The CPR Model For Summarizing Video

M. Fayzullin, V.S.Subrahmanian *
University of Maryland
{fms,vs}@cs.umd.edu

A. Picariello
Università di Napoli
picus@unina.it

M.L. Sapino
Università di Torino
mlsapino@di.unito.it

ABSTRACT

Most past work on video summarization has been based on selecting key frames from videos. We propose a model of video summarization based on three important parameters: Priority (of frames), Continuity (of the summary), and non-Repetition (of the summary). In short, a summary must include high priority frames, must be continuous and non-repetitive. An optimal summary is one that maximizes an objective function based on these three parameters. We develop formal definitions of all these concepts and provide algorithms to find optimal summaries. We briefly report on the performance of these algorithms.

Categories & Subject Descriptors:

H.2.4 Multimedia Databases

General Terms:

Algorithms, Human Factors, Measurement, Theory

Keywords:

Multimedia, Video, Summarization

1. INTRODUCTION

Despite the vast amount of work on video databases, there has been relatively little work [3, 5, 4, 13, 6] to date on summarizing video and almost no work at all that both takes the content into account and that summarizes video in a manner that scales to massive data applications. For example, if FIFA (the International Soccer Federation) wanted to sell videos of soccer games, there would be tens of thousands of such videos. Potential customers may wish to watch small clips of the video to decide which videos they wish to buy. Though financial resources may be available to manually summarize each video, the ability to automatically summarize such videos is likely to be attractive.

In this paper, we propose a formal model for video summarization that takes three important properties into account:

1. **Continuity:** The summarized video must be as continuous as possible. A summary with a lot of "jumps" in it is unlikely to be attractive to users.

2. **Priority:** In a given summarization application (e.g. summarizing soccer videos), certain objects or events may be more important than others (e.g. a goal may be more important than a midfield pass). A summary must contain high priority items in it — but the domain expert must have the ability to set priorities.

3. **Repetition:** Even though an event may have high priority, if the event occurs repeatedly throughout the video, then it may become important to not repeat it over and over again and instead select other important events to show.

These three important criteria, which we call the **CPR** criteria, form the core basis for our summarization framework. Existing models such as those of [3, 4] do not take item (2) above into account. For instance, [3, 5] focuses on key frames that are determined by the amount of change, but key frames may not necessarily be important for the user. Similarly, [6] creates a user attention model based on the feature changes in the video to pick key frames and insert them into a summary. [4] focuses on using Powerpoint slides accompanying a presentation to create summaries and does not use the content of the video itself. Instead, attention span information is used to find "important" slides and include corresponding frames into a summary.

MMDB'03, November 7, 2003, New Orleans, Louisiana, USA.

Our **CPR** model consists of two key components: (i) use of rules to specify which objects and/or activities in a video are of interest (i.e. have high priority) for inclusion in a summary and (ii) an objective function that merges together the relative importance of priorities of objects/events, vis-a-vis continuity and repetition criteria. Once the rules and the objective function are articulated, *any* suite of video processing algorithms can be used for feature/activity extraction. We provide three algorithms for creating summaries: a dynamic programming based algorithm called *CPRdyn*, a genetic programming based algorithm called *CPRgen*, and a specialized *summary extension algorithm (SEA)*. About 200 students at the University of Naples have tested out our algorithms on 50 soccer videos with a view to determining which algorithm produced the "best quality" summaries (as evaluated by the students). We concluded that that the SEA algorithm is the fastest and also produces the best quality summaries.

2. SUMMARIZATION: FORMAL MODEL

Throughout this paper, we assume that every video v has a *length* $\mathsf{len_v}$ describing the number of frames in the video - the frames in a video of length $\mathsf{len_v}$ are labelled $1, \ldots, \mathsf{len_v}$.

In many cases, we may coalesce a group of contiguous frames into *blocks* and then create summaries based on determining which blocks (rather than frames) to include in the summary. The advantage of this approach is that the number of blocks in a video is much smaller than the number of frames. Our framework applies both to blocks and frames.

2.1 Summarization Content Specification

One of the most important parts of video summarization is to specify what kinds of content should be included within the summary. In this section, we propose the concept of a summarization content specification.

DEFINITION 2.1 (k-SUMMARY). *Suppose v is a video, $k \geq 0$ is an integer, and $S \subseteq \{1, \ldots, \mathsf{len_v}\}$ is a set of frames such that $card(S) \leq k$. Then S is called a* k-summary *of v.*

In other words, a k-summary of a video is any set of k or fewer frames from the video. Of course, some summaries will be better than others - the rest of this section describes a method to describe which frames are of *interest* to a user.

