ヒューマンインタフェース 87－7
（2000．1．28）

字幕表示用言語 VCML の設計と その表示システムの開発

深田直秀, 渡邊 括行, 杉山 雅英

会津大学 コンピュータ理工学研究科

あらまし 本報告では, 字幕の表示やビデオ再生などを制御する記述言語 Video Caption Markup Language (VCML) を XML を利用した設計と, VCML 文書をもとに字幕を PC 上でリアルタイムで加工しビデオ画像に合成して表示するソフトウェア VCML Player の開発について述べる. VCML Player は Windows 98 上で動作する. VCML には字幕表示以外に画像表示や, 聴覚障害者用の環境音の表示や文字編集や効果の設定などの機能があり, VCML 文書を編集することにより新たな 1 つコンテンツの作成ができる. また, 実際のビデオデータとその台本を用いた字幕表示タイミングの自動決定についての実験を行い, 良好な結果を得ることができた.

Design of Video Caption Markup Language VCML and Development of VCML Player

N. Fukada, K. Watanabe, M. Sugiyama

Graduate School of Computer Science and Engineering
The University of Aizu

Abstract This paper designs a markup language named "Video Caption Markup Language(VCML)" which can control displaying caption and playing video, and develops a software named "VCML Player". VCML Player can generate caption images from the VCML document, compound a new image from video and caption images and display it. VCML functions are displaying images, environment sound images for hearing disable person, editing fonts and setting effects. VCML can create new contents by editing the VCML document. Furthermore, evaluation of a time alignment module for an actual video data is described and experimental results show a good performance.

1 まえがき

本報告では字幕を表示制御する Video Caption Markup Language (VCML) を設計と, Windows98 上において VCML を再生するためのプレーヤ, VCML Player を開発について述べる.

Windows における Real Player G2 に実装されている SMIL[2] では, 1 セットの独立したマルチメディアオブジェクトを 1 つの同期マルチメディア表現に統合を可能にしたが, 独立したオブジェクトを用いるため, 表示位置や表示方法の異なる字幕を表示する場合は別のストリームテキストを用意しなければならないという問題がある. VCML では 1 つの字幕を 1 つのオブジェクトとして表示時間を与えて表示させることで, より字幕表示に適した記述を可能にした. また, 映像の再生や画像の表示などの機能を含めることで字幕付き映像の表現を可能とした.

字幕付き映像の配送方法としてテレビ画面の走査線の 1 本を使って字幕の文字情報を送り, 専用のデコーダを用いてテレビに表示している. それ以外の配送方法は映像に直接字幕の内容を書き込み送信する方法であり, 専用の受信用機材を必要としないので最も使用されている. しかし, 映像に書き込む手法では, 字幕有無や言語の設定はできないなどの問題を抱えている. NHK ではニュース番組用に HMM を利用したディクテーションシステムが開発、試験されている [3]. TAO では視聴覚障害者向け放送ソフト製作

技術に関する研究開発が行われている [4].

2 VCMLの設計

今日のインターネット技術の発展により, 映像音声等の配布, 視聴は容易に行うことができる. そこで, インターネット上で文書やデータを交換したり, 配布したりするときの, 標準となる可能性のあるXML(拡張可能な記述言語) [1] を利用してVCMLを設計した. DTDに関しては付録を参照.

2.1 機能の要求条件

VCMLの機能を考える上でテレビ番組で使用されるテロップ (Television Opaque Projector) の技術を考察する. テロップは大きく二つに分類することができる. 一つはキャプションであり, もう一つは (サブ) タイトルである. キャプションは映画字幕などの様なもので, 話者が存在し, その発話内容に関しての文章を表示するものである. タイトルはキャプション以外のものであり, その内容は多種多様である. 例えば, エンディングに流れるスタッフロール, 大量な情報のスクロール, 図表や画像などである.

表示させる対象物はテキストや画像などであっても, 表示させるタイミングは必須であり, 表示の設定としてのフォントの種類, サイズや色, 表示位置, 表示手法 (スクロールなど) に関しては, ユーザ (制作者) の制作意図によるものとする.

画面の表示機能に関しては, 文字の領域の背景は基本的に透明であり, 映像の上に重ねて表示し, さらに, 文字には背景との区別のために境界を持つものとする. 複数のテロップの同時表示を可能とする.

2.2 VCML文書

VCML文書はヘッダとボディから構成する. ルートエレメントを<vcml>とする.

