimal energy value for $n-1$ key frames as

$$S'_{n-1} = S'_n - \min_{1 \le p \le n} \{V_p\}$$

At the same time, we correspondingly update the compensation values of the frames adjacent to the discarded frame f_p. For example, suppose we have k greedy key frames $F'_k = \{f_{i_1}, f_{i_2}, \cdots, f_{i_k}\}$, and that the minimum of compensation value is $V_{i_p} = \min_{1 \le t \le k}\{V_{i_t}\}$. Then the sequence of $k-1$ greedy key frames is $F'_{k-1} = F'_k \setminus \{f_{i_p}\}$, and the energy value of the $k-1$ greedy key frames is $S'_{k-1} = S'_k - V_{i_p}$. Finally, we delete frame f_{i_p} and update the compensation values of the key frames adjacent to f_{i_p}. Because in every step there is a "search-min" operation, we can use a binary heap, where every node in the heap records the frame number and its corresponding compensation value.

Computation complexity. Memory usage is $O(n)$. As for time, the initialization step uses $O(n)$ distance computation operations to calculate all the compensation values, and in every step we update the compensation value for a constant number of key frames; the total distance computation is $O(n)$. It is not hard to show that the arithmetic and comparison operations are $O(n \cdot log(n))$ based on the performance of the heap, and for very long videos this would dominate. However, the frame distance computation is usually much more expensive than arithmetic and comparison operations, so for a video of reasonable size the frame distance computation, which is linear, is the critical factor.

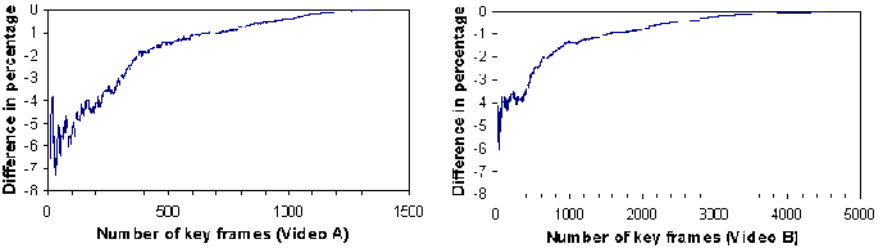

Fig. 2. The difference of energy values between optimal solution and greedy solution for two video sequences. Video A is a travelog video of 1453 frames, video B is one full scene of 4541 frames extracted from the sitcom video "Friends".

4.3 Comparison between Optimal and Greedy Algorithms

Empirical comparisons between optimal and greedy algorithms on two videos, as shown in Figure 2, show very close energy values for each level of key frames extracted by these two algorithms. We notice that for not too high of a compression ratio, the difference of energy values is very small (usually within 5%).

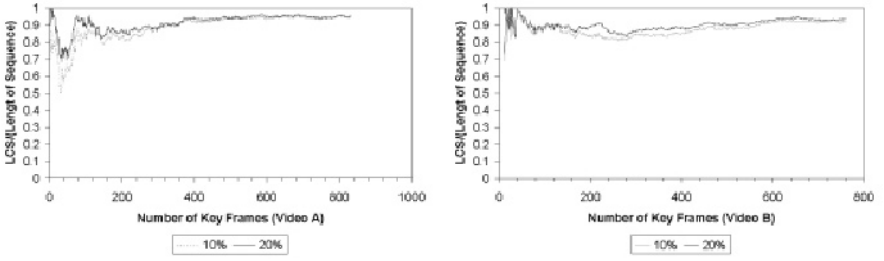

Fig. 3. The ratio of length of longest common subsequence to that of key frame sequence under different "tolerance percentages" for two videos.

