Video Algebra for Spatio-Temporal Reasoning of Iconic Videos Represented in 3D C-string

ANTHONY J. T. LEE, PING YU, HAN-PANG CHIU* AND HSIU-HUI LIN
Department of Information Management
National Taiwan University
Taipei, 106 Taiwan
**Department of Electrical Engineering and Computer Science*
Massachusetts Institute of Technology
Cambridge, MA 02139, U.S.A.

The video content management has attracted increasing attention in recent years. We have proposed a new spatio-temporal knowledge structure, called 3D C-string, to represent the spatio-temporal relations between the objects in a video and to keep track of the motions and size changes of the objects. In this paper, we propose a video algebra to infer the spatio-temporal relations between the objects in a video represented by the 3D C-string. The algebra contains four kinds of rules, namely, transitive, distributive, manipulation, and integration rules. By using those rules, all the binary relations between the objects in a video can be derived from a given 3D C-string. The algebra provides the theoretic basis for spatio-temporal reasoning and video query inference.

Keywords: video algebra, spatio-temporal relation, 3D C-string, video database, spatio-temporal reasoning

1. INTRODUCTION

With the advances in information technology, the amount of multimedia data captured, produced and stored is increasing rapidly, and hence, the need for organizing this data and accessing it from a vast amount of repositories has attracted much attention in recent years. Over the last decade, many image/video indexing and retrieval methods have been proposed. Bimbo *et al.* [1] developed a prototype system of image sequence retrieval, where video frames were processed and simple events were represented by spatial-temporal logic (STL). Chang *et al.* [2] proposed a model called VideoQ to analyze objects' motions in a video and provide an efficient and effective content-based retrieval mechanism. Naphade *et al.* [3] proposed a model to map low-level features to high-level semantics and to enforce spatio-temporal constraints in a factor graph framework. Ngo *et al.* [4] proposed a motion computation method based on a structure tensor formulation to encode visual patterns of spatio-temporal slices in a tensor histogram and to describe the motion trajectories of moving objects. VideoQA [5] allowed users to use short natural language questions with implicit constraints on contents to retrieve short precise news video summaries. Lo *et al.* [6] presented a framework for retrieving video sequences using successive modular operations on temporal similarity. Snoek *et al.* [7] developed the TIME framework to classify the semantic events in multimodal video documents.

Received November 18, 2004; revised August 18, 2005; accepted November 7, 2005.
Communicated by Ming-Syan Chen.

In image database management systems, one of the most important methods for retrieving the images is using the perception of objects and the spatial relations between the objects in the desired videos. To represent the spatial relations between the objects in an image, Chang *et al*. [8] proposed the concept of the 2D string in which the objects are projected onto the *x*- and *y*-axes to form two strings representing the relative positions of the projections in the *x*- and *y*-axes, respectively. This approach provides a natural way to construct iconic indexes for images and supports spatial reasoning and image queries. There are many follow-up approaches based on the concept of the 2D string including 2D G-string [9, 10], 2D C-string [11-13], 2D C^+-string [14], 2D RS-string [15], 2D C-Tree [16], unique-ID-based matrix [17], GPN matrix [18], virtual image [19], BP matrix [20], and 2D Z-string [21].

To represent the spatial and temporal relations between the objects in a symbolic video, many iconic indexing approaches, extended from the notion of the 2D string to represent the spatial and temporal relations between the objects in a video, have been proposed, for example, 2D B-string [22, 23], 2D C-Tree [24], 9DLT strings [25], 3D-list [26], and 3D C-string [27].

In the 3D C-string [27], we extended the concepts of the 2D C^+-string and proposed 3D C-string to represent the spatio-temporal relations between the objects and to record their motions and size changes. We also developed the string generation and video reconstruction algorithms for the 3D C-string. The string generated by the string generation algorithm is unique for a given video and the video reconstructed from a given 3D C-string is unique too. In comparison with the previously proposed approaches [22-26], there is one-to-one correspondence between strings and videos in the 3D C-string representation. This approach can provide us an easy and efficient way to retrieve, visualize and manipulate video objects in video database systems.

In this paper, we propose a video algebra for reasoning the spatio-temporal relations between the objects in a video represented by the 3D C-string. The algebra contains four kinds of rules, namely, transitive, distributive, manipulation, and integration rules. By using those rules, all the binary relations between the objects in a video can be derived from a given 3D C-string. The algebra can provide the theoretic basis for spatio-temporal reasoning and video query inference.

The rest of this paper is organized as follows. In section 2, a brief introduction of the 3D C-string approach is presented. The video algebra of reasoning the spatio-temporal relations between the objects in a video represented by the 3D C-string is discussed in section 3. In section 4, an application is presented to show the effectiveness of our proposed approach. Finally, concluding remarks are made in section 5.

2. 3D C-STRING APPROACH

In the knowledge structure of 3D C-string, we use the projections of objects to represent the spatial and temporal relations between the objects in a video. The objects in a video are projected onto the *x*-, *y*-, and *time*-axes to form three strings representing the relations and relative positions of the projections in the *x*-, *y*- and *time*-axes, respectively. These three strings are called *u*-, *v*- and *t*-strings. The projections of an object onto the *x*-, *y*- and *time*-axes are called *x*-, *y*-, and *time*-projections, respectively. In comparison with the 2D C-string and 2D C$^+$-string, the 3D C-string has one more dimension: *time* dimension. So the 3D C-string is different from the 2D C -string and 2D C$^+$-string that has only spatial relations between objects, it has spatial and temporal relations. Hence, it is required to keep track of the information about the motions and size changes of the objects in a video in the 3D C-string.

In the knowledge structure of 3D C-string, there are 13 relations for two one-dimensional intervals. For the x (or y) dimension, there are 13 spatial relations and its corresponding spatial operators have been presented in 2D C-string [11] as listed in Table 1, where $BB(P)$ and $EB(P)$ are the begin-bound (beginning point) and end-bound (ending point) of the x- (or y-) projection of object P. For example, in the x and y dimensions, $P < Q$ represents that the projection of object P is before that of object Q. In the *time* dimension, we also use those 7 operators to describe 13 possible relations between two *time*-projections. According to [11], we know that all the 13 operators except "/" can precisely (no ambiguity) represent the relations between two objects. To avoid using operator "/", we replace P / Q with $P\,]\,Q\,|\,Q$ in the 3D C-string representation where Q is called a cut object in $P\,]\,Q\,|\,Q$.

Table 1. The definitions of spatial operators in 2D C-string.

Notations	Conditions
$P < Q$	$EB(P) < BB(Q)$
$P = Q$	$BB(P) = BB(Q)$, $EB(P) = EB(Q)$
$P \mid Q$	$EB(P) = BB(Q)$
$P \% Q$	$BB(P) < BB(Q)$, $EB(P) > EB(Q)$
$P\,[\,Q$	$BB(P) = BB(Q)$, $EB(P) > EB(Q)$
$P\,]\,Q$	$BB(P) < BB(Q)$, $EB(P) = EB(Q)$
P / Q	$BB(P) < BB(Q) < EB(P) < EB(Q)$

In the knowledge structure of 3D C-string, an object is approximated by a minimum bounding rectangle (MBR) whose sides are parallel to the x-axis and y-axis. For each object, we keep track of its initial location and size. That is, we keep track of the location and size of an object in its starting frame. After keeping track of the initial location and size of an object, we record the information about its motions and size changes in the 3D C-string.

There is some metric information defined in the 3D C-string, which is listed as follows.

