A NEW ALGEBRAIC APPROACH TO RETRIEVE MEANINGFUL VIDEO INTERVALS FROM FRAGMENTARILY INDEXED VIDEO SHOTS*

Sujeet Pradhan
Kurashiki Univ. of Science & the Arts
sujeet@soft.kusa.ac.jp

Takashi Sogo
Kobe University
sogoh@db.cs.kobe-u.ac.jp

Keishi Tajima
Kobe University
tajima@db.cs.kobe-u.ac.jp

Katsumi Tanaka
Kobe University
tanaka@db.cs.kobe-u.ac.jp

Abstract Video data consists of a sequence of shots. Over the past several years, substantial progress has been made in automatically detecting shot boundaries based on changes of visual and/or audio characteristics. There has also been considerable progress in indexing such video shots by automatically extracting keywords using techniques such as speech and text recognition. Shots detected by those techniques, however, are very fragmental. A single shot itself is rarely self-contained and therefore may not carry enough information to be a meaningful unit. A meaningful interval that interests common users generally spans several consecutive shots. There hardly exists any reliable technique for identifying all such meaningful intervals in advance so that any possible query can be answered.

In this paper, rather than identifying meaningful intervals beforehand, we shift our focus on how to compute them dynamically from fragmentarily indexed shots, when queries are issued. We achieve our goal by using two techniques — *glues* and *filters*. *Glues* are algebraic operations for composing all the longer intervals, which can be meaningful answers to a given query, from a set of shorter indexed shots. Glue operations do not count on any limit to the length of resulting intervals. Consequently, lengthy intervals containing several irrelevant shots are also expected to be composed as possible answers. Therefore, we provide filter functions so that such lengthy intervals are excluded from the answer set and only few relevant intervals are returned to the user. Both glues and filters possess certain algebraic properties that are useful for an efficient query processing.

Keywords: video query model, interval query, glue operations, interval filters

*This work is supported partly by the Japanese Ministry of Education under Grant-in-Aid for Scientific Research on Priority Area: "Advanced Databases", No. 08244103 and partly by Research for the Future Program of JSPS under the Project "Researches on Advanced Multimedia Contents Processing".

1. INTRODUCTION

Video segmentation is the most fundamental process for appropriate indexing and retrieval of video intervals. In general, video streams are composed of shots[1] delimited by physical shot boundaries. Substantial work has been done on how to detect such shot boundaries automatically (Arman et al., 1993) (Zhang et al., 1993) (Zhang et al., 1995) (Kobla et al., 1997). Through the integration of technologies such as image processing, speech/character recognition and natural language understanding, keywords can be extracted and associated with these shots for indexing (Wactlar et al., 1996). A single shot, however, rarely carries enough amount of information to be meaningful by itself. Usually, it is a semantically meaningful interval that most users are interested in retrieving. Generally, such meaningful intervals span several consecutive shots.

There hardly exists any efficient and reliable technique, either automatic or manual, to identify all semantically meaningful intervals within a video stream. Works by (Smith and Davenport, 1992) (Oomoto and Tanaka, 1993) (Weiss et al., 1995) (Hjelsvold et al., 1996) suggest manually defining all such intervals in the database in advance. However, even an hour long video may have an indefinite number of meaningful intervals. Moreover, video data is multi-interpretative. Therefore, given a query, what is a meaningful interval to an annotator may not be meaningful to the user who issues the query. In practice, manual indexing of meaningful intervals is labour intensive and inadequate.

Some efforts have been made in automatically detecting and indexing meaningful intervals in advance for retrieval (Wactlar et al., 1996) and for browsing (Yeung et al., 1996). The former one (Wactlar et al., 1996) decomposes a video stream into *paragraph units* which they consider to be pre-defined answers for pre-defined queries. The latter one (Yeung et al., 1996) identifies *story units* within a video stream on the basis of visual similarity and temporal locality relationship among video shots. Although they are successful to some extent, there still remains the problem of discrepancies between the granularity of the answer intervals that have been detected and the granularity of the intervals that end users expect to retrieve. It is because unless a user issues a query, it is not clear what meaningful intervals are to be identified beforehand.

Suppose we have a live video stream of a baseball match, which has been segmented into shots $\{s1, s2, \ldots, sn\}$ and indexed (Figure 1).

- ☐ $s3$ shows Matsui (a Japanese baseball player) preparing to face the next pitch.
- ☐ $s4$ is the shot of Matsui hitting the ball.
- ☐ $s5$ is the shot of the ball clearing the fence.
- ☐ $s6$ shows a glimpse of spectators cheering.
- ☐ $s7$ shows Matsui completing the run and touching home plate.

[1]A shot is a continuous sequence of frames captured from a single camera with no shutter interruption

Now let us consider this query: "retrieve a video interval which shows `Matsui` hitting a `homer`". Rarely would a user hope to retrieve only shot $s4$, which shows the actual moment of `Matsui` hitting the ball, or only $s5$ that shows the ball crossing the fence. It is clear that at shot level, there will be only fragmentary answers to this query. A user would hope to see, at least, the interval starting from the shot $s4$ till the end of $s5$. It is difficult to identify the exact answer interval to such queries beforehand. There are many intervals (intervals represented by s3-s7, s3-s5, s4-s5, s4-s7 in Figure 1) that can be considered meaningful and thus should be returned as answers to the above query. Only the query issuer knows which is the best among these four intervals for him or her.

This paper takes a different approach towards answering keywords-based video queries. We assume that video streams are segmented only at shot level since that is the best we can do with the present technology. We also assume that keywords are associated fragmentarily with these shots. Generally, we are less interested in what kinds of queries are going to be issued and what intervals are to be indexed to answer such queries. Instead, we focus our work on how to compose intervals that could possibly be the answers to a given set of keywords. In order to do so, we define a set of new algebraic operations, what we call *glue* operations that dynamically composes answer intervals from a set of indexed shots. Intuitively, these operations will enable us to compose all possible answer intervals to a given query; first by selecting the valid video shots for each query keyword and then by gluing those shots together. It should be noted here that our approach to retrieving meaningful intervals is syntactic

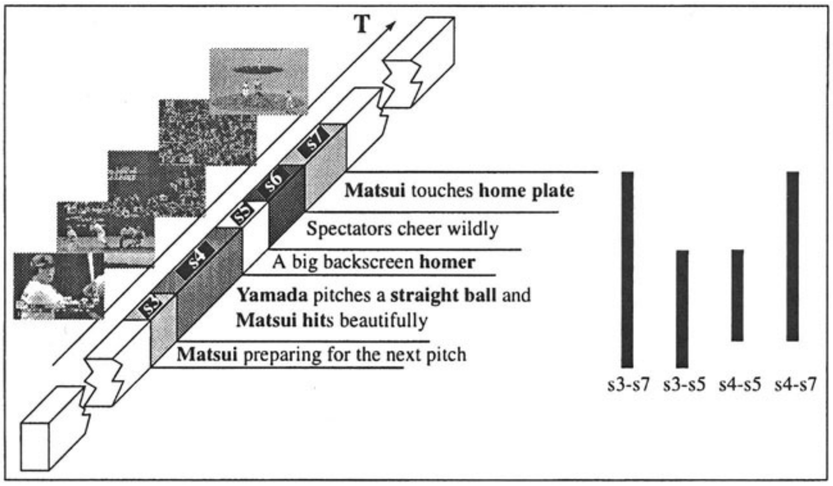
T
s3
s4
s5
s6
s7
Matsui touches home plate
Spectators cheer wildly
A big backscreen homer
Yamada pitches a straight ball and Matsui hits beautifully
Matsui preparing for the next pitch
s3-s7
s3-s5
s4-s5
s4-s7

Figure 1 Video stream segmented into shots and indexed

rather than semantic.[2] The glue operations possess certain algebraic properties which allow a simple and efficient composition of answer intervals.