Summarization content is specified using a logical language which contains a unary predicate called insum that takes a frame as input. If f is either a frame number or a variable ranging over frames, then $\mathsf{insum}(f)$ is called an insum-*atom*. When $\mathsf{insum}(f)$ is true, this means that frame f is *of interest* for inclusion in the summary. Of course, not all frames of interest may be included in the summary that is eventually selected. Furthermore, some frames may be included in the final summary even if there is no interest in them because they may be required to ensure continuity of the produced summary.

We assume that the video database on top of which our summarization tools are built supports the following API (application program interface) functions:

- $\mathsf{findframe}(v, X)$: When X is either an object or an activity, this function returns the set of all frames in the video v containing that object or activity. For instance, $\mathsf{findframe}(v, X)$ can be implemented via algorithms such as those proposed in [12].
- $\mathsf{findobj}(v, f)$: Given a video v and a frame f, this returns the set of all the objects occurring in frame f of video v. $\mathsf{findobj}(v, f)$ can be implemented via algorithms such as those in [12].
- $\mathsf{findact}(v, f)$: This is similar to the previous function except that it returns all activities occurring in frame f of video v. $\mathsf{findact}(v, f)$ can be implemented via algorithms such as those in [10].

Most existing video databases (AVIS[1], OVID[9]) can support such functions. It is important to note that all the functions above return a set as output. This will be significant for us.

DEFINITION 2.2 (VIDEO-CALL). *Suppose vc is a video database API function, and $t_1, \ldots, t_n$ are arguments to vc (of the right type). Then $vc(t_1, \ldots, t_n)$ is called a* video call.

DEFINITION 2.3 (VIDEO-ATOM). *If vc is a video call and X is either a constant or a variable of the same type as vc's output, then $(X \in vc)$ is called a* video atom. *Likewise, if X, Y are either frames or variables ranging over frames and i is an integer, $\mathsf{near}(X, Y, i)$ is a video atom.*

For example, $X \in \mathsf{findact}(v, f)$ allows the variable X to be bound to any activity in frame f of video v. Intuitively, a frame Y satisfies $\mathsf{near}(X, Y, i)$ iff Y occurs in the the interval of frames starting at $X - i$ and ending at $X + i$ (including both). The near predicate is used to ensure continuity.

DEFINITION 2.4 (VIDEO-CONDITION). *If $va_1, \ldots, va_n$ are video atoms and E is a conjunction of equalities , then*

$(va_1 \wedge \ldots \wedge va_n \wedge E)$ *is a* video condition. *We will use the symbol* ϱ *(possibility adorned with subscripts and superscripts) to denote a video condition.*

For example, $X \in \mathsf{findobj}(v, f) \wedge X \in \mathsf{findobj}(v, f')$ finds all objects X that appear *both* in frame f and frame f' of video v.

DEFINITION 2.5 (SUMMARIZATION RULE). *A* summarization rule *is an expression of the form*

$$A \quad \leftarrow \quad \varrho \wedge A_1 \wedge \ldots \wedge A_m$$

where ϱ *is a video condition, and* $A, A_1, \ldots, A_m$ *are* insum-*atoms.*

Intuitively, the above rule says that if $A_1, \ldots, A_m$ are of interest for inclusion in a k-summary, and if ϱ is true, then A is also of interest for inclusion in the k-summary.

DEFINITION 2.6 (SUMMARY CONTENT SPECIFICATION). *A* video summary content specification $\mathcal{V}$ *is a finite set of summarization rules.*

Intuitively, a video summarization content specification $\mathcal{V}$ contains a finite set of rules. Based on these rules, we can derive a finite set of instantiated (i.e. variable free) video atoms. We use $\mathsf{Der}(\mathcal{V})$ to denote the set of all insum-atoms derivable from a video summary content specification $\mathcal{V}$. These atoms are the ones deemed to be of interest for inclusion in a summary.

DEFINITION 2.7 (VALID k-SUMMARY). *Suppose* $\mathcal{V}$ *is a video summary content specification. A* k*-summary* S *is* valid *w.r.t.* $\mathcal{V}$ *iff* $S \subseteq \mathsf{Der}(\mathcal{V})$.

The above definition says that for a k-summary to be valid, the inclusion of each frame in it must be justified by some rule in the summary specification.