1. The size of object P: P_s denotes the size of the x- (y-, or *time*-) projection of object P, where $s = EB_x(P) - BB_x(P)$ ($s = EB_y(A) - BB_y(A)$, or $s = EB_{time}(P) - BB_{time}(P)$), where $BB_x(P)$ and $EB_x(P)$ are the x coordinates of the begin-bound and end-bound of P's x-projection, respectively.
2. The distance associated with operator "<": $P <_d Q$ denotes that the distance between the x- (y-, or *time*-) projection of object P and that of object Q is equal to d, where $d = BB_x(Q) - EB_x(P)$ ($d = BB_y(Q) - EB_y(P)$, or $d = BB_{time}(Q) - EB_{time}(P)$).
3. The distance associated with operator "%": $P \%_d Q$ denotes that the distance between

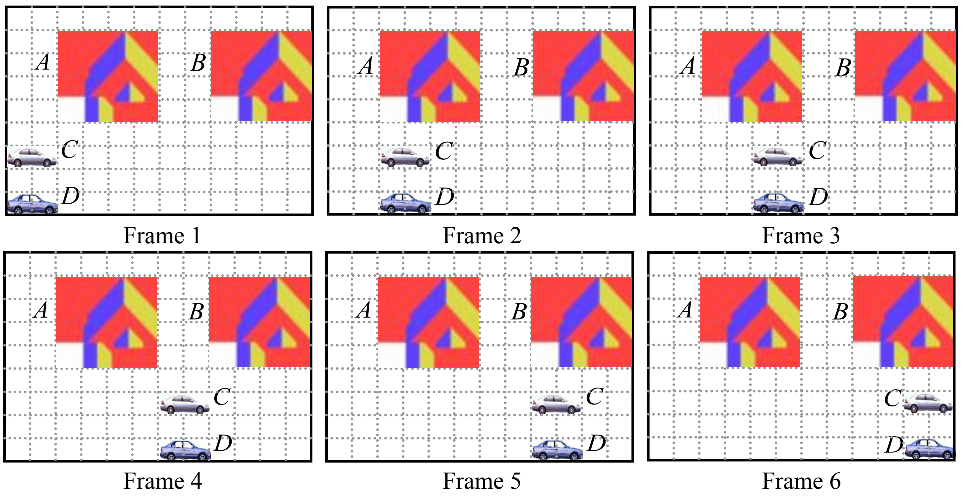
the *x*- (*y*-, or *time*-) projection of object P and that of object Q is equal to d, where $d = BB_x(Q) - BB_x(P)$ ($d = BB_y(Q) - BB_y(P)$, or $d = BB_{time}(Q) - BB_{time}(P)$).

4. The velocity and rate of size change associated with motion operators $\uparrow_{v,r}$ and $\downarrow_{v,r}$: Operator $\uparrow_{v,r}$ denotes that the object moves along the positive direction of the *x*- (or *y*-) axis. Operator $\downarrow_{v,r}$ denotes that the object moves along the negative direction of the *x*- (or *y*-) axis. v is the velocity of the motion and r is the rate of size change of an object.

To represent the time intervals when the states of an object are changed, we introduce one more temporal operator $|_t$ in the 3D C-string. For example, $P_3 |_t P_5$ denotes that in the first 3 frames, object P remains in the same state of the motion and size change. However, from frame 4 to frame 8, the state of the motion and size change of object P is changed into another. Note that in the 3D C-string representation, the motion operators are used to describe the states of motions and size changes, and the t-string is used to describe how long every state lasts.

In the 3D C-string representation, Lee *et al.* [27] also introduced the concept of template objects. A template object is a pair of separators, "(" and ")", containing a set of objects. For example, $_0(A_3 <_2 B_3)$ is a template object whose initial location is 0 and whose size is 3 + 2 + 3 = 8.

To see how 3D C-string works, let's consider the following example as shown in Fig. 1. In this example, the video contains two still objects (houses) and two moving objects (cars). All the objects are approximated by the MBRs. Let's consider how to generate the *u*-string for the video. First of all, we project the initial locations of all objects onto the *x*-axis. Next, we scan the beginning and ending points of the objects from left to right to generate the *u*-string. We find that the *x*-projects of objects C and D are identical and both objects moves along the positive direction of the *x*-axis with the velocity of 2 pixels/frame, so we have $(C_2\uparrow_{2,1} = D_2\uparrow_{2,1})$. The template object $(C_2\uparrow_{2,1} = D_2\uparrow_{2,1})$ is joined with the *x*-projection of object A, so we have $((C_2\uparrow_{2,1} = D_2\uparrow_{2,1}) \mid A_4)$. The difference



(a) A video contains 6 frames.

Fig. 1. An example video.

u-string: $(((C_2{\uparrow}_{2,1} = D_2{\uparrow}_{2,1})\,|\,A_4) <_2 B_4)$
v-string: $(D_1{\uparrow}_{0,1} <_1 (C_1{\uparrow}_{0,1} <_1 (A_4 = B_4)))$
t-string: $(A_6 = B_6 = C_6 = D_6)$

(b) The corresponding 3D C-string.

Fig. 1. (Cont'd) An example video.

between the ending bound of template object $((C_2{\uparrow}_{2,1} = D_2{\uparrow}_{2,1})\,|\,A_4)$ and the beginning bound of the x-projection of object B is equal to 2, so we have $(((C_2{\uparrow}_{2,1} = D_2{\uparrow}_{2,1})\,|\,A_4) <_2 B_4)$. Note that since both objects A and B are still in the video, there is no velocity associated with them. The corresponding 3D C-string of the video is shown in Fig. 1 (b). Objects C and D move along the positive direction of the x-axis with the velocity of 2 units/ frame in frames 1-6. Objects A and B do not move or change their sizes, so there are no motion operators for both objects. The knowledge structure of 3D C-string provides an easy and compact way to represent the spatial and temporal relations between the objects in a video.

3. INFERENCE RULES

In this section, we present the inference rules to derive the spatial and temporal relations between each pair of objects in the 3D C-string. There are four kinds of fundamental rules: transitive rules, distributive rules, manipulation rules and integration rules. In these inference rules, we abbreviate the velocities and rates of size changes since both of them are not changed in the inference process. After deriving the relation between two objects, we can obtain their velocities and rates of size changes from the given 3D C-string, and which can be easily applied to reasoning about spatio-temporal relations between the objects in a video.

Let $\mathcal{R}$ = {"<", "|", "=", "[", "]", "%", "$|_t$"} be the set of relation operators. In the *time* dimension, operator "$|_t$" can be inferred as the same as operator "|". First of all, let's consider the 3-object strings of o_1, o_2, and o_3, where o_i, $1 \le i \le 3$, may be a cut subobject (subobject for short), an object without any cutting (non-cut object for short) or a template object. There are three types of 3-object strings in the 3D C-string representation as follows, where $L0$ is the initial location associated with the template object represented by the 3-object string.

Type-I: ${}_{L0}(o_1 r_{12}(o_2 r_{23} o_3))$, where r_{12} and $r_{23} \in \mathcal{R}$; however, r_{12} and r_{23} are not "=" at the same time.

Type-II: ${}_{L0}((o_1 r_{12} o_2) r_{23} o_3)$, where r_{12} and $r_{23} \in \mathcal{R}$; however, r_{12} and r_{23} are not "=" at the same time.

Type-III: ${}_{L0}(o_1 r_{12} o_2 r_{23} o_3)$, where both r_{12} and r_{23} are "=".

3.1 Transitive Rules for a 3-Object String

For each type of strings, we can easily infer three binary relations between o_1, o_2, and o_3. These inferred binary relations can be denoted by three substrings: λ_1: ${}_{L1}(o_1 r'_{12} o_2)$,

λ_2: ${}_{L2}(o_2 r'_{23} o_3)$, and λ_3: ${}_{L3}(o_1 r'_{13} o_3)$. Each substring represents a template object which is associated with the initial location, namely $L1$, $L2$ and $L3$. If the initial location of the template object is equal to 0, it can be omitted.

Among them, it is easiest for a type-III string to derive the three binary relations by intuition, namely, ${}_{L0}(o_1 = o_2)$, ${}_{L0}(o_2 = o_3)$ and ${}_{L0}(o_1 = o_3)$. That is, the derived relation and metric information are the same as the original string. For a string of the first two types, the derived relations and metric information are shown in the following sections.

3.1.1 The relation transitive rules for type-I and type-II strings

The inferred binary relations of a 3-object string of type-I are shown in Figs. 2 (a-c). The inferred binary relations of a 3-object string of type-II are shown in Figs. 3 (a-c), where "N" denotes that the case is not available in the 3D C-string representation; "]*", "*[" and "%*" are the reverse relations of "]", "[" and "%", respectively. That is, A]* B is equal to B] A.

r'_{12}		r_{23}					
		<	|	=	[	]	%
	<	<	<	<	<	<	<
	|	|	|	|	|	|	|
r_{12}	=	[	[	=	[	]	=
	[	[	[	[	[	[	[
	]	%	%	]	]	]	]
	%	%	%	%	%	%	%

(a) The inferred relation r'_{12}.

r'_{23}		r_{23}					
		<	|	=	[	]	%
	<	<	|	=	[	]	%
	|	<	|	=	[	]	%
r_{12}	=	<	|	=	[	]	%
	[	<	|	=	[	]	%
	]	<	|	=	[	]	%
	%	<	|	=	[	]	%

(b) The inferred relation r'_{23}.

r'_{13}		r_{23}					
		<	|	=	[	]	%
	<	<	<	<	<	<	<
	|	<	<	|	|	<	<
r_{12}	=	]	]	=	[	]	%
	[	%	%	[	[	%	%
	]	]	]	]	%	]	%
	%	%	%	%	%	%	%

(c) The inferred relation r'_{13}.