We also use another criterion to compare these results. Considering the optimal key frames and greedy key frames as two sequences, the longest common subsequence (LCS) of these two sequences may indicate how much these two sequences are the same. But for video sequences, a comparison based on strict LCS may be too restrictive. We use a modified LCS computation method, which applies a "tolerance distance" in computing LCS. That is, if two frames from different video sequences have a distance less than the designated tolerance distance, they are considered to be identical. According to this definition, we select the distance tolerance as a fixed percentage of the average adjacent key frame distance, $d_{tolerance} = \epsilon \cdot S_k^*/(k-1)$, where ϵ is called the "tolerance percentage". Figure 3 shows the the amount of LCS shared between the key frame sequences

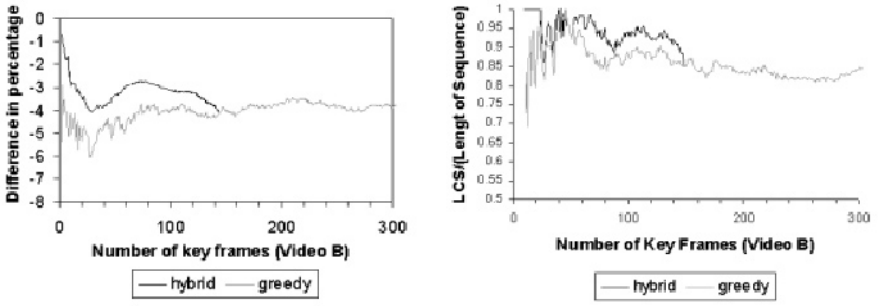

Fig. 4. Compare key frames of hybrid model and greedy model to those of optimal model in terms of energy value and longest common subsequence. Cutoff value between optimal and greedy methods is apparently at $k = 150$ for both graphs.

derived from both methods, under different tolerance percentages, and normalized as a ratio of LCS length to optimal sequence length (perfect capture by the greedy method gives a ratio of 1).

4.4 Hybrid Model

The memory usage of $O(n^2)$ and the computation time of $O(n^3)$ of the optimal method limit the length of a video that dynamic programming can process. The greedy algorithm takes less memory and almost linear computation time, which makes it more suitable for extended videos. From Figure 2, we notice that the greedy key frames are very close to optimal key frames under low compression ratios. For improved performance for long videos, one feasible approach is to use a hybrid model. That is, for a video of n frames, we first use the greedy algorithm to calculate key frames levels from $n - 1$ to level k, then apply the dynamic programming algorithm on the selected k key frames to get key frames sequences of length from 1 to $k-1$. As shown in Figure 4, the results of the hybrid model has significant improvement over those of greedy algorithm in terms of energy values and longest common subsequences: it is immediately apparent from the figure which value was chosen by the user for k.

5 Video Indexing and Interactive Content Searching

The key frame hierarchy created by the greedy algorithm has two important applications: video indexing and streaming.

The results of the greedy algorithm are quite appropriate for video indexing and summarization. Since all frames can be ranked by their significance, it is easy to select any desired length of a key frame sequence by simply choosing an appropriate cut-off point in the frame significance array, as shown in Figure 5, the frames before this point are selected as key frames.

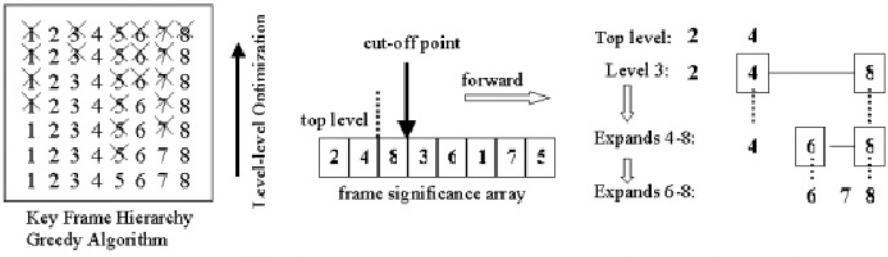

Fig. 5. Greedy algorithm and its application on interactive content retrieval. Figures from left to right are key frame hierarchy generated by greedy algorithm, frame significance array and interactive content browsing.