As we have no prior knowledge about the boundaries of meaningful intervals, glue operations will go on composing longer intervals as far as valid shots associated with the query keywords are found for gluing together. In the worst case, if two query keywords happen to fall at the beginning and at the final shot of the video stream, there is always a possibility of returning the full one hour video as a valid answer. However, lengthy intervals containing plenty of unnecessary shots are usually irrelevant to users and as many of them as possible should be excluded from the answer set. We propose filtering techniques for discarding answer intervals that can be thought of irrelevant to users. Intuitively, we provide a filter function that takes a set of video intervals as input, and returns the subset that meets some necessary conditions for a query match. Importantly the proposed filter functions also possess certain algebraic properties and can be well-integrated with the glue operations. As a result, considerable number of irrelevant intervals can be removed at the initial stage of query processing, which eventually leads to a groundwork for an efficient query mechanism. The detailed explanation is given later.

2. INTERVAL COMPOSITION TECHNIQUES

A great deal of work has been done in the past for composing new video intervals from a set of intervals. Various interval operations such as union, intersection, concatenation and their set-variants have been defined and redefined in (Oomoto and Tanaka, 1993) (Weiss et al., 1995) (Hwang and Subrahmanian, 1996) (Hjelsvold et al., 1996). However, given a set of fragmentarily indexed video shots, these operations generally produce fragmentary intervals and thus cannot always produce appropriate intervals which users generally intend to find in the first place.

2.1. QUERY INTERPRETATION

Let us again consider the query "retrieve a video interval which shows `Matsui` hitting a `homer` (Figure 1). In conventional approaches, a query result is computed by simply taking the intersection between those video intervals with an attribute value `Matsui` and those with an attribute value `homer`. An actual scene of '`Matsui` hitting a `homer`', however, usually consists of several shots. It is primarily because video productions involve lots of switching

[2] In other words, we do not consider the semantics in the keywords themselves. For example, a keyword like `bat` may mean either a *bird* or a *wooden stick* used in sports. Nor do we consider any semantics in the way they are ordered. For example, a query like $\{\mathtt{dog}, \mathrm{run}, \mathrm{man}\}$ may retrieve intervals showing not only "a dog running after a man" but also "a man running after a dog".

between the cameras, camera movements, zooming, and panning. It rarely happens that each shot in that scene contains both `Matsui` and `homer` (see Figure 1). Some of them may show only either of them, and there may even be a *filler shot* showing none of them but simply a glimpse of spectators. It should be noted that such *filler shots* are even more common in edited motion pictures. Therefore, the user who issues this query in the first place does not necessarily expect a video interval that shows `Matsui` and/or `homer` in each shot throughout its play. Intuitively, the above query can be interpreted as: "retrieve a contiguous video interval in which each keyword `Matsui` and `homer` emerges somewhere at least once", or a sequence of shots in which each keyword emerges in at least one shot.

Producing appropriate intervals from the actual video data to answer even such a simple query asks for new interval operations. Our goal is to develop such a set of operations that can compute answers to queries on a video database. The database simply contains video shots that are fragmentarily indexed by descriptive keywords.

2.2. VIDEO INTERVAL

A video interval[3] is a stream of contiguous frames and is uniquely defined by a pair of frame numbers — starting frame number and ending frame number which are represented by f_s and f_e respectively. We write $f_s(i)$ and $f_e(i)$ to indicate the starting frame number and the ending frame number respectively of an interval i, where $f_s < f_e$. An interval is denoted by i$[f_s, f_e]$, or simply by i whenever $[f_s, f_e]$ can be omitted.

A video interval is indexed by a set of keywords $\{k_1, k_2, \ldots, k_n\}$. To a query keyword k, *valid shots* are the ones which are associated with the keyword k.

2.3. GLUE OPERATIONS

Here, we present formal definitions of our interval operations required to compute all possible answer intervals to a keywords-based query. Their properties will clearly reflect the query semantics that we informally stated above.

2.3.1 Interval glue.

Given two video intervals x and y, the operation *interval glue* ($\oplus$) on these two intervals yields a single interval i as follows:

$$x \oplus y = i[f_s, f_e] \quad \text{where} \quad \begin{aligned} f_s &= min(f_s(x), f_s(y)) \quad \text{and} \\ f_e &= max(f_e(x), f_e(y)) \end{aligned}$$

[3]A shot is also a video interval, but not necessarily a meaningful interval. Interval is used instead wherever no distinction is necessary.

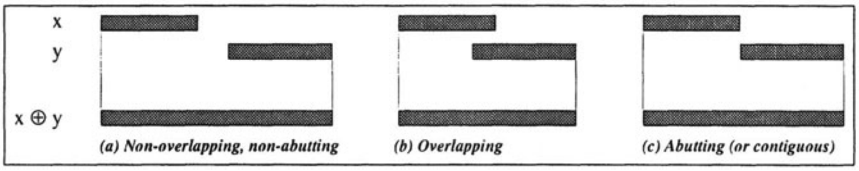
x
y
x ⊕ y
(a) Non-overlapping, non-abutting
(b) Overlapping
(c) Abutting (or contiguous)

Figure 2 Interval Glue between x and y

Thus supposing there are two video intervals $x[100, 160]$ and $y[180, 210]$ then $x \oplus y = i[100, 210]$ (Figure 2).

The basic idea of this operation is that even if two input intervals are not abutted, the resulting interval will be contiguous. This operation is important because of our assumption that keywords are associated with only fragmentary intervals. Moreover, it is clear from the example above that related shots are often separated by *filler shots*, which may not be indexed by the keywords that appear in the query. Readers must note that the *concatenation* operation between two non-contiguous intervals, which is used in many existing researches, does not produce a contiguous interval thus losing the original context of the data. Our interval glue operation is more appropriate than the concatenation operation in the context of video interval composition.

Some important algebraic properties of *interval glue* operation are:

- Commutativity: $x \oplus y = y \oplus x$ (by the definition of *interval glue*)
- Associativity: $(x \oplus y) \oplus z = x \oplus (y \oplus z)$. Hence, hereafter we write $(x \oplus y) \oplus z = x \oplus y \oplus z$ (proof omitted for space reason)
- Idempotence: $x \oplus x = x$ (by the definition of *interval glue*)

2.3.2 Pairwise glue. This is the set-variant of *interval glue* operation. Given two sets of video intervals X and Y, the operation *pairwise glue* ($\bigoplus$) returns a set of video intervals yielded by pairwise *interval glue* operation ($\oplus$) between the elements of the two input sets. (See Figure 3)

$$X \bigoplus Y = \{x \oplus y \mid x \in X \text{ and } y \in Y\}$$

For a given set of intervals $X = \{i_1, \ldots, i_n\}(n \geq 1)$, $\bigoplus(X)$ denotes $i_1 \oplus \ldots \oplus i_n$. This notation will be used in the definition of *powerset glue* operation below.