2.2.1 文書のヘッダ

<head>エレメントは, 時間や表示内容に関係しない情報を記述する.

- 文書に関する情報
  - <title> タイトル
  - <author> 著者
  - <date> 作成日時
  - <copyright> コピーライト
  - <abstract> 概要
- 画面に関する情報
  - <rootlayout/>
    描画する画面のサイズを設定
  - <region/>
    図1に示すように画面の構成を設定し, 名前を付ける. リージョンは描画外, 又は内外をまたいで設定はできない.

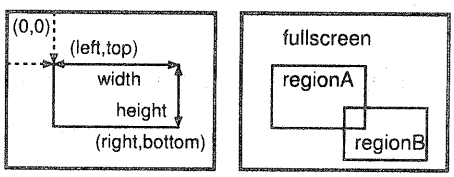

図 1: リージョン (範囲) の定義

2.2.2 文書の本体

<body>エレメントは, 表示に関する時間や表示するオブジェクトなどの情報を記述する. 以下に主なエレメントを示す.

- <time>
  表示時間 (開始時間, 終了時間) を指定する. 子として表示すべきオブジェクトを記述する. ビデオなどのストリームメディアは開始時間で再生開始する. 精度はミリ秒であり, 時間の表現 (Clock) は以下の通りである.

```
Clock ::= clock1 | clock2 | clock3
clock1 ::= (hour "h")? (min "m")? (sec "s")? (msec "msec")?
clock2 ::= (hour ":")? (min ":")? (sec)? ("." msec)?
clock3 ::= [0-9]+("." msec)?  -->単位は秒

hour ::= [0-9]?[0-9]
min ::= [0-5]?[0-9]
sec ::= [0-5]?[0-9]
msec ::= [0-9][0-9]?[0-9]?

例
1h20m30s400msec = 1 時間 20 分 30 秒 400 ミリ秒
1:1:10 = 1 時間 1 分 10 秒
1000.10 = 1000.10 秒
```

- <caption>
  指定されたリージョン内に字幕を表示する. リー

ジョンはHTMLなどのテキストフィールドと同様な働きを持つ．範囲内に文字が収まらない場合は改行して表示し，さらに，レイアウト定義としての ,
, <b>, <i>などを用いて表現する．予め用意されているリージョンとして" top", "bottom", "left", "right", "fullscreen" があるが，その中でも，"left", right" に関しては，縦書きによる表現に用いる．この5つのリージョンはrootlayoutの設定による描画サイズから一定の比率により定義されている．ヘッダで<region/>を用いて再定義する事も可能である．

- <sub>
  <caption>のように指定されたリージョン内に表示するのではなく，"fullscreen" リージョン内における相対座標により字幕を表示する．

- <read>
  <caption>や<sub>内で記述されている文字の読みを与える．読みは文字の上に表示する．

- <font>
  フォントを指定する．HTMLの機能と同一とするが，背景との区別のため境界の有無，境界色の設定を要求する．

- <layout>
  <caption>の表示リージョン，および文書の表示スタイルとしての縦書き，横書きの指定をする．

- <effect> スクロール等の効果を指定をする．

- <video/>
  指定されたリージョン内で指定のビデオの再生を行う．ビデオは<time>によって定義された時間内で再生を行うが，ビデオの長さがそれに満たない場合は，ビデオを停止する．ビデオの開始時間と終了時間は定義できるものとする．

- <img/>
  指定されたリージョン内，または相対座標上に画像の表示を行う．

- <simg/>
  聴覚障害者向けの環境音の画像表示を行う．<img/>は画像ファイルを指定するに対し，<simg/>は環境音のシンボル名を指定する．

- <chapter/>
  プレーヤによるチャプタースキップを行うためにチャプター (章) の登録を行う．

2.3 VCML文書の作成

[5] によるシステムに，字幕のテキストとそれに関する映像の音声部分を入力をすることによって，時間情報 (表示開始時間，表示終了時間) を得る．これをVCML文書に変換する．これに関しては4節において述べる．VCML文書の例を以下に示す．

```
<vcml>
  <head>
    <title>UNTITLED</title>
  </head>
  <body>
    <time begin="0">
      <video src="test.avi">
    </time>
    <time begin="5" end="10">
      <caption>This is Test.</caption>
    </time>
    <time begin="12" end="20">
      <caption>Test is over.</caption>
    </time>
  </body>
</vcml>
```