For interactive content retrieval and browsing of a video, different levels of the key frames can be displayed according to the frame significance array and the cut-off point. Moving the cut-off point in the direction of less significance displays more key frames, and vice versa. We may also use the frame significance array selectively to display more details of a video between any two designated key frames. For example, in a video of 8 frames as shown in Figure 5, the top level has frames f_2 and f_4. To display a bit more content, we move the cut-off point from "4-8" one step to "8-3", and therefore display frame f_8. Similarly, to expand content only between f_4 and f_8, we search the frame significance array starting from f_8 and display the first frame f_i such that $4 < i < 8$; here, that is frame f_6. This searching process can be simplified by storing two pointers with each frame number in the frame significance array, one pointing to the most recently discarded frame with frame index less than the current frame, and the second with frame index greater.

We have developed a software tool for interactive video browsing and searching based on these frame selection techniques. A user may browse different levels of key frames, or may view more key frames within a video segment, by a simple "click-drag" operation. The user selects a key frame as the start, clicks on a key frame to signal the end frame of a desired video segment, then drags the end frame to the right while holding the mouse button down. A number of most significant key frames will show up to fill the available space between these two key frames. To maximize the use of screen display, the key frames of selected video segment can be displayed in a new window below. Figure 6 shows the interactive video content browsing and searching results on one scene of the sit-com video "Friends".

The greedy algorithm can be easily used for dynamic compressed video steaming, particularly in a networked environment with heterogeneous platforms [8, 12,13]. The required compression ratio can change dynamically with available network bandwidth. Accordingly, the cut-off point in the significance array can also be adjusted dynamically even during the video streaming.

Fig. 6. Results of interactive video content browsing in one scene of video "friends". The numbers under frames show the actual frame numbers in the video. The top row is top level key frame overview, the second row expands two adjacent key frames in top level, the third row similarity expands a key frame pair of the second row.

6 Experimental Results

We compared our optimization algorithms to three other existing methods: a simple "first frame" method, a clustering method [17] and a set-cover method [3]. The simple method chooses key frames as follows. It selects the first frame as a key frame, and searches the frames after it. If a subsequent frame has a distance to the key frame larger than a designated threshold, this new frame is selected as the next key frame. This process continues until the end of the video. Clustering method use a dynamic clustering process, assigning frames to their nearest existing cluster, unless a new frame is "too far", in which case a new cluster is started. The set-cover method searches for the minimum number of frames whose semi-Hausdorff distance to the original frames is within a given threshold.

We propose an evaluation method for sparse sampling based on a new concept of "well-distributed" key frames. We would prefer that any key frame algorithm should avoid selecting multiple frames with the same visual content and should also avoid selecting frames within visual transitions (such as dissolves). We define a key frame which does not duplicate coverage within a shot and which do not fall within a transition between shots as "well-distributed"; this approximately measures the perceptual coverage of semantic content. For any given key frame sequence, we then use the number of "well-distributed" key frames as a criterion to compare algorithm performance.

Video A is a sample travelog video with multiple gradual shot-shot transitions, and it has a hand-measured ground truth total of 9 "well-distributed" key frames. The comparison of results of 6 key frame selection methods in Figure 8 plots the number of actual key frames selected by an algorithm versus how many are well-distributed. The ideal algorithm would grow linearly with all

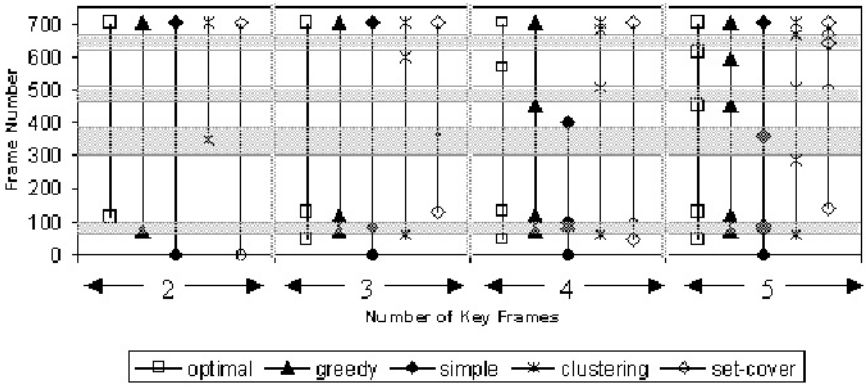

Fig. 7. Distributions of key frames selected by different methods for a video segment of 5 shots extracted from video A. The shaded areas show the transitions between shots.