EXAMPLE 2.1. *The following rules describe atoms of interest in a soccer video. According to the first rule, frames in which the action of scoring a goal appears, as well as the captain of a team, are interesting. The second rules states that celebration actions in frames make them interesting. The third rule states that any frame containing a pass action involving Totti is interesting, provided Totti appears in at least one frame whose importance has already been stated.*

$$\begin{array}{rcl}
insum(X) & \leftarrow & \text{“}goal\text{”} \in findact(v, X) \\
 & \wedge & \text{“}captain\text{”} \in findobj(v, X) \\
insum(Y) & \leftarrow & \text{“}celebration\text{”} \in findact(v, Y) \\
insum(Z) & \leftarrow & \text{“}pass\text{”} \in findact(v, Z) \\
 & \wedge & \text{“}Totti\text{”} \in findobj(v, Z) \\
 & \wedge & \text{“}Totti\text{”} \in findobj(v, X) \\
 & \wedge & insum(X)
\end{array}$$

2.2 Priority Specification

A *priority function* for a video is a mapping pri from sets of frames to natural numbers. Intuitively, $\mathsf{pri}(\{f_1, f_2\}) = 5$ means that the priority of including *both* f_1, f_2 in a summary is 5. Priority functions can be explicitly stated in one of many ways. An example priority function specification mechanism is shown below.

EXAMPLE 2.2 (AGGREGATED TABULAR PRIORITY). *In this method, we have a table having the schema* ($FrameSet$, $Priority$)*. An example is given eblow.*

$\{f_1, f_2\}$	*5*
$\{f_1\}$	*3*
$\{f_2, f_3\}$	*7*

Given such a table and a set F *of frames, many different priority functions may be defined, some of which are shown below.*

1. **Subset-average:** *This function finds all tuples in the table whose* $FrameSet$ *field is a subset of* F *and returns the average of the priority fields of such tuples. For example, with respect to the above table, if* $F = \{f_1, f_2, f_4\}$*, this function would return* 4 *(average of* 5 *and* 3*).*

2. **Maximal Subset Average:** *This function finds all tuples* t *in the table whose* $FrameSet$ *field is a* maximal *subset of* F *(i.e. there is no other tuple* t' *with* $t.FrameSet \subset t'.FrameSet$ *such that* $t'.FrameSet \subseteq F$*) and takes the average priorities of such tuples. In the above example, if* $F = \{f_1, f_2, f_3\}$*, then this priority function would return* 6 *(average of* 5 *and* 7*). Note that the second tuple would not be maximal and hence its associated priority would not be involved in the average computation.*

We can also specify priorities via rules.

2.3 Continuity Specification

Continuity is an important criterion to be taken into account when computing an appropriate summary. For example, consider a soccer match with one goal. To show the goal effectively, a summary should probably include a segment of video immediately preceding the goal, and immediately thereafter. This is an example of a continuity requirement.

Definition 2.8 (Continuity function). *Suppose v is a video, $k \geq 0$ is an integer, and Σ is the set of k-summaries of v. A* continuity function *w.r.t. v is a mapping $\chi : \Sigma \to \mathcal{N}$.*

The notion of a continuity function above is very general. Different summarization applications may use different instances of this general definition. For example, a notion of distance between frames can be used to define the continuity function, as follows.

Example 2.3. *Suppose $H(f) = (h_1, h_2, \ldots, h_n)$ is a function that returns the color histogram for a given frame f. Each h_j corresponds to the number of pixels in a region of some color space. A good perceptually uniform space is the* Hue Saturation Value, HSV *space or alternatively we may use the* Opponent Colors *space. Let d be any measure of distance between two histograms (e.g. d could be the well known L_1 or L_2 norms). Now set the distance of a summary $f_1, \ldots, f_k$ to be*

$$\Sigma_{i=1}^{k-1} d(H_i, H_{i+1})$$

where H_i is the color histogram of the i'th frame f_i in the summary.

2.4 Repetition Specification

The third important property of a summary is that it must not contain repetitive information. A video spanning 90 minutes will probably have at least a few key scenes. Summaries should probably show clips of each of these scenes, rather than just one. The goal of a repetition specification is to avoid repetitions.

Definition 2.9 (Repetition function). *Suppose v is a video, $k \geq 0$ is an integer, and Σ is the set of k-summaries of v. A* repetition function *w. r. t. v is a mapping $\rho : \Sigma \to N$.*

As in the case of continuity functions, repetition functions are very general in nature. There are thousands of possible repetition functions. Two examples are given below.