3 VCML Playerの開発

Windows 98において，"Microsoft Visual C++6.0" とIBMによる"XML for C++"[6] を使用し，VCML文書再生システム，"VCML Player" を作成した．VCML Playerの再生動作確認環境はPentiumII-450MHzのCPU, 128MBのメインメモリ，8MBのビデオメモリ，Windows 98を搭載したPCである．

3.1 VCML文書の読み込み

2.3節の例であるVCML文書を [6] のDOM Parserを用いて文法的解析を行った．結果を図2に示す．この木構造で表されるデータを更に解析し，実装可能なデータに納める．

手順 1: 木のルートとしての<vcml>から，検索を開始する．

手順 2: <head>内に記述されているデータをタイトルや著者情報は1つにまとめ，<rootlayout/>から描画サイズの設定を行い，<region/>では設定されたリージョンを登録する．

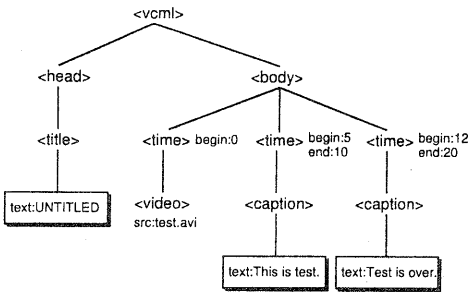

図 2: 2.3節の VCML 文書の解析結果 (木構造)

手順 3: <body>のアトリビュート情報 (基本フォントの種類, 色, 境界の設定) を初期値として設定

手順 4: <time>のアトリビュートの開始時間, 終了時間を用い, イベントデータを作成. <time>の子ノードにおける情報を字幕についてはレイアウト, フォント, テキスト情報をまとめたデータとしてイベントデータに追加する. ビデオや画像に関しても同様に位置などの情報をまとめて追加する. 子ノードの検索が終了したら, 図 3 の示すように, 開始時間による昇順になるようにイベントテーブルに追加する.

手順 5: すべてのノードを検索するまで手順 4 を繰り返す.

3.2 再生部

"VCML Player" の再生部は主に 2 つの部分から構成されている. 1 つはイベントエントリー部であり, もう 1 つはイベント実行部である. イベントエントリー部は VCML 文書を解析して得られたイベントテーブル内のイベントデータを, 再生開始からの処理時間を参照してアクティブイベントリストに追加し, さらにアクティブイベントリストに追加されたイベントが有効時間外になったらリストから削除を行う. イベント実行部はアクティブイベントリストのイベントデータについて表示処理を行う. この 2 つは独立したスレッドのもとで動作する.

3.3 課題

実際に再生を行うと, プレーヤ内では様々な処理動作が行われる. 表 1 に示すように性能の低い環境や

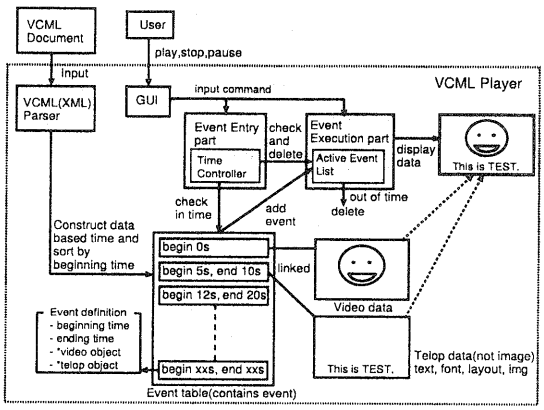

図 3: VCML Player の動作

表 1: 表示タイミングのズレ F: frame per second

対象	時間のズレ (ミリ秒)
字幕	$1000/F$
画像	ロードに要した時間 $+ 1000/F$
ビデオ	再生するまでの時間 $+ 1000/F$

重い処理を要する内容の表示等が原因で, 実際に表示されるべき時間からのズレが必ず発生する. 字幕や画像は表示開始から多少ずれてしまっていても表示する内容 (テキスト等) の変化はない. ところがストリームメディアは, 表示のタイミングが再生開始の時間となる. この時ズレが発生すると, このメディアはずれたまま再生することとなる. よって, ストリームメディアはロードから再生までの間のズレを補正する必要がある.