Fig. 2. The inference relation tables for a type-I string.

r'_{12}		r_{23}					
		<	|	=	[	]	%
	<	<	<	<	N	N	N
	|	|	|	|	N	N	N
r_{12}	=	=	=	=	=	=	=
	[	[	[	[	N	N	N
	]	]	]	]	N	N	N
	%	%	%	%	N	N	N

(a) The inferred relation r'_{12}.

r'_{23}		r_{23}					
		<	|	=	[	]	%
	<	<	|	[*	N	N	N
	|	<	|	[*	N	N	N
r_{12}	=	<	|	=	[	]	%
	[	<	<	*[	N	N	N
	]	<	|	]*	N	N	N
	%	<	<	%*	N	N	N

(b) The inferred relation r'_{23}.

r'_{13}		r_{23}					
		<	|	=	[	]	%
	<	<	<	[*	N	N	N
	|	<	<	[*	N	N	N
r_{12}	=	<	|	=	[	]	%
	[	<	|	=	N	N	N
	]	<	|	=	N	N	N
	%	<	|	=	N	N	N

(c) The inferred relation r'_{13}.

Fig. 3. The inference relation tables for a type-II string.

3.1.2 The metric information of the inferred relations for type-I and type-II strings

The metric information of the inferred relations for a type-I string is listed in Fig. 4, where "N" denotes that there is not metric information for those cases, $M(r)$ denotes the metric information of operator r and S_i denotes the size of object i. The metric information of the inferred relations for a type-II string is listed in Fig. 5.

$M(r'_{12})$		r_{23}					
		$<$	|	$=$	[	]	%
	$<$	$M(r_{12})$	$M(r_{12})$	$M(r_{12})$	$M(r_{12})$	$M(r_{12})$	$M(r_{12})$
	|	N	N	N	N	N	N
r_{12}	$=$	N	N	N	N	N	N
	[	N	N	N	N	N	N
	]	S_1-S_2-S_3-$M(r_{12})$	S_1-S_2-S_3	N	N	N	N
	%	$M(r_{12})$	$M(r_{12})$	$M(r_{12})$	$M(r_{12})$	$M(r_{12})$	$M(r_{12})$

(a) The inferred metric information of r'_{12}.

$M(r'_{23})$		r_{23}					
		$<$	|	$=$	[	]	%
	$<$	$M(r_{23})$	N	N	N	N	$M(r_{23})$
	|	$M(r_{23})$	N	N	N	N	$M(r_{23})$
r_{12}	$=$	$M(r_{23})$	N	N	N	N	$M(r_{23})$
	[	$M(r_{23})$	N	N	N	N	$M(r_{23})$
	]	$M(r_{23})$	N	N	N	N	$M(r_{23})$
	%	$M(r_{23})$	N	N	N	N	$M(r_{23})$

(b) The inferred metric information of r'_{23}.

$M(r'_{13})$		r_{23}					
		$<$	|	$=$	[	]	%
	$<$	$M(r_{12})+S_2+M(r_{23})$	$M(r_{12})+S_2$	$M(r_{12})$	$M(r_{12})$	$M(r_{12})+S_2$-S_3	$M(r_{12})+M(r_{23})$
	|	$S_2+M(r_{23})$	S_2	N	N	S_2-S_3	$M(r_{23})$
r_{12}	$=$	N	N	N	N	N	$M(r_{23})$
	[	$S_2+M(r_{23})$	S_2	N	N	S_2-S_3	$M(r_{23})$
	]	N	N	N	S_1-S_2	N	S_1-$S_2+M(r_{23})$
	%	$M(r_{12})+S_2+M(r_{23})$	$M(r_{12})+S_2$	$M(r_{12})$	$M(r_{12})$	$M(r_{12})+S_2$-S_3	$M(r_{12})+M(r_{23})$

(c) The inferred metric information of r'_{13}.

Fig. 4. The metric information of inferred relations for a type-I string.

$M(r'_{12})$		r_{23}					
		$<$	|	$=$	[	]	%
	$<$	$M(r_{12})$	$M(r_{12})$	$M(r_{12})$	N	N	N
	|	N	N	N	N	N	N
r_{12}	$=$	N	N	N	N	N	N
	[	N	N	N	N	N	N
	]	N	N	N	N	N	N
	%	$M(r_{12})$	$M(r_{12})$	$M(r_{12})$	N	N	N

(a) The inferred metric information of r'_{12}.

$M(r'_{23})$		r_{23}					
		$<$	|	$=$	[	]	%
	$<$	$M(r_{23})$	N	N	N	N	N
	|	$M(r_{23})$	N	N	N	N	N
r_{12}	$=$	$M(r_{23})$	N	N	N	N	$M(r_{23})$
	[	S_1-S_2-$M(r_{23})$	S_1-S_2	N	N	N	N
	]	$M(r_{23})$	N	N	N	N	N
	%	S_1-S_2-$M(r_{12})+M(r_{23})$	S_1-S_2-$M(r_{12})$	$M(r_{12})$	N	N	N

(b) The inferred metric information of r'_{23}.

Fig. 5. The metric information of inferred relations of a type-II string.

$M(r'_{13})$	r_{23}: <	|	=	[	]	%
r_{12}: <	$S_2+M(r_{12})+M(r_{23})$	$S_2+M(r_{12})$	N	N	N	N
|	$S_2+M(r_{23})$	S_2	N	N	N	N
=	$M(r_{23})$	N	N	N	N	$M(r_{23})$
[	$M(r_{23})$	N	N	N	N	N
]	$M(r_{23})$	N	N	N	N	N
%	$M(r_{23})$	N	N	N	N	N

(c) The inferred metric information of r'_{13}.

Fig. 5. (Cont'd) The metric information of inferred relations of a type-II string.

3.1.3 The initial locations

The initial locations associated with substrings λ_1, λ_2 and λ_3 are the same as the locations of o_1, o_2 and o_1, respectively. The location of o_1 is the same as the initial location associated with the original 3-object string. So, we only need to compute the initial location associated with substring λ_2. The initial location associated with substring λ_2, $L2$, can be derived from Figs. 6 (a) and (b) for type-I and type-II strings, respectively. For a type-III string, the derived relation and metric information are the same as the original string.

$L2$	r_{23}: <	|	=	[	]	%
r_{12}: <	$L0+S_1+M(r_{12})$	$L0+S_1+M(r_{12})$	$L0+S_1+M(r_{12})$	$L0+S_1+M(r_{12})$	$L0+S_1+M(r_{12})$	$L0+S_1+M(r_{12})$
|	$L0+S_1$	$L0+S_1$	$L0+S_1$	$L0+S_1$	$L0+S_1$	$L0+S_1$
=	$L0$	$L0$	$L0$	$L0$	$L0$	$L0$
[	$L0$	$L0$	$L0$	$L0$	$L0$	$L0$
]	$L0+S_1-S_2-S_3-M(r_{12})$	$L0+S_1-S_2-S_3$	$L0+S_1-S_2$	$L0+S_1-S_2$	$L0+S_1-S_2$	$L0+S_1-S_2$
%	$L0+M(r_{12})$	$L0+M(r_{12})$	$L0+M(r_{12})$	$L0+M(r_{12})$	$L0+M(r_{12})$	$L0+M(r_{12})$

(a) The initial location associated with substring λ_2 for a type-I string.

$L2$	r_{23}: <	|	=	[	]	%
r_{12}: <	$L0+S_1+M(r_{12})$	$L0+S_1+M(r_{12})$	$L0$	N	N	N
|	$L0+S_1$	$L0+S_1$	$L0$	N	N	N
=	$L0$	$L0$	$L0$	$L0$	$L0$	$L0$
[	$L0$	$L0$	$L0$	N	N	N
]	$L0+S_1-S_2$	$L0+S_1-S_2$	$L0$	N	N	N
%	$L0+M(r_{12})$	$L0+M(r_{12})$	$L0$	N	N	N

(b) The initial location associated with substring λ_2 for a type-II string.

Fig. 6. The initial location associated with substring λ_2 for type-I and type-II string.