The *pairwise glue* has the following algebraic properties.

- Commutativity: $X \bigoplus Y = Y \bigoplus X$ (by the definition of *pairwise glue*)
- Associativity: $(X \bigoplus Y) \bigoplus Z = X \bigoplus (Y \bigoplus Z)$ (proof omitted)

However, the following example clarifies that the idempotence law does not hold. Suppose, $X = \{i_1, i_2\}$. Then $(X \bigoplus X) = \{(i_1 \oplus i_1), (i_2 \oplus i_2), (i_1 \oplus i_2)\} = \{i_1, i_2, (i_1 \oplus i_2)\}$ which is different from X. Hence, $X \neq (X \bigoplus X)$.

It should also be noted that $(X \bigoplus X) = (X \bigoplus X \bigoplus X) = (X \bigoplus \ldots \bigoplus X)$. This property is used in transformation of the *powerset glue* operation, which will be explained in the following subsection.

Supposing S_{k1} and S_{k2} are two sets of shots associated with the keywords k_1 and k_2 respectively. Assuming that each shot does not contain both the keywords, then $S_{k1} \bigoplus S_{k2}$ represents a set of intervals such that for any keyword k_1, k_2, each interval will contain exactly one shot.

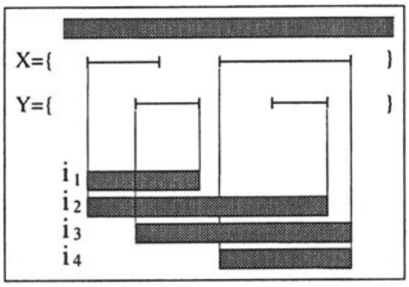

Figure 3 Pairwise Glue Operation between X and Y

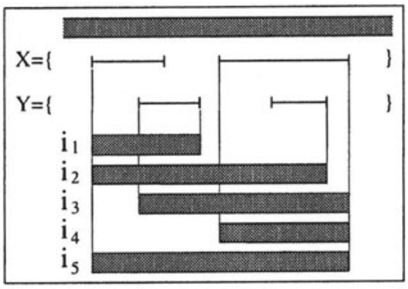

Figure 4 Powerset Glue Operation between X and Y

2.3.3 Powerset glue. Given two sets of video intervals X and Y, the operation *powerset glue* ($\bigotimes$) returns a set of video intervals. These intervals are yielded by applying *interval glue* operation ($\oplus$) to an arbitrary number (but not 0) of elements in X and Y.

$$\begin{aligned} X \bigotimes Y = \; & \{\bigoplus(X' \cup Y') \mid X' \subseteq X, Y' \subseteq Y, \\ & X' \neq \emptyset \text{ and } Y' \neq \emptyset\} \end{aligned}$$

In Figure 4, intervals $i_1, \ldots, i_4$ are yielded by applying *interval glue* operation on pairs of intervals; each pair consisting of one element from both X and Y. The interval i_5 is yielded by the same operation on a set of intervals, the set consisting of two elements from X and either one or two elements from Y. We may also consider intervals produced by taking two elements from Y and one from X, but the results will be the same as i_2 and i_3.

The *powerset glue* operation between X and Y is formulated as:

$$\begin{aligned} X \bigotimes Y \; = \; & (X \bigoplus Y) \cup (X \bigoplus X \bigoplus Y) \cup \\ & (X \bigoplus Y \bigoplus Y) \cup (X \bigoplus X \bigoplus X \bigoplus Y) \cup \\ & (X \bigoplus X \bigoplus Y \bigoplus Y) \cup (X \bigoplus Y \bigoplus Y \bigoplus Y) \cup \\ & \vdots \end{aligned}$$

The primary difference between the *pairwise glue* ($\bigoplus$) and *powerset glue* ($\bigotimes$) operations with two sets of intervals is that the former considers only one interval from each set whereas the latter considers one or more than one interval from each set. The *powerset glue* is the operation that we actually use for computing answer intervals to a query. It should be noted that the *pairwise glue* operation is not enough to compute all the possible answer intervals, since it considers only one interval from each set of valid shots. However, *powerset glue* is able to compute each possible answer interval, no matter how many valid shots are required to make up such an interval.

Again, supposing S_{k1} and S_{k2} are two sets of shots associated with the keywords k_1 and k_2 respectively. Then $S_{k1} \bigotimes S_{k2}$ represents a set of intervals such that for any keyword k_1, k_2, each interval will contain one or more than one valid shots.

It is obvious that the definition of *powerset glue* operation is complex. It will take an enormous amount of computation, especially when the number of intervals contained in X or Y is large. However, the original definition of *powerset glue* can be transformed into a simpler and more efficient expression which involves only three *pairwise glue* operations. This is one big contribution of this paper. The following theorem states the newly transformed expression.

Theorem 1 *For any intervals sets X and Y, the following equation holds.*

$$X \bigotimes Y = (X \bigoplus X) \bigoplus (Y \bigoplus Y)$$

Proof: *See Appendix.*

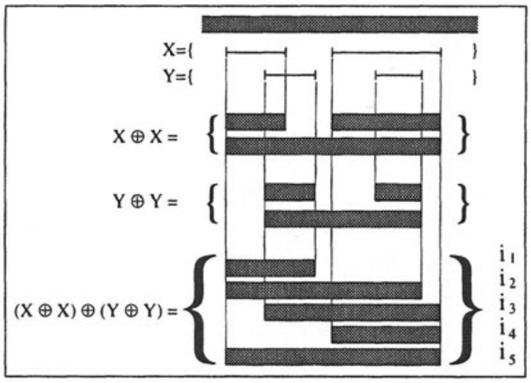

Figure 5 Transformation of Powerset Glue definition into a simpler expression

In Figure 5, the operations $(X \bigoplus X)$ and $(Y \bigoplus Y)$ produce two sets each consisting three intervals. Further *pairwise glue* operation on these two sets yields five intervals $i_1, \ldots, i_5$ which is the same desired set of results (See Figure 4).

2.4. ANSWER TO A QUERY

A query Q is represented by a set of keywords $\{k_1, \ldots, k_n\}$, which is interpreted as "retrieve contiguous video intervals in which each keyword $k_1, \ldots k_n$ appear somewhere. The answer to this query will be a set of intervals — each interval containing at least one valid shot for every query keyword $k_1, \ldots, k_n$. Formally, it is represented as:

$$Ans(Q) = S_{k1} \otimes \ldots \otimes S_{kn},$$

where $S_{k1} \ldots S_{kn}$ are the sets of valid shots for the query keywords $k_1 \ldots k_n$ respectively.

When a query is issued, valid shots are first selected for each query term. Then the operation powerset glue which we derived in Theorem 1 is applied to these sets of valid shots to produce answer intervals. If the query consists of more than two query terms, powerset glue is performed first between any two arbitrary sets of valid shots, which produces an intermediate set of intervals. The powerset glue operation is recursively performed between the intermediate set and each remaining set of valid shots until no sets of valid shots are left.

Consider again the query $Q = \{\texttt{Matsui}, \texttt{homer}\}$ that we mentioned in Section 2.1. Suppose, we have the following shots indexed by the corresponding keywords (See Figure 1 in Introduction).