4 字幕表示時間情報の決定

ここでは, 図 4 に示す時間整合部について述べる. 入力としてビデオデータ中の音声と発話内容 (表示すべき字幕を含む) の書かれた台本を用いる. ビデオデータは映像と音声部分から構成されているが, ここでは音声のみを入力として使用する. 入力音声は短時間のフレーム毎に特徴抽出され, 特徴ベクトル系列に変換される. 一方, 台本は漢字 → かな変換, かな → 音素変換, 音素 → 特徴ベクトル変換により特徴ベクトル系列へと変換される. 入力音声から変換された特徴ベクトル系列と台本から変換された特徴ベク

トル系列は, DP マッチング (DTW: Dynamic Time Warping) によってこれらの時間軸整合が行われる. 以上により与えられた字幕の表示と消去する時間情報が決定される. 予備実験 [5] においてクリーンな音

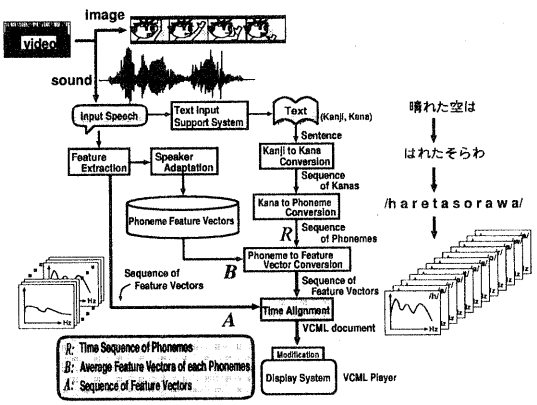

図 4: 時間軸整合部の処理の流れ

声データに対する時間軸整合部の評価を行った. 音素毎の正解データ (ラベル) とのずれは平均で 39.0 msec であった. 本報告では, 実際のビデオデータデータから取り出した音声部分を実験に用い評価を行う.

4.1 実験と評価方法

実験用のビデオデータとして" 会津大学 2000 年入学案内" のビデオプログラムを用た. PC に接続したビデオキャプチャボードにビデオデッキを接続し, ビデオデータを作成する. ビデオデータのフォーマットは RGB 15bit (320x240pixel, 30 フレーム/秒, ステレオ 44.1kHz) である. ビデオデータから音声部分を抽出し, raw フォーマット (モノラル 12kHz) に変換する. LPC ケプストラム分析によって音声部分の特徴抽出を行い, 21.3 msec 毎に 17 次元の特徴ベクトルに変換する. 使用した音声の分析条件を表 2 に示す. 学長講話部分のデータ (67.4 秒, SNR:23.7dB) を実験に用い, 字幕表示タイミングのラベルとのずれを評価した. 字幕表示タイミングは in-time と out-time から構成されており, in-time は字幕をディスプレイに表示し, out-time は字幕を消すタイミングである. 使用したデータに対する in-time, out-time の正解 (ラベル) は人手によって作成した. 漢字仮名まじり文で書かれている台本から kakasi を用いて読みに変換し, さらに読み誤り等を手で修正した後, 音素記号列に変換した. 台本から変換された音素数は 541, in-time と out-time の総数は 18ヶ所である.

表 2: 音声の分析条件

音声特徴量	LPC ケプストラム係数 + power, 17 次
標本化周波数	12 kHz
LPC 分析	14 次
LPC ケプストラム分析	16 次
窓フレーム長	21.3 ms (256 点)
フレーム周期	21.3 ms (256 点)
窓関数	ハミング窓
高域強調	$(1 - 0.97z^{-1})$
話者	男性 1 名

4.2 実験結果

実験結果を表 3 に示す. 字幕表示単位におけるラベルとの最小の時間差は 1.1 msec, 最大で 360.5 msec, 平均は 53.5 msec であった. 結果は時間軸整合部が良好に動作していることを示している. また, 67.4 秒の音声データに対する処理時間は PentiumIII-500MHz の CPU, 128MB のメインメモリを搭載したマシンを用いた場合, 1.16 秒であり高速な処理が可能である.

表 3: 正解と時間軸整合部の結果のずれ

	時間差
最小	1.1 msec
最大	360.5 msec
平均	53.5 msec

5 むすび

時間軸整合を使用した表示タイミングの決定, VCML 文書の変換により字幕の自動作成を可能にした. VCML 文書を簡単に編集するためオーサリングソフトの開発が必要であろう.

参考文献

[1] T. Bray, J. Paoli, C.M. Sperberg-McQueen, editors, Extensible Markup Language (XML) 1.0, W3C, Feb. 3, 1998.

