マルチメディア・モンタージュ ——対位法に基づいた映像合成に関する研究

Multimedia Montage ——A Study on the Counterpoint Synthesis of Movies

鈴 木 良太郎

要 約 本稿では，イメージに基づいたコミュニケーションの可能性を広げるための枠組みとして，映像や音声によるマルチメディア素材を時空間上で構造的に合成する「マルチメディア・モンタージュ（Multimedia Montage）」を提案する．本研究における映像合成手法の特徴は，その構成手法として西洋古典音楽における対位法を導入し，またその構造をスクリプト言語を用いて記述する点にある．対位法に基づいた合成映像の事例の作成，映像合成用プロトタイプシステムの開発，プロトタイプにおける映像合成実験を通して，本合成手法がマルチメディアを用いたイメージ表現の手法として有効である事を確認する事ができた，また映像合成実験における同期認識の観察結果に基づき，映像から抽出したリズム情報を利用して映像の同期合成を行う研究“Image Wave”を今年度から開始した．

Abstract In this paper, the author propose “Multimedia Montage”, which is the structural synthesis of multimedia contents, such as movies and sounds, in time and space, for the purpose of communications by images. In this study, the author introduce the counterpoint theory in music to compose movie structures and use a script to describe the structure. By making example movies based on counterpoint theory, developing a prototype system, and making a movie synthesis experiment using the prototype, the author confirm the effectiveness of our counterpoint method in image expression using multimedia. In addition, from this year, the author started a new study named “Image Wave” that synchronize movie clips utilizing the rhythm information extracted from the clips. The concept of this new study is based on the observation of the movie synchronization recognition in the movie synthesis experiment.

1. はじめに

筆者が現在所属する(株)ATR知能映像通信研究所第三研究室では，マルチメディアを利用したイメージ表現とそれに基づいたコミュニケーションの様々な可能性について，研究を進めている．筆者は，対位法に基づいたマルチメディア素材の新しい構成技法として，97年度よりマルチメディア・モンタージュ（Multimedia Montage）[1]~[11]の研究を行ってきた．

本稿では，まずマルチメディア・モンタージュの概念と対位法の導入意義について説明する．次に，対位法に基づいた映像合成のためのプロトタイプシステムについて述べる．最後にプロトタイプシステムによる映像合成実験の結果とそれに基づいた同期合成に関する新たな研究について，報告する．

2. マルチメディア・モンタージュと対位法的映像

2.1 マルチメディア・モンタージュ

今日の情報処理技術の目覚しい発達は，多様な自己表現とそれに基づくコミュニケーションの新たな可能性を我々に提示している．そこで実際に，様々なマルチメディア素材を合成して何らかのイメージを表現しようとする時，それらの素材を合成する構成方法の在り方が問題となる．

筆者はこのようなマルチメディアの構成方法の問題に着目し，イメージ表現のために多様なマルチメディア情報を効果的に合成し得る構成方法を検討する事とした．素材の空間配置による構成技法としては，コラージュ，アサンブラージュ等があるが，特にマルチメディアを対象とする場合には，動画像，音声等の時間的な素材を扱う必要がある．筆者は，このような時空間上のマルチメディア情報を構造的な管理の下で合成する概念を「マルチメディア・モンタージュ（Multimedia Montage）」と名付け，それに基づいたイメージ表現に関する研究を行った．

研究の当面の目標は，このようなイメージ表現の装置を映像合成装置として構築する事とし，対象とするマルチメディア素材としては，動画像，静止画像，音声を考察の対象とした．なお，本研究において，映像とはこれらのマルチメディア素材が合成された総体を指す．

2.2 エイゼンシュテインのモンタージュ理論

イメージ表現のための構成技法は，映像の世界ではモンタージュ理論の中にその典型を見出す事ができる．モンタージュ理論は，エイゼンシュテイン，クレショフ，ヴェルトフ等の旧ソビエトの映画監督によって主に 1920 年代に研究された．モンタージュ理論とは，当初はショットの組み合わせを如何に構成すべきかという編集の理論であった．今日でも，短いクローズアップのショットの組み合わせの挿入による編集技法そのものを指してモンタージュと呼ぶ事が多い．