frames well-distributed, until it plateaus at 9, as shown in the figure. Figure 7 shows the distributions of key frames extracted by different methods on a video of 5 shots. In Table 1, different key frame selection approaches are compared on two more videos, one is a sit-com video and another is a documentary. All results indicate that the optimal dynamic programming has the best results. The greedy algorithm and hybrid method are very close in results to optimal method, and all three are better than the three other existing algorithms. The clustering method is a little better than simple and set-cover methods, but still not as good as optimization methods, especially when the number of selected key frames is larger than the number of well-distributed frames.

More theoretically, when compared to our optimization methods, there are two other drawbacks in the clustering and set-cover methods. Firstly, these methods show non-monotonic behavior: sometimes more selected key frames generate less well-distributed key frames, usually due to the selection of the midpoint frame of a transition as a new cluster center or a new covering frame. In contrast, for the greedy method monotonicity is guaranteed, since key frames in a more restricted level always appear in less restricted level. Secondly, convergence is not guaranteed in the clustering and set-cover methods, which makes some numbers of key frames not available, as shown in the figure 8 and table 1. To retrieve a designated number of key frames using either method, it is necessary to iteratively adjust thresholds, but sometimes such a process does not converge. In contrast, for any of our three new methods, any number of key frames can be retrieved.

7 Conclusions and Future Research

In this paper, we presented and evaluated several novel approaches to video key frame selection. We cast the key frame selection problem to a global optimization

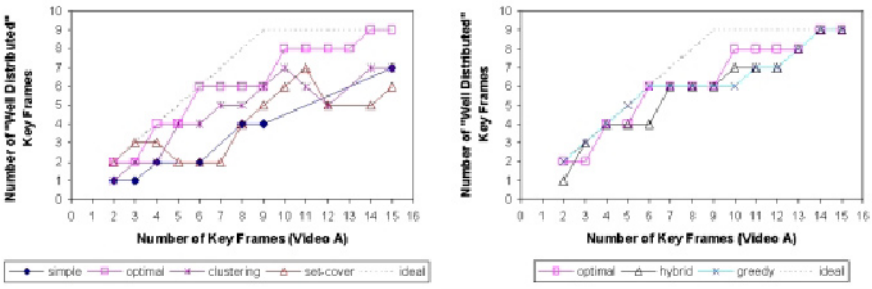

Fig. 8. Comparison of results of different key frame selection methods on a video of 9 "well-distributed" video key frames.

Table 1. Comparison of key frame selection results of different methods on two videos. Video B is one scene of sit-com video "Friends" of 31 "well distributed" key frames; Video C is a video segment extracted from documentary video "A Great Day in Harlem". Video C has 25 observed "well distributed" key frames. "NA" means value not available because of non-convergence.

	Video B							Video C						
number of key frames	6	12	15	22	26	30	40	4	8	12	16	20	25	30
optimal	5	11	14	21	25	29	31	4	8	11	15	18	20	24
hybrid	5	11	14	21	25	28	31	4	8	11	14	18	20	24
greedy	5	11	14	21	25	28	30	4	8	11	14	18	20	24
simple	5	11	13	NA	23	26	28	NA	7	11	13	15	NA	20
clustering	NA	10	12	19	24	26	30	4	NA	9	NA	17	19	20
set-cover	5	6	9	13	17	NA	18	2	5	7	12	12	NA	15

problem, and presented an optimal solution based on dynamic programming. Our method allows any key frame selection criteria as long as it can be expressed as a maximization of inter frame distance (for example, color, texture, and other measures derived from them that can be cast as a metric). To further reduce computation complexity, we developed a greedy algorithm using a bottom-up level-to-level optimization procedure, and the results of these two methods and their hybrid method were compared. We applied the results of key frame selection to video indexing and interactive video content retrieval. Experiments show our algorithm has better performance compared to other existing approaches, based on a definition of well-distributed key frames.