[2] S. Bugai, D. Bulterman, Synchronized Multimedia Working Group, Synchronized Multimedia Integration Language (SMIL) 1.0, W3C, Jun. 15, 1998

[3] 小林 彰夫, 他, "ニュース音声認識システムの検討," 音響学会講演論文集, 3-1-9, pp.103-104 (1997-9).

[4] 白石 克彦, 他, "視聴覚障害者向け放送ソフト製作技術研究開発プロジェクトの研究状況," 通信・放送機構 視聴覚障害者のためのテレビ用字幕製作に関する国際ワークショップ論文集, pp.7-30 (Nov. 1999).

[5] K.Watanabe, M.Sugiyama, Time Alignment between Caption and Acoustic Signal for Automatic Caption Generation, Technical Report of Speech Processing Research Committee, SP99-27, pp.7-14(Sep. 1999)

[6] XML Parser IBM IBM http://www.alphaworks.ibm.com/Home/, "XML for C++ 3.0.0", 1999

付録

```
<!-- DTD of Video Caption Markup Language(VCML) -->
<!-- file: vcml.dtd -->

<!-- The Document VCML -->
<!ELEMENT vcml (head?,body?)>

<!-- The Document Head -->
<!ELEMENT head (title?,author?,date?,copyright?,abstract?,
rootlayout?,region*)>

<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT copyright (#PCDATA)>
<!ELEMENT abstract (#PCDATA)>

<!-- region Entity -->
<!ENTITY % rect "
left CDATA #REQUIRED
top CDATA #REQUIRED
right CDATA #IMPLIED
bottom CDATA #IMPLIED
width CDATA #IMPLIED
height CDATA #IMPLIED ">

<!-- Root Layout Element -->
<!ELEMENT rootlayout (region*)>
<!ATTLIST rootlayout
width  CDATA #REQUIRED
height  CDATA #REQUIRED >

<!-- Region Element -->
<!ELEMENT region EMPTY>
<!ATTLIST region
id CDATA #REQUIRED
%rect; >

<!-- Document Body -->
<!ELEMENT body ANY>
<!ATTLIST body
fgcolor CDATA #IMPLIED
bbcolor CDATA #IMPLIED
bgcolor CDATA #IMPLIED
bbtype CDATA #IMPLIED >

<!-- Time Element -->
<!ELEMENT time ANY>
<!ATTLIST time
begin CDATA #REQUIRED
end CDATA #IMPLIED >

<!-- Caption Element -->
<!ELEMENT caption ANY>

<!-- for caption field -->
<!-- Center Element -->
<!ELEMENT center ANY>

<!-- Br Element -->
<!ELEMENT br EMPTY>

<!-- B Element BOLD font -->
<!ELEMENT b ANY>

<!-- I Element Italic font -->
<!ELEMENT i ANY>

<!-- Sub(Title) Element -->
<!ELEMENT sub ANY>
<!ATTLIST sub
x CDATA #REQUIRED
y CDATA #REQUIRED >

<!-- Read Element -->
<!ELEMENT read (#PCDATA)>
<!ATTLIST font text CDATA #REQUIRED>

<!-- Font Element -->
<!ELEMENT font ANY>
<!ATTLIST font
type CDATA #IMPLIED
size CDATA #IMPLIED
color CDATA #IMPLIED
bbcolor CDATA #IMPLIED
bbtype CDATA #IMPLIED >

<!-- Layout Element -->
<!ELEMENT layout ANY>
<!ATTLIST layout
pos CDATA #IMPLIED
style CDATA #IMPLIED >

<!-- Effect Element -->
<!ELEMENT effect ANY>
<!ATTLIST effect
type CDATA #REQUIRED
speed CDATA #IMPLIED >

<!-- Video Element -->
<!ELEMENT video EMPTY>
<!ATTLIST video
src CDATA #REQUIRED
id CDATA #IMPLIED
begin CDATA #IMPLIED
end CDATA #IMPLIED >

<!-- Image Element -->
<!ELEMENT img EMPTY>
<!ATTLIST img
src CDATA #REQUIRED
id CDATA #IMPLIED
left CDATA #IMPLIED
right CDATA #IMPLIED >

<!-- Sound Image Element -->
<!ELEMENT simg EMPTY>
<!ATTLIST simg name CDATA #REQUIRED>

<!-- Chapter Element -->
<!ELEMENT chapter EMPTY>
<!ATTLIST chapter name  CDATA #REQUIRED>
```