Example 2.4 (frame-distance based repetition). *Suppose S is a summarization of a video and d is a distance function. Then we could define three repetition functions min_d, sum_d and avg_sum_d as follows.*

$$\begin{aligned} min_d(S) &= min\{d(f_1, f_2) \mid f_1, f_2 \in S \wedge f_1 \neq f_2\}. \\ sum_d(S) &= \Sigma_{f_1, f_2 \in S \wedge f_1 \neq f_2} d(f_1, f_2). \\ avg\_sum_d(S) &= \frac{\Sigma_{f_1, f_2 \in S \wedge f_1 \neq f_2} d(f_1, f_2)}{card(S)}. \end{aligned}$$

It is important to note that various frame distance functions can be used which measure the distance between one frame and another.

Example 2.5 (object/activity repetition). *Given a function wt which assigns weights to objects, an object o, and a set S of frames, the repetition of o in S is given by*

$$rep_S(o) = wt(o) \cdot card(\{f \in S \mid o \in f\}).$$

Given an activity a, $rep_S(a)$ may be defined similarly. We may now define the repetition of S as

$$rep(S) = \Sigma_o rep_S(o) + \Sigma_a rep_S(a).$$

2.5 Optimal Summary

Suppose a summarization developer has specified a summarization content specification, a repetition function, a priority function, and a continuity function. In order to define what an optimal summary is, we first need a way of evaluating a summary based on the criteria of continuity, priority, and repetition. This is formalized by the concept of a summary valuation below.

Definition 2.10 (summary valuation). *Suppose $\mathcal{V}$ is a video summary content specification, SUM is the set of all summarizations of a given video v, and $\alpha, \beta, \gamma \geq 0$ are integers. A* summary valuation *is a function $eval : SUM \to R$, of the form*

$$eval(S) = \alpha \cdot \chi(S) + \beta \cdot \mathsf{pri}(S) - \gamma \cdot \rho(S).$$

In the above definition, the constants α, β, γ denote the respective importance to be given to continuity, priority, and repetition criteria. Users do not have to explicitly write such an objective function - they can use simple sliders on a GUI to set these weights.

Definition 2.11 (k-summary computation problem). *Suppose $\mathcal{V}$ is a video summary content specification. A k-summary S is* optimal *w.r.t. $\mathcal{V}$ and a summary valuation $eval(S)$ iff (i) it is valid w.r.t. $\mathcal{V}$ and (ii) there is no other valid k-summary S' w.r.t. $\mathcal{V}$ such that $eval(S) < eval(S')$.*

THEOREM 2.1. *Computing an optimal k-summary is NP-complete.*

The proof is by a reduction of the knapsack problem [2] to the optimal k-summary computation problem.

3. SUMMARIZATION ALGORITHMS

In this section, we introduce several alternative summarization algorithms. The first algorithm, **CPRopt** finds an optimal k-summarization without making any assumptions about the priority, continuity, and repetition functions. However, as the optimal k-summary computation problem is NP-complete, this algorithm takes an exponential amount of time (w.r.t. the length of a video) which is clearly unacceptable. Even if we assume frames are being played at 15 frames per second and we have a 1-hour video, we would have $60 \times 60 \times 15 = 54,000$ frames – so any algorithm for optimally finding a k summary for this video would have complexity $\mathbf{O}(2^{54,000})$ which is a staggeringly large number.

As a consequence, we also designed and implemented three alternative heuristic k-summarization algorithms. The first algorithm, CPRdyn is based on dynamic programming, the second called CPRgen is based on genetic programming, while the third algorithm called the Summary Extension Algorithm (SEA for short) is based on a concept called summary extension.

3.1 The Optimal Summarization Algorithm

The **CPRopt** algorithm is a recursive algorithm that is always guaranteed to find an optimal k-summarization. The outline of the algorithm is as follows. This algorithm is given for completeness of exposition. As argued above, any algorithm for finding optimal k-summaries is going to be exponential (unless $P = NP$).

- **Compute** $\mathsf{Der}(\mathcal{V})$**:** The first major step of the **CPRopt** algorithm is to compute the set of all (variable free) insum-atoms in $\mathsf{Der}(\mathcal{V})$. It is easy to see that this step can be executed in time linear in the number of frames in the video v. We know by definition that any valid summarization is a subset of (or equals) $\mathsf{Der}(\mathcal{V})$.

- **Examine subsets:** Let $\Sigma = \{S \mid S \subseteq \mathsf{Der}(\mathcal{V})$ and $card(S) \leq k\}$. Each such subset is a valid k-summary of $\mathcal{V}$.