- □ $i_3[1350, 1429] \rightarrow \{\texttt{Matsui}\}$
- □ $i_4[1430, 1469] \rightarrow \{\texttt{Yamada}, \texttt{straight}, \texttt{Matsui}, \texttt{hit}\}$
- □ $i_5[1470, 1499] \rightarrow \{\texttt{homer}\}$
- □ $i_6[1500, 1529] \rightarrow \{\}$
- □ $i_7[1530, 1550] \rightarrow \{\texttt{Matsui}, \texttt{home plate}\}$

Supposing A_1 and A_2 are the sets of valids shots for `Matsui` and `homer` respectively, then $A_1 = \{i_3, i_4, i_7\}$ and $A_2 = \{i_5\}$. Then the answer to this query will be:

$$\begin{aligned} A &= \{i_{37}[1350, 1550], i_{35}[1350, 1470], i_{45}[1430, 1470], \\ &\quad i_{47}[1430, 1550], i_{57}[1470, 1550]\} \end{aligned}$$

Up to this point we have not considered any lengthy intervals that glue operations produce if any temporally far-off shot associated with the keyword either `Matsui` or `homer` is available within the video stream. The following section explains how to avoid considering such intervals by using filter techniques.

3. FILTERING TECHNIQUES

One problem we encountered while defining our glue operations is the existence of 'noise' within an answer interval. Intuitively, 'noise' is a single shot or a sequence of shots that cannot be matched with any of the terms appearing in a query. Existing interval operations such as *intersection*, *union*, *concatenation* defined in (Oomoto and Tanaka, 1993) (Weiss et al., 1995) (Hwang and

Subrahmanian, 1996) (Hjelsvold et al., 1996) do not offer adequate support for computing answer intervals containing 'noise'. It is mainly because keywords-based queries are generally interpreted as — "In an answer interval,

- ☐ all query keywords must appear throughout it (AND-type),
- ☐ at least one of the query keywords must appear throughout it (OR-type)".

In the actual video, however, as we mentioned above, a meaningful interval ($s3$ to $s7$ in Figure 6) may contain a short sequence that act only as a *filler* shot. If we strictly interpret a video query as an AND/OR-type query, an interval, even if it contains only a filler shot, will not be included in the query result. In practical applications of video databases, such a strict interpretation of the query makes little sense.

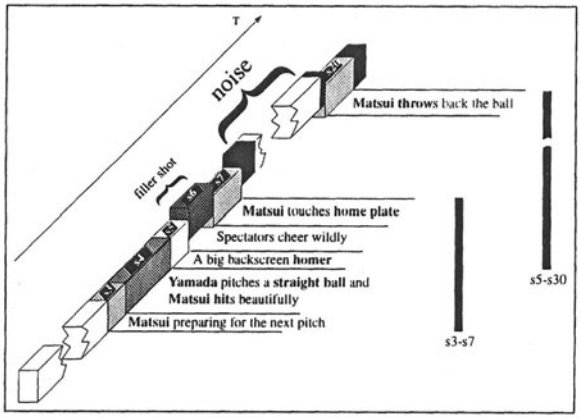
T
noise
filler shot
s3
s5
s6
s7
s21
Matsui throws back the ball
Matsui touches home plate
Spectators cheer wildly
A big backscreen homer
Yamada pitches a straight ball and Matsui hits beautifully
Matsui preparing for the next pitch
s3-s7
s5-s30

Figure 6 A possible meaningful interval (s3-s7) containing a *filler* shot (s6) and an irrelevant interval (s5-s30) containing *noisy* shots

3.1. IRRELEVANT INTERVALS

Given a query, our initial goal is to compute as many considerable answer intervals as possible. While we have achieved this goal successfully by defining the glue operations, we have also created a new problem for ourselves. Our assumption is that video streams are segmented only on the basis of physical shot boundaries and that we have no prior knowledge about the boundaries of meaningful intervals. As a result, under the pretext of composing answer intervals, the glue operations will go on considering longer intervals as far as valid shots associated with the query keywords are found for gluing together. In the worst case, if two query keywords happen to fall at the beginning and at the final shot of the video stream, there is always a possibility of returning the full one hour video as a valid answer. This will ultimately result in a large set of unwanted intervals. For example in Figure 6, an answer interval represented by $s5$ to $s30$ is one such unwanted interval to a query $\{\texttt{Matsui}, \texttt{homer}\}$, considering that the keyword `Mastui` is associated with a distantly separated shot

$s30$ in the same video stream. An answer set should exclude as many unwanted intervals as possible if we are to achieve an effective query mechanism.

3.2. INTERVAL FILTERS

Here, we discuss our filtering techniques for discarding intervals that can be thought of irrelevant to a given query. Intuitively, we provide a filter function that takes a set of video intervals as input, and returns the subset that meets some necessary conditions for a query match. As far as query processing is concerned, excluding irrelevant intervals only after computing all the possible answer intervals will have very little impact on query efficiency. A great amount of computation can be reduced if, at the initial stage of query processing, we can discard intervals that will eventually become useless for computing relevant answer intervals. In order to do so, the filter function must possess certain algebraic properties which we will describe below. Given that a filter function possesses such properties, it can be safely integrated with the glue operations.

The simplified definition of *powerset glue* that we stated in Theorem 1 must be well-supported by the filter function. Formally, for any two interval sets X and Y, the following must hold:

$$\mathrm{F}(X \bigotimes Y) = \mathrm{F}(\mathrm{F}(X \bigoplus X) \bigoplus \mathrm{F}(Y \bigoplus Y))$$

where F is an *interval filter* function.

If an arbitrary filter function holds this property, it will ensure that all the desired intervals will be included in the answer set. However, in order to exclude the irrelevant intervals at the initial stage of processing, the followings also must hold.

$$\mathrm{F}(x \oplus y) = \mathrm{F}(\mathrm{F}(x) \oplus \mathrm{F}(y)) \text{ and}$$
$$\mathrm{F}(X \bigoplus Y) = \mathrm{F}(\mathrm{F}(X) \bigoplus \mathrm{F}(Y)),$$

where F is an *interval filter* function.

In the following sections, we describe two interval filters – *time-window* and *maximal noise-width* that are practically applicable to video queries. We will also show that the proposed *glue operations* can be well-integrated with these filters.

3.2.1 Time-Window Filter.

A video interval has a temporal duration that is generally expressed either in temporal unit such as seconds, minutes or in number of frames. It is natural for a query issuer to assume that meaningful intervals fall within a certain *time-window* such as 40 seconds or 1200 frames. Users should be able to specify such a *time-window* filter so that any answer interval longer than the specified duration will be filtered out from the response set. Below, we will show that *time-window* filter not only excludes unwanted intervals from the answer set but also reduces the number of candidate intervals to be considered even before composing the answer intervals.

Suppose $\mid i \mid$ denotes the temporal duration of an interval, i.e. $\mid i \mid = f_e(i) - f_s(i)$. We define a *time-window* filter $\mathtt{F_w}$ as a mapping from a set of intervals X to a subset of X. We first define $\mathtt{F_w}$ for each interval i in X as:

$$\mathtt{F_w}(i) = \begin{cases} i, & \text{if } \mid i \mid \le w; \\ \text{undefined}, & \text{otherwise} \end{cases}$$

where w is a specified *time-window*.

For any interval set X, we extend this definition to the following:
$\mathtt{F_w}(X) = \{i \mid i \in X \text{ and } \mid i \mid \le w\}$.