Now, let's consider a type-I string $_{10}(A_7 <_2 (B_3 \,|\, C_2))$, where the subscript 10 is the initial location associated with the template object and the relations of r_{12} and r_{23} are "<" and "|", respectively. We can derive the following substrings: ω_1: $(A_7 < B_3)$ from Fig. 2

(a), ω_2: $(B_3 | C_2)$ from Fig. 2 (b), and ω_3: $(A_7 < C_2)$ from Fig. 2 (c). The distance associated with operator < in substring ω_1 is equal to $M(<_2) = 2$ from Fig. 4 (a), operator | in substring ω_2 does not have metric information from Fig. 4 (b), and the distance associated with operator < in substring ω_3 is equal to $M(<_2) + S_B = 2 + 3 = 5$ from Fig. 4 (c). The initial location associated with substring ω_2 is equal to $10 + S_A + M(<_2) = 10 + 7 + 2 = 19$ from Fig. 6 (a).

The relation and metric information between any two objects in a 3-object string can be derived from the tables as shown in Figs. 2 to 6. For a 3D C-string containing more than 3 objects, we need more inference rules and discuss them in following subsections.

3.2 Transitive Rules for the String with More Than 3 Objects

(TR-1) If $r_{12} \in \mathcal{R}$, and $r_{(i-1)i} \in$ {"=", "[", "]", "%"}, $2 < i \le n$, and the first level of the 3D C-string is of type-I, the string will be in the form of ${}_{L0}(o_1r_{12}(o_2r_{23}(o_3r_{34}(\ldots(o_{(n-1)}r_{(n-1)n}\, o_n)\ldots))))$. Let o'_3 be $(o_3r_{34}(\ldots(o_{(n-1)}r_{(n-1)n}o_n)\ldots))$. The string can be rewritten as ${}_{L0}(o_1r_{12}(o_2\, r_{23}o'_3))$. So, we can get the following three substrings.

λ_4: ${}_{L0}(o_1r'_{12}o_2)$, where r'_{12} can be derived from Fig. 2 (a),
λ_5: ${}_{L2}(o_2r_{23}o'_3)$, that is, ${}_{L2}(o_2r_{23}(o_3r_{34}(\ldots(o_{(n-1)}r_{(n-1)n}o_n)\ldots)))$, where $L2$ can be derived from Fig. 6 (a),
λ_6: ${}_{L0}(o_1r'_{13}o'_3)$, that is, ${}_{L0}(o_1r'_{13}(o_3r_{34}(\ldots(o_{(n-1)}r_{(n-1)n}o_n)\ldots)))$, where r'_{13} can be derived from Fig. 2 (c).

For example, for a 3D C-string ${}_{L0}(A \,\%\, (B\,]\,(C\,[\,D)))$, we will have the following three substrings: ω_4: ${}_{L0}(A \,\%\, B)$, ω_5: ${}_{L2}(B\,]\,(C\,[\,D))$, and ω_6: ${}_{L0}(A \,\%\, (C\,[\,D))$, where the metric information is omitted. Because the substrings ω_5 and ω_6 are in the form of rule (TR-1), rule (TR-1) can be recursively applied to find every binary relation between the objects.

(TR-2) If $r_{(n-1)n} \in$ {"<", "|", "$|_t$"}, and $r_{(i-1)i} \in$ {"=", "[", "]", "%"}, $1 < i < n$, and the first level of the 3D C-string is of type-II, the string will be in the form of ${}_{L0}((o_1r_{12}(o_2r_{23}(\ldots(o_{(n-2)}r_{(n-2)(n-1)}o_{(n-1)})\ldots)))r_{(n-1)n}o_n)$. Let o'_2 be $(o_2r_{23}(\ldots(o_{(n-2)}r_{(n-2)(n-1)}o_{(n-1)})\ldots))$. The string can be rewritten as ${}_{L0}((o_1r_{12}o'_2)r_{(n-1)n}o_n)$. So, we will have the following three substrings.

λ_7: ${}_{L0}(o_1r_{12}o'_2)$, that is, ${}_{L0}(o_1r_{12}(o_2r_{23}(\ldots(o_{(n-2)}r_{(n-2)(n-1)}o_{(n-1)})\ldots)))$,
λ_8: ${}_{L2}(o'_2r'_{(n-1)n}o_n)$, that is, ${}_{L2}((o_2r_{23}(\ldots(o_{(n-2)}r_{(n-2)(n-1)}o_{(n-1)})\ldots))r'_{(n-1)n}o_n)$, where $r'_{(n-1)n}$ and $L2$ can be derived from Fig. 3 (b) and Fig. 6 (b), respectively,
λ_9: ${}_{L0}(o_1r'_{1n}o_n)$, where r'_{1n} can be derived from Fig. 3 (c).

For example, for a 3D C-string, ${}_{L0}((A\,[\,(B\,]\,(C \,\%\, D)))\,|\,E)$, we will have the following three substrings: ω_7: ${}_{L0}(A\,[\,(B\,]\,(C \,\%\, D)))$, ω_8: ${}_{L2}((B\,]\,(C \,\%\, D)) < E)$, and ω_9: ${}_{L0}(A\,|\,E)$, where the metric information is omitted. Substring ω_7 is in the form of (TR-1), and substring ω_8 is in the form of (TR-2), so (TR-1) and (TR-2) can be recursively applied to find every binary relation between the objects.

3.3 Distributive Rules

We have discussed the 3D C-strings with the form of nested parentheses in the previous section. To infer the relations between the objects in the 3D C-strings of another form, we first consider a four-symbol string in the form of ${}_{L0}((o_1r_{12}\,o_2)r_{23}(o_3r_{34}o_4))$. Let o'_3 be $(o_3r_{34}o_4)$. The string can be rewritten as ${}_{L0}((o_1r_{12}o_2)r_{23}o'_3)$ which is of type-II. So, we can have following three substrings: λ_{10}: ${}_{L0}(o_1r'_{13}o'_3)$ which is ${}_{L0}(o_1r'_{13}(o_3r_{34}o_4))$, λ_{11}: ${}_{L2}(o_2\,r'_{23}o'_3)$ which is ${}_{L2}((o_2r'_{23}(o_3r_{34}o_4))$, and λ_{12}: ${}_{L0}(o_1r_{12}o_2)$.

Or, we can replace $(o_1r_{12}o_2)$ with o'_2. The string can be rewritten as ${}_{L0}(o'_2r_{23}\,(o_3r_{34}o_4))$ which is of type-I. So, we can have the following three substrings: λ_{13}: ${}_{L0}(o'_2r'_{23}o_3)$, that is, ${}_{L0}((o_1r_{12}o_2)r'_{23}o_3)$, λ_{14}: ${}_{L2}((o_3r_{34}o_4)$, and λ_{15}: ${}_{L0}(o'_2r'_{24}o_4)$, that is, ${}_{L0}((o_1r_{12}o_2)r'_{24}o_4)$.

After the distribution, we can easily apply the transitive rules to infer the relations between each pair of objects.

For example, for a 3D C-string, ${}_{L0}((A\,\%\,B)<(C\,[\,D))$, we can replace $(A\,\%\,B)$ with o'_2. The string can be written as ${}_{L0}(o'_2<(C\,[\,D))$ which is of type-I. So, we can have the following three substrings: ω_{10}: ${}_{L0}(o'_2<C)$ which is ${}_{L0}((A\,\%\,B)<C)$, ω_{11}: ${}_{L0}(o'_2<D)$ which is ${}_{L0}((A\,\%\,B)<D)$, ω_{12}: ${}_{L2}(C\,[\,D)$. For the substrings ω_{10} and ω_{11}, we can apply the transitive rules to infer the relations between the objects.

Table 2. The manipulation rules.