- **Evaluate k-Summaries:** Apply the evaluation functions to all of summarizations in Σ and choose the best one.

```
Procedure CPRopt(𝒱,k)
    𝒱 is a video summary content specification
    k is a desired summary length
begin
    V := ∅
    Δ := ∅
    repeat
        // A, A1, . . . , An are variable-free
        V := V ∪ Δ
        Δ := {A | A ← ϱ ∧ A1 ∧ . . . ∧ An ∈ 𝒱 ∧ ϱ ∧ {A1, . . . , An} ⊆ V}
    until Δ \ V = ∅
    Σ := {S | S ⊆ V and card(S) ≤ k}
    BestS := S ∈ Σ such that α · χ(S) + β · pri(S) − γ · ρ(S) is maximal
    return BestS
end.
```

3.2 Heuristic Algorithms

In this section, we develop three heuristic algorithms to compute k-summaries. These algorithms are much faster than the CPRopt algorithm, but may tradeoff optimality of the summary produced.

3.2.1 *The* CPRdyn *Algorithm*

The CPRdyn algorithm is based on *dynamic programming* [2]. The algorithm maintains a variable $v_{current}$ describing the best solution found so far. Initially, $v_{current}$ consists of k randomly chosen frames which are derivable from $\mathcal{V}$. The algorithm changes $v_{current}$ in each iteration by checking to see whether replacing a frame in $v_{current}$ by a frame which is absent from $v_{current}$ will lead to a better summary. CPRdyn can be summarized as follows.

```
Procedure CPRdyn(𝒱,k)
    𝒱 is a video summary content specification
    k is a desired summary length
begin
    // Fill v_current with k randomly selected frames from Der(𝒱).
    v_current := {f_i | i ∈ [1, k] ∧ f_i ∈ Der(𝒱)}
    // Put the remaining frames into v^C.
    v^C := Der(𝒱) − v_current
    while v^C ≠ ∅
        subs := false
        r := 1
        while r < k and subs = false
            // Build a new tentative solution by replacing f_r with a frame from v^C.
            v_tentative := (v_current \ {f_r}) ∪ {first(v^C)}
            if eval(v_current) < eval(v_tentative) then
                v_current := v_tentative
                add f_r to the tail of v^C
                subs := true
            else
                r := r + 1
            end if
        end while
        remove first(v^C) from v^C
    end while
    return v_current
end.
```

3.2.2 *The* CPRgen *Algorithm*

We now present the CPRgen algorithm which uses genetic programming methods [2] to compute a k-summary. The first issue is to represent the problem in terms of decisional variable strings. We use a *binary* representation: each frame has an associated binary variable indicating the presence/absence of the frame in the optimal summary. The main idea of the algorithm is described below.

- **Initialization:** A random population of summaries each satisfying the summary size requirement is chosen.

- **Fitness evaluation:** The fitness function is equal to the described $eval()$ function
- **Selection:** We consider members of the population elements according to a decreasing fitness.
- **Generation:** A mutation operator is applied over the selected elements, thus creating a random transformation of the summary.
- **Elimination:** The element having the smallest evaluation value is eliminated.
- **Termination:** The algorithm stops when the new summaries are similar, i.e. the variation of the fitness functions within the population of solutions is less than a threshold ϵ.

```
Procedure CPRGen(𝒱,k,N,δ)
    𝒱 is a video summary content specification
    k is a desired summary length
    N is the desired number of iterations
    δ is the desired fitness threshold
begin
    R := ⌈len_𝒱 / k⌉
    Compute an initial population of random solutions V := (v_i)_{i=1...R}
    based on frames from Der(𝒱)
    for j ∈ [1, N]
        for i ∈ [1, R]
            v̄ := a solution randomly chosen among the ones in V
            Select a frame f̄ from the video
            if f̄ ∈ v̄ then
                Choose another frame from the video
                Insert the new frame in v̄ eliminating f̄
                Add v̄ to the population of solutions V
                Eliminate from V the solution with the smallest fitness
                if max_{v1,v2∈V} |fitness(v1) − fitness(v2)| ≤ δ then
                    Return any solution from V
                end if
            end if
        end for
    end for
    Return the best solution from V
end.
```

3.3 The Summary Extension Algorithm (SEA)

The SEA algorithm uses the CPR model described in this paper in a specific way in order to compute summaries. Before defining the algorithm, some intermediate definitions are needed. We first introduce the concept of *frame coverage*. Given some condition C that we want frames to satisfy (e.g. containing a goal in a soccer video), the pair (f, p) describes how well frame f satisfies the condition C. The larger p is, the better frame f satisfies the condition C.