3.2.2 Maximal Noise-Width Filter. As mentioned above, while a short 'noise' that act only as *filler* shots within a meaningful interval are significant for its semantic continuity, a long 'noise' can simply be thought of as a sequence of shots that splits off intervals which are temporally far-off. Here, we give the formal definition of a 'noise' and provide a new interval filter for discarding intervals containing a long 'noise'.

For a keyword k, we define 'noise' as a set of intervals in which the keyword k does not appear. Supposing X is the set of all the intervals where the keyword k appears and U is the set of all the subintervals of the video data in the database. Then, for the keyword k, noise is computed as:

$$\begin{aligned} &Noise(k) = \max(\overline{X}) \quad \text{where} \\ &\quad \overline{X} = \{i \in U \mid (\forall i'(\neq i) \in X)(i' \cap i = \emptyset)\} \quad \text{and} \\ &\quad \max(Z) = \{i \mid i \in Z \quad \text{and} \\ &\quad\quad (\forall i'(i' \neq i) \in Z)(i \supseteq i' \quad \text{or} \quad i \cap i' = \emptyset)\} \end{aligned}$$

For a set of keywords $K = \{k_1, k_2, \ldots, k_n\}$, 'noise' is defined as a set of those intervals in which none of keyword $\{k_1, k_2, \ldots, k_n\}$ appear at all. In order to compute the 'noise' for a set of keywords, we need to provide the usual definition of interval intersection operation and its set-variant.

Given two video intervals i_1 and i_2, the operation *interval intersection* ($\odot$) on these two video intervals yields a single video interval i as follows:

$$\begin{aligned} i_1 \odot i_2 = i[f_s, f_e] \quad \text{where} \quad & f_s < f_e && \text{and} \\ & f_s = max(f_s(i_1), f_s(i_2)) && \text{and} \\ & f_e = min(f_e(i_1), f_e(i_2)) \end{aligned}$$

Thus supposing there are two video intervals $i_1[10, 20]$ and $i_2[15, 40]$ then $i_1 \odot i_2 = i[15, 20]$.

The *interval set intersection* ($\bigodot$) operation on two sets of video intervals X and Y returns a set of video intervals constituting the pairwise intersection ($\odot$)

between the elements of the two input sets.

$$X \bigodot Y = \{x \odot y \mid x \in X \quad \text{and} \quad y \in Y\}$$

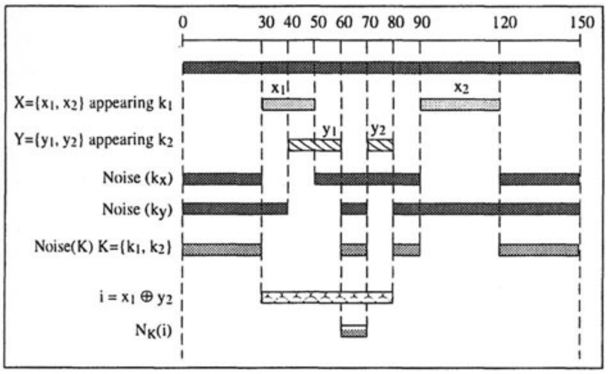

Figure 7 Definition of 'Noise'

Now, supposing $X_1, X_2, \ldots, X_n$ are the sets of intervals associated with the the keywords $k_1, k_2, \ldots, k_n$ respectively, then the 'noise' for such a set of keywords is defined as:

$$\begin{aligned} Noise(K) &= Noise(k_1) \bigodot \ldots \bigodot Noise(k_n) \\ &= \max(\overline{X_1}) \bigodot \ldots \bigodot \max(\overline{X_n}) \end{aligned}$$

As it is obvious that, to a given query consisting of a set of keywords, 'noise' can be easily computed by applying these definitions. For example, in Figure 7, $X = \{x_1[30,50], x_2[90,120]\}$ is a set of intervals associated with the keyword k_x. Similarly, $Y = \{y_1[40,60], y_2[70,80]\}$ is another set of intervals associated with the keyword k_y. 'Noise' for each k_x and k_y can be computed by $Noise(k_x)$ and $Noise(k_y)$ respectively as shown in the figure. By performing *interval intersection* operation on these two sets of intervals, we get the set of *noisy* intervals (containing 'noise') $Noise(K)$ for the combined set of keywords $K = \{k_x, k_y\}$.

For a set of keywords K, we can easily compute 'noise' contained in an arbitrary interval i by applying the following.

$$N_K(i) = \{i\} \bigodot Noise(K)$$

For a set of keywords K, Figure 7 shows the 'noise' $N_K(i)$ contained in an interval $i = x_1 \oplus y_2$, which is an intersection operation between the interval i and each interval of the set $Noise(K)$.

Based on this definition of 'noise', we now define a new filter. First, we specify N as the *maximal noise-width* that can be allowed in an answer interval. N is expressed in terms of temporal duration such as seconds or number of frames. Then for a query expressed by a set of keywords K, we define a *maximal noise-width* filter $\mathrm{F}_{\mathrm{N,K}}$ as a mapping from a set of intervals X to a subset of X. We first define $\mathrm{F}_{\mathrm{N,K}}$ for each interval i in X as:

$$\mathrm{F}_{\mathrm{N,K}}(i) = \begin{cases} i, & \max\{\,|\,i'\,|\;\;|\,i' \in N_K(i)\} \leq N; \\ \text{undefined}, & \text{otherwise} \end{cases}$$

where N is a specified *maximal noise-width.*

Again, for any interval set X, we extend this definition into the following:

$$\mathrm{F}_{\mathrm{N,K}}(X) = \{i \mid i \in X \text{ and } \max\{\,|\,i'\,|\;\;|\,i' \in N_K(i)\} \leq N\}$$

3.2.3 Glue Operations and Filter Functions. As stated above, in order to enable us to incorporate a filter function easily in our query mechanism, it must possess certain algebraic properties. The following theorem states the proposed filter functions can indeed be integrated with *powerset glue* operation for computing desired answer intervals to video queries.

Theorem 2 *For any interval sets X and Y, the following expression holds:*

$$\mathrm{F}(X \bigotimes Y) = \mathrm{F}(\mathrm{F}(X \bigoplus X) \bigoplus \mathrm{F}(Y \bigoplus Y))$$

where F *is either a* time-window $(\mathrm{F}_{\mathrm{w}})$ *or a* maximal noise-width $(\mathrm{F}_{\mathrm{N,K}})$ *filter.*
Proof: *See Appendix.*

The following two lemmas present some fundamental properties which provide key leverage for deriving Theorem 2.

Lemma 1 *For any two intervals x and y, the* interval glue *operation has the following property.*

$$\mathrm{F}(x \oplus y) = \mathrm{F}(\mathrm{F}(x) \oplus \mathrm{F}(y)),$$

where F *is either a* time-window $(\mathrm{F}_{\mathrm{w}})$ *or a* maximal noise-width $(\mathrm{F}_{\mathrm{N,K}})$ *filter.*
Proof: *See Appendix.*

Lemma 2 *For any two sets X and Y whose elements are indexed video intervals, the* pairwise glue *operation has the following property.*

$$\mathrm{F}(X \bigoplus Y) = \mathrm{F}(\mathrm{F}(X) \bigoplus \mathrm{F}(Y)),$$

where F *is either a* time-window $(\mathrm{F}_{\mathrm{w}})$ *or a* maximal noise-width $(\mathrm{F}_{\mathrm{N,K}})$ *filter.*
Proof: *See Appendix.*

Despite its computational complexity as compared to *time-window*, *maximal noise-width* filter is one natural way of reducing the over-populated answer set. In practical applications, unlike *time-window* filter, a high value for *maximal noise-width* filter can be assumed implicitly by the system. It is because we are concerned to filter out only those intervals which contain considerable length of 'noise'.