Rule	String format	Resultant string	Size of P	Size of Q	Metric of the relation
MR-1	$(P^1\|(P^2\ldots\|(P^{n-1}\|P^n)\ldots))$	P	$S_P=S_{P1}+S_{P2}+\ldots+S_{Pn}$		N
MR-2	$((P^1=Q^1)\|((P^2=Q^2)\|(\ldots\|(P^n=Q^n)\ldots)))$	$P=Q$	$S_P=S_{P1}+S_{P2}+\ldots+S_{Pn}$	$S_Q=S_{Q1}+S_{Q2}+\ldots+S_{Qn}$	N
MR-3	$((P^1=Q^1)\|(P^2[Q^2))$	$(P[Q)$	$S_P=S_{P1}+S_{P2}$	$S_Q=S_{Q1}+S_{Q2}$	N
MR-4	$((P^1=Q^1)\|P^2)$	$(P[Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	N
MR-5	$((P^1[Q)\|P^2)$	$(P[Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	N
MR-6	$(P^1\|(P^2]Q))$	$(P]Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	N
MR-7	$(P^1\|(P^2=Q))$	$(P]Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	N
MR-8	$((P^1]Q^1)\|((P^2=Q^2))$	$(P]Q)$	$S_P=S_{P1}+S_{P2}$	$S_Q=S_{Q1}+S_{Q2}$	N
MR-9	$(P^1\|(P^2\%Q))$	$(P\%Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	$M(\%)=M(\%)+S_P$
MR-10	$(P^1\|(P^2[Q))$	$(P\%Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	$M(\%)=S_{P1}$
MR-11	$((P^1]Q^1)\|(P^2[Q^2))$	$(P\%Q)$	$S_P=S_{P1}+S_{P2}$	$S_Q=S_{Q1}+S_{Q2}$	$M(\%)=S_{P1}-S_{Q1}$
MR-12	$((P^1]Q)\|P^2)$	$(P\%Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	$M(\%)=S_{P1}-S_Q$
MR-13	$((P^1\%Q)\|P^2)$	$(P\%Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	$M(\%)=M(\%)$
MR-14	$(P^1\|(P^2/Q))$	(P/Q)	$S_P=S_{P1}+S_{P2}$	S_Q	$M(/)=M(/)$
MR-15	$(P^1\|(Q[P^2))$	(P/Q)	$S_P=S_{P1}+S_{P2}$	S_Q	$M(/)=S_{P2}$
MR-16	$((P^1]Q^1)\|(Q^2[P^2))$	(P/Q)	$S_P=S_{P1}+S_{P2}$	$S_Q=S_{Q1}+S_{Q2}$	$M(/)=S_{P2}+S_{Q1}$
MR-17	$((P]Q^1)\|Q^2)$	(P/Q)	S_P	$S_Q=S_{Q1}+S_{Q2}$	$M(/)=S_{Q1}$
MR-18	$((P/Q^1)\|Q^2)$	(P/Q)	S_P	$S_Q=S_{Q1}+S_{Q2}$	$M(/)=M(/)$

3.4 Manipulation Rules

In the 3D C-string representation, the subobjects would only be present in the relation of *part_overlap*, "/". It is necessary for us to manipulate (merge) the subobjects. The manipulation rules are used to merge together the subobjects in one string (or substring). The integration rules in section 3.4 are used to merge together the subobjects in two strings (or substrings). We present 18 manipulation rules in this subsection as shown in Table 2 and 21 integration rules in the next subsection. In Table 2, the first column shows the name of a rule, and the second column presents the original string format, where P^i, $1 \le i \le n$, are the subobjects of P. The merged result is listed in the third column, and the size of P or Q and metric of the relation are shown in the remaining columns, where 'N' denotes that there is not metric information for those cases, and S_P denotes the size of object P.

We do not use operator / in the 3D C-string representation; however, we do use operator / in the inference process. The distance associated with operator /, $P /_d Q$, is equal to the distance between the x- (y- or *time*-) projection of object P and that of object Q, where $d = EB_x(P) - BB_x(Q)$ ($d = EB_y(P) - BB_y(Q)$, or $d = EB_{time}(P) - BB_{time}(Q)$).

These manipulation rules are used to merge the subobjects in a string together. The location of merged object keeps unchanged. Let's consider an example as follows, where objects A and B are partitioned into several subobjects in a string, $(A_2 \mid (A_2 \mid (((A_4] B_2) \mid ((A_2 = B_2) \mid (A_2 = B_2))) \mid (B_2 \mid B_2))))$. We can merge objects A and B together in the following steps.

$$\begin{array}{ll}
 & (A_2 \mid (A_2 \mid (((A_4] B_2) \mid ((A_2 = B_2) \mid (A_2 = B_2))) \mid \underline{(B_2 \mid B_2)}))) \\
\text{(MR-1)} & (A_2 \mid (A_2 \mid (((A_4] B_2) \mid \underline{((A_2 = B_2) \mid (A_2 = B_2))}) \mid B_4))) \\
\text{(MR-2)} & (A_2 \mid (A_2 \mid \underline{(((A_4] B_2) \mid (A_4 = B_4))} \mid B_4))) \\
\text{(MR-8)} & (A_2 \mid (A_2 \mid \underline{((A_8] B_6) \mid B_4)})) \\
\text{(MR-17)} & (A_2 \mid \underline{(A_2 \mid (A_8 /_6 B_{10}))}) \\
\text{(MR-14)} & \underline{(A_2 \mid (A_{10} /_6 B_{10}))} \\
\text{(MR-14)} & (A_{12} /_6 B_{10})
\end{array}$$

where the underline parts will be merged together in the next step. Finally, the relation between A and B is $(A_{12} /_6 B_{10})$.

3.5 Integration Rules

The integration rules are used to merge the subobjects in several substrings together. There are 21 integration rules in total as shown in Table 3. In Table 3, the first column shows the name of a rule, the second and third columns present the two substrings to be combined. The integrated result is listed in the next column, and the size of P or Q and metric of relation are shown in the remaining columns, where "N" denotes that there is not metric information for those cases, M_1 and M_2 are the metric information of the operators in String 1 and String 2, respectively.

Table 3. The integration rules.

Rule	String 1	String 2	Resultant string	Size of P	Size of Q	Metric of the relation
IR-1	$(P^1<Q)$	$(P^2<Q)$	$(P<Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	$M(<)=\mathrm{Min}(M_1(<), M_2(<))$
IR-2	$(P<Q^1)$	$(P<Q^2)$	$(P<Q)$	S_P	$S_Q=S_{Q1}+S_{Q2}$	$M(<)=\mathrm{Min}(M_1(<), M_2(<))$
IR-3	$(P^1<Q)$	$(P^2\|Q)$	$(P\|Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	N
IR-4	$(P\|Q^1)$	$(P<Q^2)$	$(P\|Q)$	S_P	$S_Q=S_{Q1}+S_{Q2}$	N
IR-5	$(P^1=Q^1)$	$(P^2=Q^2)$	$(P=Q)$	$S_P=S_{P1}+S_{P2}$	$S_Q=S_{Q1}+S_{Q2}$	N
IR-6	$(P^1=Q^1)$	$(P^2[Q^2)$	$(P[Q)$	$S_P=S_{P1}+S_{P2}$	$S_Q=S_{Q1}+S_{Q2}$	N
IR-7	$(P^1=Q)$	$(Q\|P^2)$	$(P[Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	N
IR-8	$(P^1[Q)$	$(Q<P^2)$	$(P[Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	N
IR-9	$(P^1<Q)$	$(P^2]Q)$	$(P]Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	N
IR-10	$(P^1\|Q)$	$(P^2=Q)$	$(P]Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	N
IR-11	$(P^1]Q^1)$	$(P^2=Q^2)$	$(P]Q)$	$S_P=S_{P1}+S_{P2}$	$S_Q=S_{Q1}+S_{Q2}$	N
IR-12	$(P^1<Q)$	$(P^2\%Q)$	$(P\%Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	$M(\%)=M(\%)+S_{P1}$
IR-13	$(P^1\|Q)$	$(P^2[Q)$	$(P\%Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	$M(\%)=S_{P1}$
IR-14	$(P^1]Q^1)$	$(P^2[Q^2)$	$(P\%Q)$	$S_P=S_{P1}+S_{P2}$	$S_Q=S_{Q1}+S_{Q2}$	$M(\%)=S_{P1}-S_{Q1}$
IR-15	$(P^1]Q)$	$(\mathrm{Q}\|P^2)$	$(P\%Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	$M(\%)=S_{P1}-S_Q$
IR-16	$(P^1\%Q)$	$(\mathrm{Q}<P^2)$	$(P\%Q)$	$S_P=S_{P1}+S_{P2}$	S_Q	$M(\%)=M_1(\%)$
IR-17	$(P^1<Q)$	(P^2/Q)	(P/Q)	$S_P=S_{P1}+S_{P2}$	S_Q	$M(/)=M_2(/)$
IR-18	$(P^1\|Q)$	$(\mathrm{Q}[P^2)$	(P/Q)	$S_P=S_{P1}+S_{P2}$	S_Q	$M(/)=S_{P2}$
IR-19	$(P^1]Q^1)$	$(\mathrm{Q}^2[P^2)$	(P/Q)	$S_P=S_{P1}+S_{P2}$	$S_Q=S_{Q1}+S_{Q2}$	$M(/)=S_{P2}+S_{Q1}$
IR-20	$(P]Q^1)$	$(P\|Q^2)$	(P/Q)	S_P	$S_Q=S_{Q1}+S_{Q2}$	$M(/)=S_{Q1}$
IR-21	(P/Q^1)	$(P<Q^2)$	(P/Q)	S_P	$S_Q=S_{Q1}+S_{Q2}$	$M(/)=M_1(/)$

These integration rules can be used to merge the subobjects in several substrings together. For example, object A is partitioned into two subobjects: A_2^1 and A_3^2, and B is partitioned into three subobjects: B_3^1, B_3^2, and B_4^3. Those subobjects appear in six substrings: ω_{13}: $(A_2^1 \mid B_3^1)$, ω_{14}: $(A_3^2 = B_3^1)$, ω_{15}: $(A_2^1 \mid B_3^2)$, ω_{16}: $(A_3^2 <_3 B_3^2)$, ω_{17}: $(A_2^1 <_7 B_4^3)$, and ω_{18}: $(A_3^2 <_4 B_4^3)$. We can use the integration rules to derive the relation between objects A and B in the following steps.