Definition 3.1 (frame coverage pair). *If f is a frame of video v, and $p \in [0, 1]$, then (f, p) is a* frame coverage pair.

Definition 3.2 (FCP Set Union). *Given two sets V_1, V_2 of frame-coverage pairs, the* FCP set union

$$V_1 \cup V_2 = \{(f,p) \mid p = \left\{ \begin{array}{l} p_1 \ if \ (f,p_1) \in V_1 \wedge \nexists (f,p_2) \in V_2 \\ p_2 \ if \ (f,p_2) \in V_2 \wedge \nexists (f,p_1) \in V_1 \\ p_1 \ if \ (f,p_1) \in V_1 \wedge (f,p_2) \in V_2 \wedge p_1 \geq p_2 \\ p_2 \ if \ (f,p_1) \in V_1 \wedge (f,p_2) \in V_2 \wedge p_1 < p_2 \end{array} \right. \}.$$

The SEA model induces priorities on insum-atoms by first attaching weights to rules in video content specifications, and by assuming that every variable occurring in a rule also appears in an insum-atom (in either head or body of a rule). Under this assumption on video content specifications, we may define what it means for a frame coverage pair to satisfy a rule.

Definition 3.3. *Suppose r is a rule in a video content specification $\mathcal{V}$, S is an* FCP *set representing a summary, X is a variable in the head of r, and f is a frame. Let w_r denote the weight of r in $\mathcal{V}$. Consider a substitution θ that replaces X with $(f, 1)$ and every other variable $X_i \in r$ with $(f_i, p_i) \in S$. Then we define a $[0, 1]$-valued function $\phi(r, f, S)$ as follows.*

$$\phi(r, f, S) \quad = \quad w_r \cdot max_{\{\theta \mid \varrho\theta = true\}}(min_{f_i \in body(r\theta)}(p_i))$$

When rule r has an empty body, $\phi(r, f, S) = w_r$. We extend ϕ to apply to a video content specification by setting $\phi(\mathcal{V}, f, S) = max_{r \in \mathcal{V}}(\phi(r, f, S))$.

Intuitively, $\phi(\mathcal{V}, f, S)$ assesses the value of inserting f into summary S. Notice that $\phi(r, f, S)$ is defined in terms of p_i values of the insum-atoms participating in r that, in turn, were results of computing ϕ. As r's body is a conjunction, $\phi(r, f, S)$ is computed as the minimal ϕ of its atoms.

Definition 3.4 (satisfaction). *The* FCP $(f, \phi(\mathcal{V}, f, S))$ satisfies $\mathcal{V}$ *w.r.t. summary S iff $\phi(\mathcal{V}, f, S) > 0$. A summary S is called* satisfactory *iff every pair $(f, p) \in S$ satisfies $\mathcal{V}$ w.r.t. S.*

We are only interested in summaries containing frames that satisfy the video content specification in question and of these, to pick one that optimizes an objective function involving priority, continuity and (non) repetition. The SEA algorithm finds satisfactory summaries by using the *valid summary extension*.

Definition 3.5 (Valid Summary Extension). *Let $\mathcal{V}$ be a video content specification, and S be a satisfactory summary. Then the* valid summary extension $\mathsf{VSE}_{\mathcal{V}}(S)$ *is the set $\{(f, \phi(\mathcal{V}, f, S)) \mid \phi(\mathcal{V}, f, S) > 0\}$.*

In other words, $\mathsf{VSE}_{\mathcal{V}}(S)$ is the set of all frame-coverage pairs that satisfy $\mathcal{V}$ with respect to the summary S. Here is an algorithm to compute $\mathsf{VSE}_{\mathcal{V}}(S)$:

```
Procedure VSE(𝒱,S)
    𝒱 is a video content specification
    S is the current summary
begin
    S' := ∅
    for each video frame f
        for each rule r ∈ 𝒱 such that head(r) = insum(X)
            // Compute φ(r, f, S) and add f to the result if φ(r, f, S) > 0.
            p_out := ChooseVars(r, {X = (f, 1)}, S)
            if p_out > 0 then S' := S' ∪ {(f, p_out)}
        end for
    end for
    return S'
end.
```