Theorem 2 is applied to compute desired answer intervals to a query while filtering out irrelevant intervals from the answer set. In Lemma 2, we showed that $\mathtt{F}(X \bigoplus X) = \mathtt{F}(\mathtt{F}(X) \bigoplus \mathtt{F}(X))$. One big advantage of this property is that the number of candidate intervals to be considered are greatly reduced even at the initial stage of query processing by applying mapping functions $\mathtt{F}(X)$ and $\mathtt{F}(Y)$ with the *time-window* filter. Consequently, an efficient query mechanism can be achieved by integrating filter functions with our video queries.

4. EXPERIMENTAL EVALUATION

We carried out an experiment to evaluate two performance gains. The first performance gain we anticipated was as an effect due to the transformation of the *powerset glue* operation into three *pairwise glue* operations. Theoretically, if $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ are two sets of intervals, the time complexity of powerset glue operation $X \bigotimes Y$ becomes 2^{m+n}. However, the time complexity of the transformed powerset glue operation $(X \bigoplus X) \bigoplus (Y \bigoplus Y)$ is simply n^2m^2.

Naturally, we observed a very large performance gain from transformed definition of powerset glue operation. Given any query, it showed that the ratio of the time taken to compose the answer intervals without transformation to the time taken after the transformation was very large.

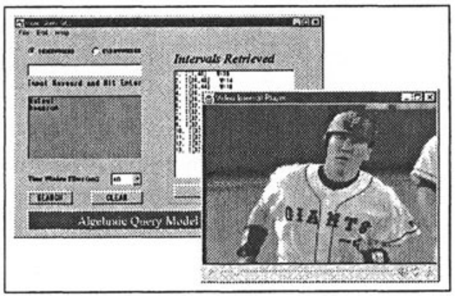

Figure 8 A prototype system implemented in Java. A five minutes long MPEG video data of a live baseball match was used as experimental data. There were 31 shots fragmentarily indexed by 13 unique keywords.

The second performance gain expected was as a result due to the application of the filter function to the queries. Although there was no reduction in time complexity of powerset glue operations, we could expect a considerable performance gain by reducing the candidate intervals before performing powerset glue operations on them. The experiment showed that the ratio of the time taken to compose the answer intervals without filter functions to the time taken after applying the filter functions was also substantially large. However, we observed that this type of performance gain totally depended upon the number of candidate intervals that could be discarded at the initial stage. Further experiments are necessary to establish a ground theory for benchmarking our query mechanism.

5. RELATED WORK

There is now growing interest in querying the large resources of digital video data. Allen's work on temporal intervals (Allen, 1983) laid the foundation for many researches concerned with time intervals (Little and Ghafoor, 1993) (Lorentzos and Mitsopoulos, 1997). He showed that there are 13 distinct temporal relationships that can exist between two arbitrary time intervals. Some researches on video databases have been greatly influenced by Allen's temporal model. In (Little and Ghafoor, 1993), temporal-interval-based models have been presented for time-dependent multimedia data such as video. However, our work is orthogonal with the ones based on temporal logic. We were rather interested in the semantics of a keywords-based video query and focused our work on the operations required for synthesizing new intervals in order to answer such queries. As we saw in the example above, in any meaningful video interval, the same keyword may emerge for an indefinite number of times in no particular order. Further investigation is required if we are to integrate the query condition specifications based on their temporal relationships into the current query mechanism[4].

Work by (Oomoto and Tanaka, 1993) (Weiss et al., 1995) (Hwang and Subrahmanian, 1996) (Hjelsvold et al., 1996) put emphasis on annotation model rather than on query model. As stated above, no matter what the annotation models are, defining all possible interval answers in advance in the indexing scheme itself is not feasible with the present technology. Although they have defined several interval operations to compute new intervals from existing intervals, they all lack the kind of algebraic operations required for our purpose.

Informedia project (Wactlar et al., 1996) uses the full text information retrieval system based on well-known technique of *tf/idf* (term frequency/inverse document frequency) for keywords-based queries on video databases. However, it also presumes the answer intervals in terms of the granularity of the indexed units, or what is called *video paragraphs*.

The idea of assuming two visually similar shots as components of different story units on the basis of *temporal locality* was first proposed in (Yeung et al., 1996). They have applied this idea to clustering video shots that are visually similar and temporally local. The basic assumption is that if two visually similar shots do not fall within a certain time window, they can be considered as the shots from different story units. Our *maximal noise-width* filter is somewhat inspired by this concept.

[4]The result of our preliminary work on an algebraic video query model is going to be published in the journal of IEEE Transactions on Knowledge and Data Engineering (Pradhan et al., 2000). Our recent work deals with temporal filters which allow us to specify temporal relationships between keywords in a video query.

6. CONCLUSIONS

In any video database system with fragmentarily indexed intervals, end-users often have difficulty in retrieving the intervals that they desire to see. This is because the intervals they are hoping to find may not have not been defined as answer units in the database. In such cases, intervals need to be computed dynamically from the indexed units that currently exist in the database.

Algebraic operations such as union, intersection, concatenation does not always produce the desired answers since these operations do not consider the presence of 'noise' in a meaningful interval, which is so natural and common in video data. In order to compute interval answers to a video query represented by a set of keywords, we defined a set of interval operations, called *glue operations*, which is the main contribution of this paper. Given a query, these operations enable us compute all the possible boundaries for interval answers from a set of stored video shots.

We also proposed a set of *interval filters* that can be incorporated in video queries so that irrelevant answer intervals are excluded from the answer set. We then investigated the characteristics of each filter and found that the filters can be directly integrated into the proposed *powerset glue* operation. As a result, a considerable number of irrelevant intervals can be removed at the initial stage of query processing, which has led to a groundwork for an efficient query mechanism.

Acknowledgment

We would like to thank all the referees for their invaluable comments on the manuscript.

References

Allen, J. (1983). "Maintaining Knowledge about Temporal Intervals". *Communications of the ACM*, 26(11):832–843.

Arman, F., Hsu, A., and Chiu, M. (1993). "Image Processing on Compressed Data for Large Video Databases". In *Proc. of ACM Multimedia Conferences*, pages 267–272.

Hjelsvold, R., Midtstraum, R., and Sandst, O. (1996). "Searching and Browsing a Shared Video Database". Multimedia Database Systems. Design and Implementation Strategies. chapter 4. Kluwer Academic Publishers.

Hwang, E. and Subrahmanian, V. (1996). "Querying Video Libraries". *Journal of Visual Communications and Image Representation*, 7(1):44–60.

Kobla, V., Doermann, D., and Faloutsos, C. (1997). "VideoTrails: Representing and Visualizing Structure in Video Sequences". In *Proc. of ACM Multimedia*, pages 335–346.

Little, T. and Ghafoor, A. (1993). "Interval-Based Conceptual Models for Time-Dependent Multimedia Data". *IEEE Transactions on Knowledge and Data Engineering*, 5(4):551–563.