(IR-10) Integrate ω_{13} and ω_{14} to ω_{19}: $(A_5^{1,2}] B_3^1)$
(IR-3) Integrate ω_{15} and ω_{16} to ω_{20}: $(A_5^{1,2} \mid B_3^2)$
(IR-1) Integrate ω_{17} and ω_{18} to ω_{21}: $(A_5^{1,2} <_4 B_4^3)$
(IR-20) Integrate ω_{19} and ω_{20} to ω_{22}: $(A_5^{1,2} /_3 B_6^{1,2})$
(IR-21) Integrate ω_{21} and ω_{22} to ω_{23}: $(A_5^{1,2} /_3 B_{10}^{1,2,3})$ => $(A_5 /_3 B_{10})$

where a superscript denotes a sequence number of the subobject and a subscript denotes the size of a subobject. Finally, the relation between A and B is $(A_5 /_3 B_{10})$.

3.6 Relation Derivation Algorithm

After presenting the inference rules, we propose the *relation derivation algorithm* to infer the spatio-temporal relations between the objects in a 3D C-string. Assume that there are k levels of template objects and m objects in a given 3D C u-string (v- or *t*-string), and those objects are numbered from 1 to *m*. For each subobject of object *i*, we assign a sequence number to it. For example, if object *i* is cut into three subobjects, those subobjects are numbered as *i*.1, *i*.2 and *i*.3, where 1, 2, and 3 are the sequence numbers. The *relation derivation algorithm* can infer a list of relations between any two objects in a given 3D C *u*-string (*v*- or *t*-string). The *relation derivation algorithm* is described in detail in Fig. 7.

```
Algorithm   relation derivation
Input: a 3D C-string $u$-string ($v$- or $t$-string).
Output: $\varphi$, the relations between any two objects.
1.  Let the relation list $\varphi$ be null and the string list $\psi$ be null.
2.  Assign a sequence number to each subobject of object $i$.
3.  Add the input string to $\psi$.
4.  while ($\psi$ is not empty) do
5.      Remove the first string $\eta$ from $\psi$.
6.      if ($\eta$ contains subobjects) then
7.          Apply the manipulation rules to merge the subobjects in $\eta$.
8.      end if
9.      Apply the transitive or distributive rules to $\eta$.
10.     for each generated substring $\rho$ do
11.         if ($\rho$ contains only two objects and is not in $\varphi$) then
12.             Append $\rho$ to $\varphi$.
13.         elseif ($\rho$ is not in $\psi$)
14.             Append $\rho$ to $\psi$.
15.         end if
16.     end for
17. end while
18. Apply the integration rules to the strings in $\varphi$.
```

Fig. 7. Relation derivation algorithm.

In this algorithm, we process a template object for each loop in steps 4-17. In steps 6-8, if η contains subobjects, we use the manipulation rules to merge the subobjects in η. Then, we apply the transitive or distributive rules to infer the relations between objects in step 9. In steps 10-16, if the generated substring ρ contains only two objects, collect it into φ; otherwise, collect it into ψ. The substrings in ψ need to be processed in the later steps. In the last step, we apply the integration rules to merge the subobjects in φ.

Lemma 1 For a 3D C *u*- (*v*-, or *t*-) string, the time complexity of the *relation derivation algorithm* is bounded to $O(k^2 + c \times m \times t + c^2 \times t^2)$, where *c* is the maximum number of cuttings among all individual objects, *m* is the number of objects, *t* is the number of cut objects, *k* is the number of levels of the template object in the input string.

Theorem 1 For a 3D C-string, the time complexity of the *relation derivation algorithm* is bounded to $O(k^2 + c \times m \times t + c^2 \times t^2)$, where *c* is the maximum number of cuttings among all individual objects; m is the number of objects in the input string; k is the maximum value of k_1, k_2, k_3; k_1, k_2, k_3 are the numbers of levels of the template objects in the u-, v-, and t-strings, respectively; and t is the maximum value of t_1, t_2, t_3; t_1, t_2, t_3 are the numbers of cut objects in u-, v-, t-strings, respectively.

Let's consider the example as shown in Fig. 1. The u-string of the video is ${}_0(((C_2\uparrow_{2,1} = D_2\uparrow_{2,1}) \mid A_4) <_2 B_4)$, which is of type-II. We can use the transitive rules to generate the following three substrings: ω_{24}: ${}_0((C_2\uparrow_{2,1} = D_2\uparrow_{2,1}) \mid A_4)$, ω_{25}: ${}_2(A_4 <_2 B_4)$, and ω_{26}: ${}_0((C_2\uparrow_{2,1} = D_2\uparrow_{2,1}) <_6 B_4)$.

The relations associated with substrings ω_{25} and ω_{26} can be derived from Figs. 3 (b) and (c). The metric information and initial location of substring ω_{25} can be derived from Fig. 5 (b) and Fig. 6 (b), and the metric information of substring ω_{26} can be derived from Fig. 5 (c). That is, $M(r_{13}) = S_2 + M(r_{23}) = 4$ (the size of object A) + 2 (metric of operator "<") = 6. Likewise, we can apply the same rule to substrings ω_{24} and ω_{26}. From substring ω_{24}, we can obtain the substrings ω_{27}: ${}_0(C_2\uparrow_{2,1} = D_2\uparrow_{2,1})$, ω_{28}: ${}_0(D_2\uparrow_{2,1} \mid A_4)$ and ω_{29}: ${}_0(C_2\uparrow_{2,1} \mid A_4)$. From substring ω_{26}, we can obtain the substrings ω_{30}: ${}_0(C_2\uparrow_{2,1} = D_2\uparrow_{2,1})$, ω_{31}: ${}_0(D_2\uparrow_{2,1} <_6 B_4)$ and ω_{32}: ${}_0(C_2\uparrow_{2,1} <_6 B_4)$. Hence, we can get all the relations between each pair of objects in the x dimension.

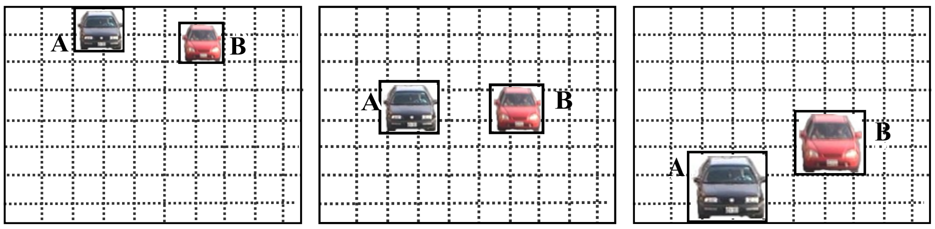
Similarly, from v-string $(D_1\uparrow_{0,1} <_1 (C_1\uparrow_{0,1} <_1 (A_4 = B_4)))$ we can derive the relation between each pair of objects in the y dimension. First of all, we apply the transitive rules to the v-string and obtain the following three substrings: ω_{33}: ${}_0(D_1\uparrow_{0,1} <_1 C_1\uparrow_{0,1})$, ω_{34}: ${}_2(C_1\uparrow_{0,1} <_1 (A_4 = B_4))$, and ω_{35}: ${}_0(D_1\uparrow_{0,1} <_3 (A_4 = B_4))$. Second, we can obtain the following substring: ω_{36}: ${}_2(C_1\uparrow_{0,1} <_1 A_4)$, ω_{37}: ${}_4(A_4 = B_4)$ and ω_{38}: ${}_2(C_1\uparrow_{0,1} <_1 B_4)$ from ω_{34}, and ω_{39}: ${}_0(D_1\uparrow_{0,1} <_3 A_4)$, ω_{40}: ${}_4(A_4 = B_4)$ and ω_{41}: ${}_0(D_1\uparrow_{0,1} <_3 B_4)$ from ω_{35}. Similarly, from t-string $(A_6 = B_6 = C_6 = D_6)$, that is of type-III, we can easily derive the relations between the objects, namely, the relation between each pair of objects in the *time* dimension is "=". That is, ω_{42}: ${}_0(A_6 = B_6)$, ω_{43}: ${}_0(A_6 = C_6)$, ω_{44}: ${}_0(A_6 = D_6)$, ω_{45}: ${}_0(B_6 = C_6)$, ω_{46}: ${}_0(B_6 = D_6)$, and ω_{47}: ${}_0(C_6 = D_6)$.