The $VSE()$ algorithm iterates over all rules in $\mathcal{V}$ and all frames in a video looking for frames that satisfy $\mathcal{V}$. $VSE()$ uses the $ChooseVars()$ algorithm to compute $\phi(r, f, S)$ for each frame f and rule r, and adds $(f, \phi(r, f, S))$ to the output if $\phi(r, f, S) > 0$. As FCP *set union* is used to add new frame-coverage pairs to the output, pairs with lower coverages are automatically replaced with higher coverage pairs.

```
Procedure ChooseVars(r,θ,S)
        r is a rule
        θ is a substitution
        S is the current summary
begin
        if there is variable X ∈ r such that X is not affected by θ then
                p_out := 0
                for each FCP (f, p) ∈ S
                        // Assign one more variable and recurse, maximizing p_out.
                        p' := ChooseVars(r, θ ∪ {X = (f, p)}, S)
                        if p' > p_out then p_out := p'
                end for
        else
                // Substitute variables and compute p_out.
                p_out := min_(f',p')∈θ (p')
        end if
        // Return φ(r, f, S).
        return w_r · p_out
end.
```

Suppose we start with some rule set $\mathcal{V}$ and an empty summary $S = \emptyset$ that *is* satisfactory w.r.t. $\mathcal{V}$. $S' = \mathsf{VSE}_{\mathcal{V}}(\emptyset)$ will contain all assignments satisfying the rules whose bodies are free of membership atoms. Satisfaction of such "self-supporting" rules does not require any blocks to be in the summary. Notice that S' is *always* going to be a satisfactory summary w.r.t. $\mathcal{V}$ and it is always true that $\forall(f, \phi) \in S : \exists(f, \phi') \in S' : \phi' \geq \phi$.

We continue to apply the $\mathsf{VSE}()$ operator to S' until it stops growing. We now present the SEA algorithm that finds the FCP set corresponding to the best summary by performing a greedy breadth-first search with the branching factor limited to N:

```
Procedure SEA(V,S,l,N)
        V is a video content specification
        S is the initial summary
        l is the maximal summary length
        N is the maximal branching factor
begin
        // Q is a sorted list of up to N summaries
        Q := ∅
        S' := VSE_V(S)
        // Remove assignments already present in S.
        for each FCP (f, p') ∈ S' such that ∃(f, p) ∈ S
                if p' > p then S := S ∪ {(f, p')}
                S' := S' − {(f, p')}
        end for
        // Find N best summaries...
        for each FCP set V ⊆ S' such that length(V ∪ S) ≤ k
                Q.add(V ∪ S, worth(V ∪ S))
                if size(Q) > N then Q.delete(tail(Q))
        end for
        // ...and try to grow them.
        if Q = ∅ then BestS := S
        else
                BestS := head(Q)
                for each summary V ∈ Q
                        V' := SEA(R, V, l, N)
                        if worth(V') > worth(BestS) then BestS := V'
                end for
        end if
        return BestS
end.
```

4. EXPERIMENTS

We have implemented the three heuristic algorithms proposed above in JAVA (with Oracle 8i and MS Access back-ends) on a Windows 2000 platform. The implementation consisted of approximately 2500 lines of code.

Using a collection of 50 soccer videos, a group of approximately 200 students at the University of Naples evaluated the quality of the summaries produced. Each summary received an A through E rating. Figure 1 shows a graph of the qualities of the summaries produced. In 67% of the cases, the SEA algorithm was deemed to produce the best results. Furthermore, 81% of the participants gave the SEA algorithm an A.

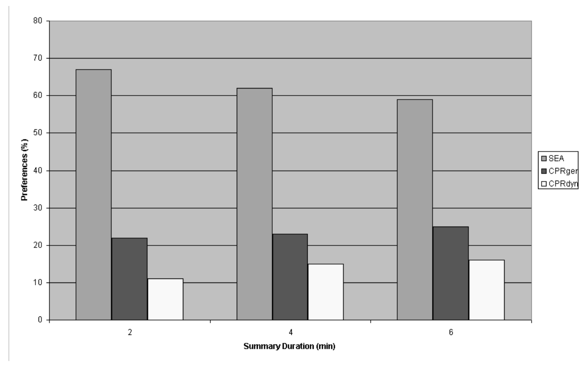

Figure 1: Comparing Quality of Results

In addition, we assessed the performance of the three algorithms using a Pentium3 800MHz machine with 128MB SDRAM. Figure 2 shows the results. As the reader can see, the SEA algorithm outperforms the other two algorithms.