Lorentzos, N. and Mitsopoulos, Y. (1997). "SQL Extension for Intervals Data". *IEEE Transactions on Knowledge and Data Engineering*, 9(3):480–499.

Oomoto, E. and Tanaka, K. (1993). "OVID: Design and Implementation of a Video-Object Database System". *IEEE Transactions on Knowledge and Data Engineering*, 5(4):629–643.

Pradhan, S., Tajima, K., and Tanaka, K. (2000). "A Query Model to Synthesize Answer Intervals from Indexd Video Units". *Accepted for publication in the forthcoming issue of IEEE Transactions on Knowledge and Data Engineering.*

Smith, T. A. and Davenport, G. (1992). "The Stratification System: A Design Environment for Random Access Video". In *Proc. 3rd Int'l Workshop on Network and Operating System Support for Digital Audio and Video*, pages 250–261.

Wactlar, H., Kanade, T., Smith, M., and Stevens, S. (1996). "Intelligent Access to Digital Video: Informedia Project". *IEEE Computer*, 29(5):46–52.

Weiss, R., Duda, A., and Gifford, D. (1995). "Composition and Search with a Video Algebra". *IEEE MultiMedia*, 2(1):12–25.

Yeung, M., Yeo, B., and Liu, B. (1996). "Extracting Story Units from Long Programs for Video Browsing and Navigation". In *International Conference on Multimedia Computing and Systems*, pages 296–305.

Zhang, H., Kankanhalli, A., and Smoliar, S. (1993). "Automatic Parsing of Full-Motion Video". *Multimedia Systems*, 1:10–28.

Zhang, H., Low, C., Smoliar, S., and Wu, J. (1995). "Video Parsing and Retrieval and Browsing: An Integrated and Content-based Solution". In *Multimedia 95 Proceedings*, pages 15–24.

Biographies

Sujeet Pradhan received a BE in Mechanical Engineering from the University of Rajasthan, India in 1988, an MS in Instrumentation Engineering in 1995 and a Ph.D. in Intelligence Science in 1999 from Kobe University, Japan. Since 1999 May, he is a lecturer of the Department of Computer Science and Mathematics at Kurashiki University of Science and the Arts, Japan. He was a Colombo Plan scholar during 1984–1988 and a Mombusho scholar during 1995–1997. A JSPS (Japan Society for the Promotion of Science) Research Fellow during the period between 1997 and 1999, his research interests include video databases, multimedia authoring, prototype-based languages and semi-structured databases. Dr. Pradhan is a member of Information Processing Society of Japan.

Takashi Sogo is currently a graduate student of the Department of Computer and Systems Engineering at Kobe University. He received a B.E. in Computer Science from the Kobe University in 1998. His research interests include video databases and non-traditional databases.

Keishi Tajima received the B.Sc., M.Sc., and D.Sc. degrees in information science from the University of Tokyo, in 1991, 1993, and 1996 respectively. In 1996, he joined the Department of Computer and Systems Engineering at Kobe University, as a research associate. His research interests include datamodel for non-traditional database applications, database programming languages, and security in database systems.

Katsumi Tanaka received the B.S., M.S., and Ph.D degrees in information science from Kyoto University, in 1974, 1976, and 1981 respectively. In 1986, he joined the Department of Instrumentation Engineering at Kobe University, as an associate professor. Since 1994, he is a professor of the Department of Computer and Systems Engineering and since 1997, he is a professor of Division of Media and Computer Science of Graduate School of Science and Technology at Kobe University. His research interests include object-oriented databases, video databases, historical database models, and hypermedia systems. Dr. Tanaka is a member of the ACM, IEEE Computer Society and the Information Processing Society of Japan.

Appendix

Theorem 1: *For any intervals sets X and Y, the following expression holds.*

$$X \bigotimes Y = (X \bigoplus X) \bigoplus (Y \bigoplus Y)$$

Proof: Supposing $U = X \bigotimes Y$ and $V = (X \bigoplus X) \bigoplus (Y \bigoplus Y)$, then to prove $U = V$, we need to show that $U \supseteq V$ and $U \subseteq V$. First to prove, $U \supseteq V$, consider an arbitrary element $z \in V$. Then, there must be $x \in (X \bigoplus X)$ and $y \in (Y \bigoplus Y)$ such that $x \oplus y = z$. By the definition of *interval glue* ($\oplus$), the following four possible cases can be considered on the basis of what values z takes for its starting and ending frame.

I. $z \equiv z[f_s(x), f_e(y)]$ if $f_s(x) \leq f_s(y)$ and $f_e(y) \geq f_e(x)$

II. $z \equiv z[f_s(y), f_e(x)]$ if $f_s(y) \leq f_s(x)$ and $f_e(x) \geq f_e(y)$

III. $z \equiv z[f_s(x), f_e(x)]$ if $f_s(x) \leq f_s(y)$ and $f_e(x) \geq f_e(y)$

IV. $z \equiv z[f_s(y), f_e(y)]$ if $f_s(y) \leq f_s(x)$ and $f_e(y) \geq f_e(x)$

We will first consider Case I. Since $x \in (X \bigoplus X)$, there must be $x' \in X$ such that $f_s(x') = f_s(x)$ and $f_e(x') \leq f_e(x)$. Similarly, since $y \in (Y \bigoplus Y)$, there must be $y' \in Y$ such that $f_e(y') = f_e(y)$ and $f_s(y') \geq f_s(y)$. Now, since U is the *powerset glue* of X and Y, there must be $z' \in U$ such that $z' = x' \oplus y'$.

Since $f_s(x') = f_s(x)$, $f_s(x) \leq f_s(y)$ and $f_s(y') \geq f_s(y)$, we can conclude $f_s(x') \leq f_s(y')$. Similarly, since $f_e(y') = f_e(y)$, $f_e(y) \geq f_e(x)$ and $f_e(x') \leq f_e(x)$, we can conclude $f_e(y') \geq f_e(x')$. Therefore, $z' \equiv z'[f_s(x'), f_e(y')]$. Since $f_s(x') = f_s(x)$ and $f_e(y') = f_e(y)$, we can say that $z' = z$. Therefore, there must be $z \in U$ too. Refer to Fig.9.

Similar proof can be shown for Case II. The proof for Case III can be shown by considering two intervals in X and one in Y (Refer to the right hand side of Fig.9). Case IV can be proved in a similar manner. Hence, $U \supseteq V$.

Next we show that $U \subseteq V$. Again, consider an arbitrary element $z \in U$. We know, U is the product of *powerset glue* operation between X and Y. Hence, we can write $z = x \oplus y$ where x and y are yielded by performing *pairwise glue* operation on one or more elements of X and Y respectively. Here also, by the definition of *interval glue* ($\oplus$), four possible cases can be considered on the basis of what values z takes for its starting and ending frame.