本研究におけるモンタージュという言葉は，特にエイゼンシュテイン（S. M. Eisenstein）のモンタージュ理論に基づいている．1930 年以降，トーキーやさらにカラー映画が普及してくると，エイゼンシュテインは，モンタージュ理論を単なる編集の理論からよりマルチメディア的な統合理論へと発展させて行った．エイゼンシュテインによるモンタージュ理論の分類を彼の著作「映画における四次元」[12]と「垂直のモンタージュ」[13]に基づいてごく簡単に整理すると，次のようにまとめる事ができる．

1） メトリック（韻律的）モンタージュ　←ショット長
2） リズミック（律動的）モンタージュ　←動き
3） トーン（音調）モンタージュ　←色彩/明度
4） オーバートーン（倍音）モンタージュ←それらの合成
5） 知的オーバートーンモンタージュ　←記号，意味
6） 垂直モンタージュ　←音声

このように，エイゼンシュテインは映像を構築するための全ての構成要素の統合化理論としてモンタージュ理論を位置付けている．

ここでさらに注目に値するのは，「垂直のモンタージュ」において，彼が対位法を取り上げている事である．エイゼンシュテインは，映像（の動画像）と音声との問題

について，音声（特に音楽）は映像の単なる従属物ではなく，両者の関係はそれぞれが自律的な対位法的関係でなければならない事を強調した．

2.3 対　位　法[14]～[17]

筆者等はこの「対位法」という言葉に着目し，その構成技法としての有効性に関する考察を行った．より一般的には，「対位法」という言葉は二つの意味を持っている．音楽の専門用語としてでなく対位法という言葉が用いられる場合には，二つの要素を対比的，対照的に並置する事によって，ある事象を強調する表現技法を指す．例えば，映画においては，悲しい場面に明るい曲を流す事によって，逆に悲しさを強調するような技法である．

一方，西洋古典音楽における対位法の意味は，そのような捉え方とは微妙に異なる．通常，対位法という言葉は和声法の対照概念として扱われる．多声音楽において，時間軸を水平方向とした時，垂直方向の音の重なり関係をより重視するのが和声法であり，それに対して，水平方向の各声部の線的独立性をより重視するのが対位法である．但し，実際にはこれらは対立する概念ではなく，両者のバランスの下で総体としての音楽が成立していると言えるだろう．

特にマルチメディアの構成技法としての応用の可能性という観点から見た時，音楽理論としての対位法の特徴を次のように整理する事ができる．

1） それぞれが独立した複数の自律的な声部による構成
2） 対応する複数の音符の間の時間的関係
3） 時間的に一致する音符間の和声関係
4） 主題（dux）と応答（comes）
5） 模倣における旋律の変換（図 1）

対位法の最大の特徴は，それぞれが自律的な複数の要素を時間的に並置する事に基づいて，ある全体を構成するという事であり，それが対位法の根底を成す最も重要な概念であると考えられる．

2.4 対位法的映像

対位法の持つ特徴は，イメージ表現のための一般構成技法として高い応用力を持っている．またさらに，今日のマルチメディア情報処理技術との整合性も高い．そこで本研究では，以下に示すような形で，対位法をマルチメディア・モンタージュの映像合成に適用する事にした．

1） 自律性を持った複数の映像要素の合成
2） 合成される映像要素の時間的同期
3） 同期対象となる映像要素間の表現の整合化
4） 主題の引用，反復による映像要素間のコミュニケーション
5） 映像要素の引用における移動，拡大，縮小，等の時間変換操作