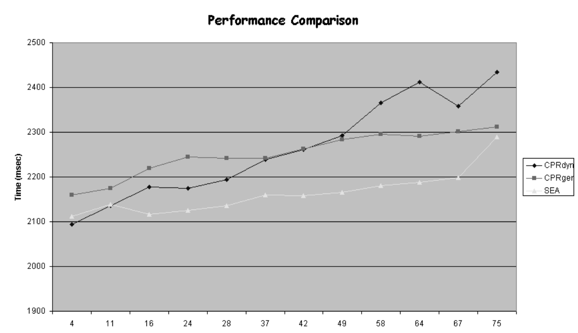

Figure 2: Comparing Algorithms' Performance

5. RELATED WORK

He et. al. [4] summarize videos of talks that are accompanied by PowerPoint slides. The priority of a video segment is determined by: (i) the moment when slides were changed, (ii) lecturer's voice pitch, and (iii) users' interest in different parts of the presentation. We do not assume that videos are accompanied by PPT presentations. In their framework, no video analysis was performed and application users have no control over what will be summarized.

DeMenthon et. al. [3] represented a changing vector of frame features (such as overall macroblock luminances) with a multi-dimensional curve and applied a curve simplification algorithm to select "key" frames. While this approach works well for the key frame detection, it does not consider the fact that certain events have higher priorities than others, and that continuity and repetition are important. Ju et. al. [5] propose another key frame based approach that chooses frames based on the motion and gesture estimation.

Zhou et. al. [13] attempt to analyze video content, extract and cluster features to classify video semantically. They apply a rule-based classification system to basketball videos and report on the results.

Ma et. al. [6] present a generic framework for video summarization based on estimated user attention. The framework uses computational attention models to predict attention.

6. CONCLUSIONS

This is the first model for summarizing video based on the semantic content of the video as well as based on user input about the objects and events in the video deemed to be independent. We developed a theoretical model for summarization, showed that computing summaries is NP-complete, and developed several algorithms to compute summaries. We performed an experimental analysis showing our algorithms produce excellent summaries in a short time.

Acknowledgements. Work supported in part by ARO grant DAAD190310202, ARL grants DAAD190320026 and DAAL0197K0135, and NSF grants IIS0329851 and 0205489.

7. REFERENCES

[1] S. Adali, K.S. Candan, S.-S. Chen, K. Erol, and V.S.Subrahmanian. *Advanced Video Information Systems.* ACM Multimedia Systems Journal, Vol. 4, 1996, pp. 172-186.

[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *"Introduction to Algorithms, 2nd Edition"* MIT Press, 2001.

[3] D. DeMenthon, D.S. Doermann, and V. Kobla. *Video Summarization by Curve Simplification.* Proc. ACM Multimedia, Bristol, England, 1998, pp. 211-218.

[4] L. He, E. Sanocki, A. Gupta, and J. Grudin. *Auto-Summarization of Audio-Video Presentations.* ACM Proc. on Multimedia, 1999, pp. 489-498.

[5] S. Ju, M. Black, S. Minneman, and D. Kimber. *Summarization of Videotaped Presentations: Automatic Analysis of Motion and Gesture.* IEEE Trans. on Circuits and Systems for Video Technology, Vol. 8(5), 1998, pp. 686-696.

[6] Y.P. Ma, L. Lu, H.J. Zhang, and M. Li. *A User Attention Model for Video Summarization.* Proc. ACM Multimedia, 2002.

[7] H.Martin and R.Lozano. *Dynamic Generation of Video Abstracts Using an Object Oriented Video DBMS.* Networking and Information Systems Journal, Vol. 3(1), 2000, pp. 53-75.

[8] H.R. Naphide and T.S. Huang. *A Probabilistic Framework for Semantic Video Indexing, Filtering, and Retrieval.* IEEE Transactions on Multimedia, Vol. 3(1), 2001, pp. 141-151.

[9] E. Oomoto and K. Tanaka. *OVID: Design and Implementation of a Video-Object Database System.* IEEE TKDE (Multimedia Information Systems), Vol. 5(4), 1993, pp. 629-643.

[10] C. Stauffer and E. Frimson. *Learning Patterns of Activity Using Real-Time Tracking.* IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 22(8), 2000, pp 747-757.

[11] V.S. Subrahmanian. *"Principles of Multimedia Database Systems"* Morgan Kaufmann, 1998.

[12] D. Zhong and S.F. Chang. *Video Object Model and Segmentation for Content-Based Video Indexing.* IEEE Intern. Conf. on Circuits and Systems, June, 1997, Hong Kong.

[13] W. Zhou, A. Vellaikal, and C.C. Jay Kuo. *Rule-Based Video Classification System for Basketball Video Indexing.* Proc. ACM Multimedia Workshop, 2000, pp. 213-216.