From substrings ω_{25}, ω_{37} and ω_{42}, we know that the relations between objects A and B are ${}_2(A_4 <_2 B_4)$ in the x dimension, ${}_4(A_4 = B_4)$ in the y dimension and ${}_0(A_6 = B_6)$ in the *time* dimension. So, we can find that objects A and B are both still, disjoined, and of the same size. From the substrings ω_{27}, ω_{33} and ω_{47}, we know that the relations between objects C and D are ${}_0(C_2\uparrow_{2,1} = D_2\uparrow_{2,1})$ in the x dimension, ${}_0(D_1\uparrow_{0,1} <_1 C_1\uparrow_{0,1})$ in the y dimension and ${}_0(C_6 = D_6)$ in the *time* dimension. So, we can find that objects C and D are moving at the same speed and object D is one unit below object C in the y dimension.

Let's consider another example. We add six more frames to the video shown in Fig. 1, where a moving car, E, has the same initial location and velocity as object C in frames 7~12. Hence, the corresponding 3D C-string for the video is shown as follows: u-string: $(((C_2\uparrow_{2,1} = D_2\uparrow_{2,1} = E_2\uparrow_{2,1}) \mid A_4) <_2 B_4)$, v-string: $(D_1\uparrow_{0,1} <_1 ((C_1\uparrow_{0,1} = E_1\uparrow_{0,1}) <_1 (A_4 = B_4)))$, t-string: $((A_{12} = B_{12}) = ((C_6 = D_6) \mid E_6))$. Then, we can infer the relation between objects C and E and obtain the following relations: ${}_0(C_2\uparrow_{2,1} = E_2\uparrow_{2,1})$ in the x dimension, ${}_0(C_1\uparrow_{0,1} = E_1\uparrow_{0,1})$ in the y dimension, and ${}_0(C_6 \mid E_6)$ in the *time* dimension. Both objects C and E have the "=" relation and the same velocity in the x and y dimensions. In the *time* dimension, both objects have the same size and "|" relation. That is, object E appears right after object C disappears. Therefore, with these rules, a user can easily derive the relations between the objects in a video represented by the 3D C-string.

4. AN APPLICATION

In this section, we demonstrate a video query example to show the effectiveness of our video algebra. The query video *V* of an overtaking event is shown in Fig. 8, where car *A* is overtaking car *B*. The query video is shown in Fig. 8 (a) and the corresponding 3D C-string is shown in Fig. 8 (b).



Frame 1 Frame 2 Frame 3

(a) A query video.

u-string: $(A_{16}\uparrow_{0,1.4} <_{20} B_{14}\uparrow_{0,1.2})$
v-string: $((B_{12}\downarrow_{15,1.4}]\, A_{10}\downarrow_{30,1.4}) \,|\, A_6)$
t-string: $(A_3 = B_3)$

(b) The corresponding 3D C-string.

Fig. 8. A video query example of an overtaking event.

u-string: $(A_{42}\uparrow_{0,1.4} \,|\, ((D_{45}\,]\, B_{12}\uparrow_{0,1.2}) \,|\, (C_{38}\uparrow_{0,1.2}\, [\, B_{28})))$
v-string: $((B_{62}\downarrow_{32,1.6}\,]\, A_{20}\downarrow_{45,1.8}) \,|\, ((A_{40} = C_{40}\downarrow_{30,1.5})\,]\, D_{30}))$
t-string: $((A_{60} = B_{60})\,]\, (C_{50}\,]\, D_1))$

Fig. 9. The 3D C-string of the matched video.

To find the videos similar to *V* from a database, we first choose the database videos that contain more than one car object. For example, let's consider a database video containing four car objects *A*, *B*, *C*, and *D*, and its corresponding 3D C-string is shown in Fig. 9.

Next, from the *u*-string, we can use the video algebra to infer the relation between each pair of objects in the *x* dimension, and obtain the following six substrings, ω_{48}: $_0(A_{42}\uparrow_{0,1.4} <_{33} B_{40}\uparrow_{0,1.2})$, ω_{49}: $_0(A_{42}\uparrow_{0,1.4} <_{45} C_{38}\uparrow_{0,1.2})$, ω_{50}: $_0(A_{42}\uparrow_{0,1.4} \,|\, D_{45})$, ω_{51}: $_{33}(B_{40}\uparrow_{0,1.2} /_{28} C_{38}\uparrow_{0,1.2})$, ω_{52}: $_{42}(D_{45} /_{12} B_{40}\uparrow_{0,1.2})$, and ω_{53}: $_{42}(D_{45} \,|\, C_{38}\uparrow_{0,1.2})$.

Similarly, from the *v*-string, we can derive the relation for each pair of objects in the *y* dimension, and obtain the following six substrings, ω_{54}: $_0(B_{62}\downarrow_{32,1.6} /_{20} A_{60}\downarrow_{45,1.8})$, ω_{55}: $_{62}(A_{60}\downarrow_{45,1.8}\,]\, C_{40}\downarrow_{30,1.5})$, ω_{56}: $_{62}(A_{60}\downarrow_{45,1.8}\,]\, D_{30})$, ω_{57}: $_0(B_{62}\downarrow_{32,1.6} \,|\, C_{40}\downarrow_{30,1.5})$, ω_{58}: $_0(B_{62}\downarrow_{32,1.6} <_{10} D_{30})$, and ω_{59}: $_{62}(C_{40}\downarrow_{30,1.5})\,]\, D_{30})$. From *t*-string, we can derive the relation for each pair of objects in the *time* dimension, and obtain the following six substrings, ω_{60}: $_0(A_{60} = B_{60})$, ω_{61}: $_0(A_{60}\,]\, C_{50})$, ω_{62}: $_0(A_{60}\,]\, D_1)$, ω_{63}: $_0(B_{60}\,]\, C_{50})$, ω_{64}: $_0(B_{60}\,]\, D_1)$, and ω_{65}: $_{10}(C_{50}\,]\, D_1)$.

According to the overtaking event between objects A and B in the query video, we can derive the "<" (disjoin) relation in the x dimension, the "/" (partly-overlap) relation in the y dimension, the "=" (equal) relation in the time dimension, and object A with a higher velocity than object B in the y dimension. In the video shown in Fig. 9, we can find that the relations between objects A and B are ω_{48}: ${}_0(A_{42}\uparrow_{0,1.4} <_{33} B_{40}\uparrow_{0,1.2})$ in the x dimension, ω_{54}: ${}_0(B_{62}\downarrow_{32,1.6} /_{20} A_{60}\downarrow_{45,1.8})$ in the y dimension, and ω_{60}: ${}_0(A_{60} = B_{60})$ in the time dimension, and object A with a higher velocity than object B in the y dimension. Thus, we can retrieve the database video similar to the query video and the database video is showed in Fig. 10.

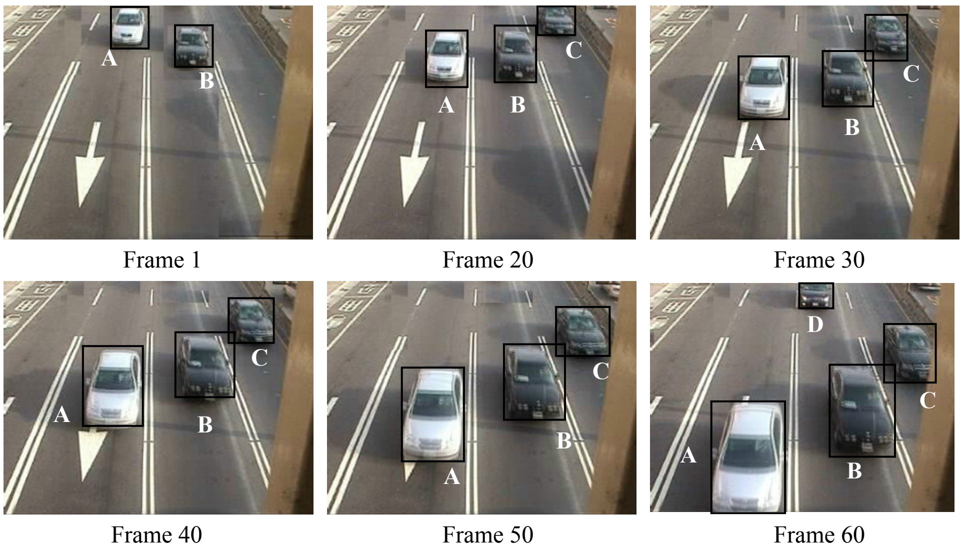


Fig. 10. The matched video.