本研究では，このようにして対位法に基づいて合成された映像を「対位法的映像（counterpoint movie）」と呼ぶ．

3. プロトタイプシステム

以上の映像合成に対する対位法の適用に関する考察に基づいて，対位法的映像の構

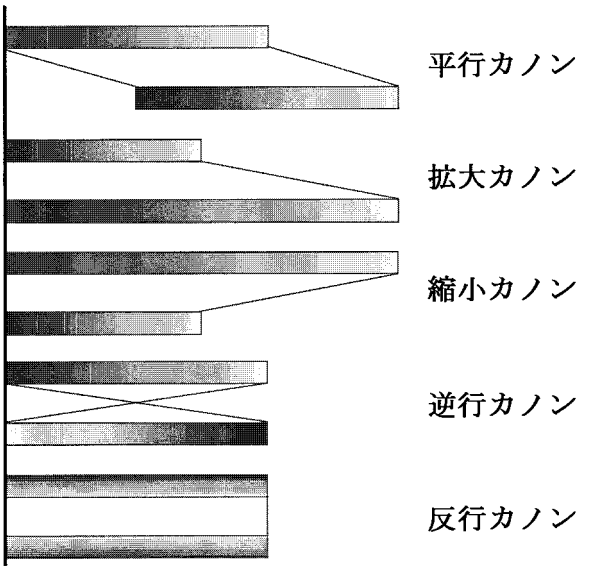

図 1　カノンのバリエーション

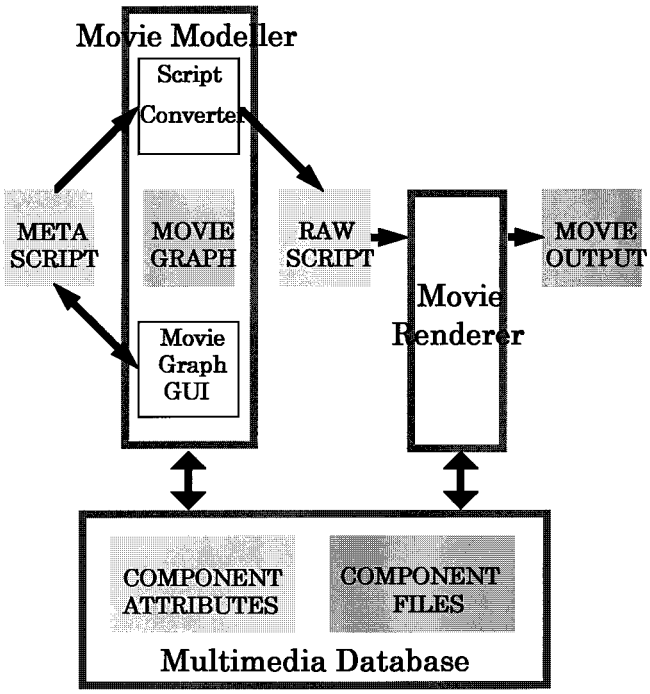

図 2　プロトタイプのシステム構成

造を記述するためのスクリプト言語として Meta Script を考案し，その記述に基づいて映像合成を行うプロトタイプシステムを開発した．

3.1 システム構成

以下のモジュールから構成される（図 2）．

1） Movie Modeller

Meta Script の内部表現形式である Movie Graph を用いて，対位法的映像の構造を定義する．Movie Graph GUI と Script Converter から構成される．前者は，GUI による Movie Graph の編集と，その Meta Script との相互変換を行う．後者は，Meta Script を実際の映像合成の実行環境に依存した表現形式である Raw Script に変換する．

2） Movie Renderer

Raw Script の記述に基づき，Multimedia Database から必要なコンテンツを参照して，映像を合成する．

3） Multimedia Database

静止画像，動画像，音声，等のコンテンツおよびそれらの属性情報のテーブルから成るデータベース．コンテンツのファイル名と属性情報を管理，編集するための GUI が提供される．属性情報は，Movie Modeller により参照され，コンテンツの検索に使用される．