Currently our key frame selection methods, both the optimal dynamic programming method and the greedy method, are applied to videos off-line. We are investigating a buffer-based real-time process. In this paper, we choose the energy function to maximize as the sum of all distances of adjacent key frames, but other energy functions and other distance definitions may also be used for key frame selection, according to particular user requirements and applications.

References

1. Edoardo Ardizzone and Mohand-Said Hacid. A Semantic Modeling Approach for Video Retrieval by Content. In *IEEE International Conference on Multimedia Computing and Systems*, pages 158–162, 1999.
2. J. Boreczky and L. Rowe. Comparison of Video Shot Boundary Detection Techniques. In *Storage and Retrieval for Still Image and Video Databases*, pages 170–179, 1996.
3. H. S. Chang, S. Sull, and Sang Uk Lee. Efficient Video Indexing Scheme for Content-based Retrieval. In *IEEE Trans. on Circuits and Systems for Video Technology*, pages 1269–1279, Dec. 1999.
4. Tat-Seng Chua and Li-Qun Ruan. A Video Retrieval and Sequencing System. In *ACM Transactions on Information Systems*, pages 373–407, Oct. 1995.
5. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2001.
6. Madirakshi Das and Shih-Ping Liou. A New Hybrid Approach to Video Organization for Content-Based Indexing. In *IEEE International Conference on Multimedia Computing and Systems*, 1998.
7. Andreas Girgensohn and John Boreczky. Time-Constrained Keyframe Selection Technique. In *IEEE International Conference on Multimedia Computing and Systems*, pages 756–761, 1999.
8. I.Busse, B. Deffner, and H.Schulzrinne. Dynamic QoS Control of Multimedia Applications based on RTP. In *Computer Communications*, Jan. 1996.
9. F. Idris and S. Panchanathan. Review of Image and Video Indexing Techniques. In *Journal of Visual Communication and Image Representation*, pages 146–166, June 1997.
10. Jia-Ling Koh, Chin-Sung Lee, and Arbee L.P. Chen. Semantic Video Model for Content-based Retrieval. In *IEEE International Conference on Multimedia Computing and Systems*, pages 472–478, 1999.
11. M. K. Mandal, F. Idris, and S. Panchanathan. A Critical evaluation of image and video indexing techniques in compressed domain. In *Image and Vision Computing*, pages 513–529, 1999.
12. Myung-Ki Shin and Jin-Ho Hahm. Applying QoS Guaranteed Multicast Audio and Video to the Web. In *IEEE International Conference on Multimedia Computing and Systems*, pages 26–30, 1999.
13. Hugh M. Smith, Matt W. Mutka, and Eric Torng. Bandwidth Allocation for Layered Multicasted Video. In *IEEE International Conference on Multimedia Computing and Systems*, pages 232–237, 1999.
14. M. Smith and T. Kanade. Video Skimming and Characterization through the Combination of Image and Language Understanding. In *Proceedings of the IEEE International Worksop on Content-based Access of Image and Video Databases (ICCV'98)*, 1998.
15. M. Yeung and B. Liu. Efficient Matching and Clustering of Video Shots. In *Proceedings of the International Conference on Image Processing*, pages 338–341, 1995.
16. M. Yeung and B.L. Yeo. Time-Constrained Clustering for Segmentation of Video into Story Units. In *International Conference on Pattern Recognition*, pages 375–380, 1996.
17. Yueting Zhuang, Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Adaptive Key Frame Extraction Using Unsupervised Clustering. In *IEEE International Conference on Image Processing*, pages 866–870, 1998.