5. CONCLUSIONS

The video content management has attracted increasing attention in recent years. We have proposed a new spatio-temporal knowledge structure, called 3D C-string, to represent the spatio-temporal relations between the objects in a video and to keep track of the motions and size changes of the objects. In this paper, we propose a video algebra to infer the spatio-temporal relations between the objects in a video represented by the 3D C-string. The algebra contains four kinds of rules, namely, transitive, distributive, manipulation, and integration rules. By using those rules, all the binary relations between the objects in a video can be derived from a given 3D C-string. These rules provide us the theoretic basis for spatio-temporal reasoning and video query inference. How to expand the reasoning result to generate the high-level semantics and to support high-level semantic video queries is worth further study.

REFERENCES

1. S. F. Chang, W. Chen, H. Meng, H. Sundaram, and D. Zhong, "A fully automatic content-based video search engine supporting multi-object spatio-temporal queries," *IEEE Transactions on Circuit Systems and Video Technology*, Vol. 8, 1998, pp. 602-615.
2. M. R. Naphade, I. V. Kozintsev, and T. S. Huang, "Factor graph framework for semantic video indexing," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, 2002, pp. 40-52.
3. C. W. Ngo, T. C. Pong, and H. J. Zhang, "Motion analysis and segmentation through spatio-temporal slices processing," *IEEE Transactions on Image Processing*, Vol. 12, 2003, pp. 341-355.
4. H. Yang, L. Chaisorn, Y. Zhao, S. Neo, and T. Chua, "VideoQA: question answering on news video," in *Proceedings of ACM International Conference on Multimedia*, 2003, pp. 632-641.
5. C. C. Lo, S. J. Wang, and L. W. Huang, "Video retrieval using successive modular operations on temporal similarity," *Computer Standards and Interfaces*, Vol. 26, 2004, pp. 317-328.
6. C. G. M. Snoek and M. Worring, "Multimedia event based video indexing using time intervals," *IEEE Transactions on Multimedia*, Vol. 7, 2005, pp. 638- 647.
7. S. K. Chang, Q. Y. Shi, and C. W. Yan, "Iconic indexing by 2D strings," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 9, 1987, pp. 413-429.
8. S. K. Chang, E. Jungert, and Y. Li, "Representation and retrieval of symbolic pictures using generalized 2D strings," in *Proceedings of SPIE on Visual Communications and Image Processing*, 1989, pp. 1360-1372.
9. E. Jungert and S. K. Chang, "An algebra for symbolic image manipulation and transformation," *Visual Database Systems*, North-Holland: Elsevier Science Publishers B.V., 1989.
10. S. Y. Lee and F. J. Hsu, "2D C-string: a new spatial knowledge representation for image database system," *Pattern Recognition*, Vol. 23, 1990, pp. 1077-1087.
11. S. Y. Lee and F. J. Hsu, "Picture algebra for spatial reasoning of iconic images represented in 2D C-string," *Pattern Recognition Letters*, Vol. 12, 1991, pp. 425-435.
12. S. Y. Lee and F. J. Hsu, "Spatial reasoning and similarity retrieval of images using 2D C-string knowledge representation," *Pattern Recognition*, Vol. 25, 1992, pp. 305-318.
13. P. W. Huang and Y. R. Jean, "Using 2D C^+-string as spatial knowledge representation for image database systems," *Pattern Recognition*, Vol. 27, 1994, pp. 1249-1257.
14. P. W. Huang and Y. R. Jean, "Spatial reasoning and similarity retrieval for image database systems based on RS-strings," *Pattern Recognition*, Vol. 29, 1996, pp. 2103-2114.
15. F. J. Hsu, S. Y. Lee, and B. S. Lin, "2D C-trees spatial representation for iconic image," *Journal of Visual Languages and Computing*, Vol. 10, 1999, pp. 147-164.
16. Y. I. Chang, H. Y. Ann, and W. H. Yeh, "A unique-ID-based matrix strategy for efficient iconic indexing of symbolic pictures," *Pattern Recognition*, Vol. 33, 2000, pp. 1263-1276.
17. Y. I. Chang, B. Y. Yang, and W. H. Yeh, "A generalized prime-number-based ma-

trix strategy for efficient iconic indexing of symbolic pictures," *Pattern Recognition Letters*, Vol. 22, 2001, pp. 657-666.
18. G. Petraglia, M. Sebillo, M. Tucci, and G. Tortora, "Virtual images for similarity retrieval in image databases," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 13, 2001, pp. 951-967.
19. Y. I. Chang, B. Y. Yang, and W. H. Yeh, "A bit-pattern-based matrix strategy for efficient iconic indexing of symbolic pictures," *Pattern Recognition Letters*, Vol. 24, 2003, pp. 537-545.
20. A. J. T. Lee and H. P. Chiu, "2D Z-string: A new spatial knowledge representation for image databases," *Pattern Recognition Letters*, Vol. 24, 2003, pp. 3015-3026.
21. K. Shearer, S. Venkatesh, and D. Kieronska, "Spatial indexing for video databases," *Journal of Visual Communication and Image Representation*, Vol. 7, 1996, pp. 325-335.
22. K. Shearer, D. Kieronska, and S. Venkatesh, "Resequencing of video using spatial indexing," *Journal of Visual Languages and Computing*, Vol. 8, 1997, pp. 193-214.
23. F. J. Hsu, S. Y. Lee, and B. S. Lin, "Video data indexing by 2D C-trees," *Journal of Visual Languages and Computing*, Vol. 9, 1998, pp. 375-397.
24. Y. K. Chan and C. C. Chang, "Spatial similarity retrieval in video databases," *Journal of Visual Communication and Image Representation*, Vol. 12, 2001, pp. 107-122.
25. C. C. Liu and A. L. P. Chen, "3D-list: a data structure for efficient video query processing," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 14, 2002, pp. 106-122.
26. A. J. T. Lee, H. P. Chiu, and P. Yu, "3D C-string: a new spatio-temporal knowledge structure for video database systems," *Pattern Recognition*, Vol. 35, 2002, pp. 2521-2537.

Anthony J. T. Lee (李瑞庭) received the B.S. degree from National Taiwan University, Taiwan, in 1983. He got the M.S. and Ph.D. degree in Computer Science from University of Illinois at Urbana-Champaign, U.S.A., in 1990 and 1993, respectively. In August 1993, he joined the Department of Information Management at National Taiwan University and he is now a professor. His current research interests include multimedia databases, temporal and spatial databases, and data mining.

Ping Yu (余平) is currently a Ph.D. candidate in the Department of Information Management, National Taiwan University, Taipei, Taiwan R.O.C. He received the B.S. degree from Chung Cheng Institute of Technology, Taiwan, in 1988, and the M.S. degree from National Defense Management College, Taiwan, in 1994. His current research interests include multimedia databases, temporal and spatial databases.

Han-Pang Chiu (邱漢邦) is currently a Ph.D. candidate in the Learning and Intelligent Systems group at the Computer Science and Artificial Intelligence Laboratory at Massachusetts Institute of Technology, U.S.A. He received his BBA degree from National Taiwan University, Taiwan, in 1999, and his MBA degree from the Department of Information Management, National Taiwan University, in June 2001. His current research interests include computer vision and machine learning. He is focusing on the topics of learning in visual perception, 3D object class recognition, and wide baseline stereo matching. His doctoral research is part of the Transfer Learning project supported by the Defense Advanced Research Projects Agency (DARPA).

Hsiu-Hui Lin (林秀慧) is currently a Ph.D. candidate in the Department of Information Management, National Taiwan University, Taipei, Taiwan, R.O.C. She received the B.S. degree from Department of Computer Science and Information Engineering, National Taiwan University, in 1988, and the M.S. degree from Department of Computer Science and Information Engineering, National Taiwan University, in 1991. Her current research interests include multimedia databases, and digital copyright management.