Movie Renderer 以外のモジュールは JAVA で記述されており，各種の PC やワークステーション上での実行が可能である．一方，Movie Renderer とその入力である Raw Script は機種依存であり，ソースコードレベルでも互換性を持たない．Movie Renderer の初期バージョンとしては，1998 年度に SGI の O 2 ワークステーションの Movie Library を利用したバージョンが，まず開発された．本稿における映像合成実験は，同バージョンを用いて行った．その後，汎用化と合成精度の向上を目的として，1999 年度から QuickTime[18] の SDK（まず MAC 版）によるバージョンの開発を進めている．同バージョンが完成すれば，QuickTime（Ver. 3 以降）に対応する各種 PC 上で本システムが利用可能になる．

3.2 Meta Script

Meta Script は合成映像の構造を階層的なノードの集合として記述したものである．図 3 に Meta Script の実例を示す．これは，次章に対位法的映像の事例として示す“家族ゲーム・ゲーム”の代表的なシーンにタイトルショットとエンドタイトルショットを追加したものである．

Meta Script の各ノードは，ユーザが定義する独自のノード名によって識別される．またその種類ごとにノード属性を持つ．ノードは一般に次のように記述される．

```
ノード種別名（ノード名）{
    ノード属性の定義
    子ノードの定義
}
```

ノード種別には，以下のものがある．

・Movie Node　：　ノード種別名“Movie”

・Scene Node　：　ノード種別名“Scene”
・Shot Node　：　ノード種別名“Shot”
・Symbol Node　：　ノード種別名“Symbol”
・Define Node　：　ノード種別名“Define”

さらに，ノード属性は次のように記述される．（省略時にはデフォルト値が採用される．）

ノード属性名（属性値）

以下に各ノード種別ごとの仕様を示す．

・Movie Node

合成映像全体のルートノードとして必ず一つ存在する．子ノードとして，Define Node，Scene Node，Shot Node を持つ事ができる．Movie Node としての属性は持たない．

・Define Node

他で参照するためのノードの定義を行う．定義する子ノードとして，Scene Node, Shot Node, Symbol Node を持つ事ができる．Define Node はノード名を持たない．また，唯一のノード属性として，以下の属性を持つ．

offset　：　オフセット名とオフセット値

offset 属性により定義された offset 名を Scene Node や Shot Node の start_time 属性（後述）の値として指定する事ができる．オフセット名に対応するオフセット値が Define Node の親ノードのスコープ内の開始時刻情報として参照される事により，独立したノード間のセミグローバルな同期関係を記述できる．

Define Node を用いない場合も，既に定義されたノードは，それをノード名により参照する事ができる．どちらの場合も，参照されるノードの属性と下位構造が継承される．また，その任意の属性を再定義する事も可能である．

・Scene Node

子ノードとして Scene Node 自身を持つ事により映像を階層的に構成する．また，Shot Node, Define Node を子ノードとする事ができる．Scene Node は以下の属性を持つ．

Timing	：	SEQUENTIAL/PARALLEL
start_time	：	開始時刻のオフセット時間
duration	：	再生時間
speed	：	再生速度倍率
reverse	：	反転の有無
repeat	：	反復回数
mix	：	画像合成モード

Scene Node において特徴的なのは timing 属性であり，子ノードの時間的順列/並列関係を決定する．それ以外の属性は，Shot Node と共通な属性であり，start_time から repeat までは，そのノードに対応する映像要素に対する時間変換のパラメータを表す．

mix 属性は，そのノードに対応する（音声以外の）映像要素が他の映像要素と時間

的に並列な関係になる時，それをどのように合成するかを示す属性である．現在の仕様では，OVERLAY と CHROMAKEY の何れかを指定できる．また mix 属性を Multimedia Database の映像コンテンツに対して指定する事も可能であり，双方で指定した場合には Meta Script の指定を優先する．

・Shot Node

映像の階層構造における最小の集合単位であり，最下位の Scene Node としての働きをする．子ノードとして Symbol Node を持ち，それによって具体的映像コンテンツと結び付けられる．Shot Node は timing 以外の Scene Node と共通の属性を持つ．その他，さらに以下の属性を持つ．