I. $z \equiv z[f_s(x), f_e(y)]$ if $f_s(x) \leq f_s(y)$ and $f_e(y) \geq f_e(x)$

II. $z \equiv z[f_s(y), f_e(x)]$ if $f_s(y) \leq f_s(x)$ and $f_e(x) \geq f_e(y)$

III. $z \equiv z[f_s(x), f_e(x)]$ if $f_s(x) \leq f_s(y)$ and $f_e(x) \geq f_e(y)$

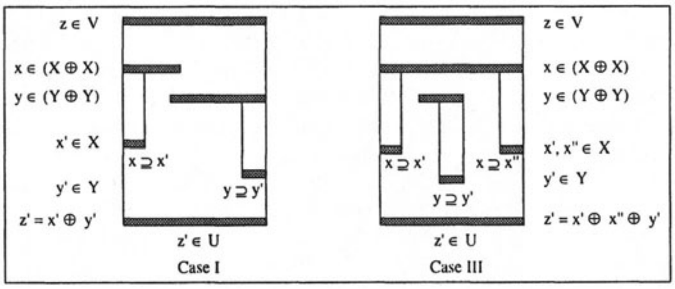

Figure 9 Illustration for the Proof of Theorem 1

IV. $z \equiv z[f_s(y), f_e(y)]$ if $f_s(y) \leq f_s(x)$ and $f_e(y) \geq f_e(x)$

We will first consider Case I. Since x is yielded by performing *pairwise glue* operation on one or more elements of X, there must be $x' \in X$ such that $f_s(x') = f_s(x)$ and $f_e(x') \leq f_e(x)$. Similarly, since y is yielded by performing *pairwise glue* operation on one or more elements of Y, there must be $y' \in Y$ such that $f_e(y') = f_e(y)$ and $f_e(y') \geq f_e(y)$.

Again, since $X \bigoplus X$ is a *pairwise glue* operation between X and X, any arbitrary element in X will definitely be in $X \bigoplus X$ too. Hence, $x' \in (X \bigoplus X)$. Similarly, $y' \in (Y \bigoplus Y)$. Therefore, there must be a $z' \in V$ such that $z' = x' \oplus y'$.

Since $f_s(x') = f_s(x)$, $f_s(x) \leq f_s(y)$ and $f_s(y') \geq f_s(y)$, we can conclude $f_s(x') \leq f_s(y')$. Similarly, since $f_e(y') = f_e(y)$, $f_e(y) \geq f_e(x)$ and $f_e(x') \leq f_e(x)$, we can conclude $f_e(y') \geq f_e(x')$. Therefore, $z' \equiv z'[f_s(x'), f_e(y')]$. Since $f_s(x') = f_s(x)$ and $f_e(y') = f_e(y)$, we can say that $z' = z$. Therefore, there must be $z \in V$ too.

Similar proof can be shown for Case II. The proof for Case III can be shown by considering two intervals in X and one in Y. Case IV can be proved in a similar manner. Hence, $U \subseteq V$. This completes the proof. □

Lemma 1: *For any two intervals x and y, the* interval glue *operation has the following property.*

$$\mathtt{F}(x \oplus y) = \mathtt{F}(\mathtt{F}(x) \oplus \mathtt{F}(y)),$$

where F *is either a* time-window ($\mathtt{F}_w$) *or a* maximal noise-width ($\mathtt{F}_{N,K}$) *filter.*

Proof: Let W denotes either the specified *time-window filter* width w or the specified maximal noise-width filter N. Also, let $\mid i \mid$ denotes the temporal duration of an interval i in the case of *time-window filter* and the maximal noise it contains in the case of *maximal noise-width filter*. The proof is obvious if $\mid x \mid \; \leq W$ and $\mid y \mid \; \leq W$. However, if $\mid x \mid \; > W$ or $\mid y \mid \; > W$, then $\mathtt{F}(\mathtt{F}(x) \oplus \mathtt{F}(y))$ = undefined. We know that $\mid (x \oplus y) \mid \; \geq \; \mid x \mid, \mid y \mid$. Hence, it is obvious that $\mid (x \oplus y) \mid \; > W$ if either $\mid x \mid \; > W$ or $\mid y \mid \; > W$. Since $\mathtt{F}(x \oplus y)$ = undefined when $\mid (x \oplus y) \mid \; > W$, we thus complete the proof. □

Lemma 2: *For any two sets X and Y whose elements are indexed video intervals, the* pairwise glue *operation has the following property.*

$$\mathtt{F}(X \bigoplus Y) = \mathtt{F}(\mathtt{F}(X) \bigoplus \mathtt{F}(Y)),$$

where F *is either a* time-window ($\mathtt{F}_w$) *or a* maximal noise-width ($\mathtt{F}_{N,K}$) *filter.*

Proof: Let U denotes $\mathtt{F}(X \bigoplus Y)$ and V denotes $\mathtt{F}(\mathtt{F}(X) \bigoplus \mathtt{F}(Y))$, then in order to prove that $U = V$, all we have to show is $U \supseteq V$ and $U \subseteq V$.

Consider any arbitrary $x \in \mathtt{F}(X)$. Naturally, $x \in X$. Therefore $X \supseteq \mathtt{F}(X)$. Similarly, $Y \supseteq \mathtt{F}(Y)$. It follows that $X \bigoplus Y \supseteq \mathtt{F}(X) \bigoplus \mathtt{F}(Y)$. Also, $\mathtt{F}(X \bigoplus Y) \supseteq \mathtt{F}(\mathtt{F}(X) \bigoplus \mathtt{F}(Y))$. Hence, $U \supseteq V$.

Next, consider an arbitrary element $z \in U$. Then $\mid z \mid \; \leq W$ and there must be $x \in X$ and $y \in Y$ such that $x \oplus y = z$. Since $\mid z \mid \; \leq W$ implies $\mid x \mid \; \leq W$, we can say that $x \in \mathtt{F}(X)$. Similarly, $y \in \mathtt{F}(Y)$. Also, since $\mid (x \oplus y) \mid \; \leq W$, $(x \oplus y) \in \mathtt{F}(\mathtt{F}(X) \bigoplus \mathtt{F}(Y))$. Therefore, $z \in \mathtt{F}(\mathtt{F}(X) \bigoplus \mathtt{F}(Y))$. Hence, $U \subseteq V$. The proof is now complete. □

Theorem 2: *For any interval sets X and Y, the following expression holds:*

$$\mathtt{F}(X \bigotimes Y) = \quad \mathtt{F}(\mathtt{F}(X \bigoplus X) \bigoplus \mathtt{F}(Y \bigoplus Y))$$

where F *is either a* time-window ($\mathtt{F}_w$) *or a* maximal noise-width ($\mathtt{F}_{N,K}$) *filter.*

Proof: By the definition of *powerset glue*, $X \bigotimes Y = (X \bigoplus X) \bigoplus (Y \bigoplus Y)$. By applying the filter F in both sides, $\mathtt{F}(X \bigotimes Y) = \mathtt{F}((X \bigoplus X) \bigoplus (Y \bigoplus Y))$. Supposing, $X \bigoplus X = X'$ and $Y \bigoplus Y = Y'$. According to the Lemma 2, $\mathtt{F}(X' \bigoplus Y') = \mathtt{F}(\mathtt{F}(X') \bigoplus \mathtt{F}(Y'))$. By substitution, $\mathtt{F}((X \bigoplus X) \bigoplus (Y \bigoplus Y)) = \mathtt{F}(\mathtt{F}(X \bigoplus X) \bigoplus \mathtt{F}(X \bigoplus X))$.

Hence, $\mathtt{F}(X \bigotimes Y) = \mathtt{F}(\mathtt{F}(X \bigoplus X) \bigoplus \mathtt{F}(X \bigoplus X))$. This proves the theorem. □