Type ： コンテンツ種別（動画像等）

file_name ： コンテンツファイル名

指定した type 属性に該当する種類のコンテンツがデータベースから参照される．コンテンツのファイル名を直接指定する場合には，file_name 属性を使用する．

・Symbol Node

映像を構成する抽象的なシンボルを表す．最下位のノードであり，子ノードを持たない．以下の属性を持つ．

casting ： 割当て名

casting 属性は，配役等のコンテンツの割り当てを表す．casting 属性を変更する事により，定義した映像単位に使用するコンテンツを切り替える事ができる．また，色やカメラアングル等，使用するコンテンツの任意の特性を Symbol Node を用いて抽象的に記述する事が可能である．

3.3 Movie Graph

Movie Graph は Meta Script のプログラム内部形式であり，ノードの階層構造がツリーグラフによって表現される．図 4 に，図 3 の Meta Script 例の Movie Graph を示す．（便宜上，図 4 の表記形式そのものを Movie Graph と呼ぶ場合もある．）

図 4 において，あるノードの右下に接続するノードは，それが第一子である事を表す．また横方向に接続するノード列は子ノード間の順列関係を表し，縦方向に接続するノード列は子ノード間の並列関係を表す．これらの関係は，実際にプログラム内部では，階層化されたノードのリンクと順列/並列のモードを表すフラグがあれば実現できるが，それをこのように図示する事により，ノードの階層構造と順列/並列関係とを同時に把握する事ができる．Movie Graph GUI では，実際にこの図表現を用いて Movie Graph の編集を行う．

4. 実験と考察

4.1 対位法的映像の合成実験

Meta Script による対位法構造の記述能力とプロトタイプシステムの映像合成能力を検証するために，実際に対位法的映像の合成実験を行った．実験の対象としては，“Dance Canonica”，“INVENTION−Happy＋Sad−”，“家族ゲーム・ゲーム”の 3 つの映像作品を使用した．これらは，対位法的映像の実例として，市販の映像編集ソフ

```
#
#     Meta Script Sample
#    "Kazoku Game Game"
#
Movie(Game1) {
# Main Casting
 Define {
  Symbol(tutor) {
   casting(Yonei)
  }
  Symbol(father) {
   casting(Suzuki)
  }
  Symbol(mother) {
   casting(Uemura)
  }
 }
# Title
 Shot(Title) {
  file_name(Title1.tif)
  duration(5.0)
 }
# Dining Scene
 Scene(Dining) {
  timing(PARALLEL)
#  Background
  Scene(Background) {
#   Part 1 : Kitchen Unit
    Shot(Kitchen) {
      type(STILL)
      duration(40.0)
      Symbol(kitchen)
    }
#   Part 2 : Beach Scene
    Scene(BEACHandBGM) {
      timing(PARALLEL)
#     Beach Image
       Scene(Beach) {
        Shot(Banana_Boat) {
          type(SILENTMOVIE)
          duration(20.0)
          Symbol(ocean)
          Symbol(banana_boat)
        }
        Shot(Girls_Middle_Shot) {
         type(SILENTMOVIE)
         duration(20.0)
         Symbol(ocean)
         Symbol(girls)
        }
        Shot(Girl_Zoom_Shot) {
         type(SILENTMOVIE)
         speed(0.5)
         duration(20.0)
         Symbol(ocean)
         Symbol(girl)
        }
       }
#    BGM
      Shot(BGM) {
       type(SOUND)
       duration(60.0)
       Symbol(BGM)
      }
    }
#   Part3 : No Background (IMPLICIT)
  }
# Foreground
  Shot(Foreground) {
   type(SOUNDMOVIE)
   Symbol(tutor)
   Symbol(father)
   Symbol(mother)
   Symbol(younger_child)
   Symbol(elder_child)
   Symbol(long_table)
  }
 }
# End Title
 Shot(End) {
  file_name(End1.tif)
  duration(5.0)
 }
}
```

図 4 “家族ゲーム・ゲーム”の Movie Graph

トウェア（Adobe　Premiere）を用いて作成したオリジナル作品である．それと全く同じ内容の映像をプロトタイプシステムによって新たに合成し，比較した．以下にその結果についての考察を行う．

4.1.1 Dance Canonica[5],[8]

“Dance　Canonica”は，カノンにおける様々な変換のバリエーション（図 1）に基づいた映像合成への対位法の応用の最も基本的な実例である．筆者が撮影したダンスのビデオ映像を素材として，同一の映像を開始時刻や再生速度を変えてオーバーレイする事により，平行カノン，拡大カノン，逆行カノン，縮小カノン，逆行反行カノンを映像化した（図 5）．

映像合成に用いたオーバーレイは，複数の映像が重なって見える点を特徴とし，クロマキーと同様によく用いられる手法である．スーパーインポーズ等，他の用語で呼ばれる場合もある．元来，フィルムの多重露光により実現されていたが，コンピュータ上ではそれが輝度合成等の画像処理に置き換えられる．Premiere ではそのキー合

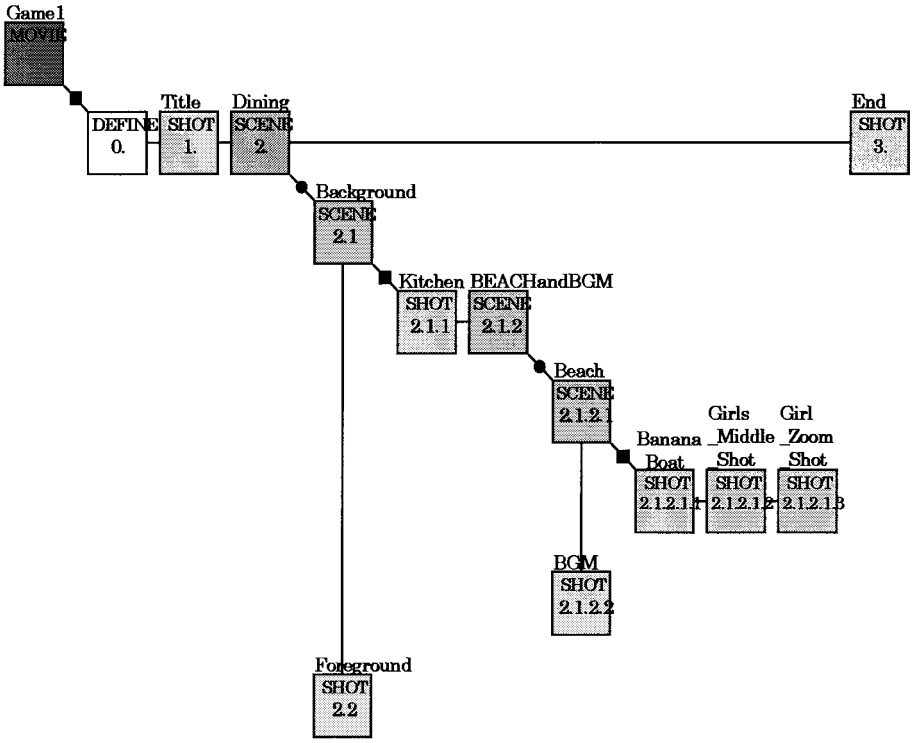

図 4 “家族ゲーム・ゲーム”の Meta Script

図 5 Dance Canonica（下段がプロトタイプの出力）

成機能の一つとしての輝度合成機能を利用したのに対して，プロトタイプでは合成モードとして OVERLAY を指定した．これも Movie Renderer の内部処理としては輝度合成を行うが，Premiere の輝度合成機能とは仕様の詳細が異なるため，厳密な意味では同一の合成映像とはならない．

O 2 版の Movie Renderer の問題としては，画像圧縮形式として JPEG 圧縮しか使えず，その圧縮/伸長を繰り返す事により，画質が劣化してしまうという問題が上げられる. 図 5 でも, プロトタイプの出力の方が鮮明度が低い. この点に関しては, QuickTime 版において改善される予定である.

Meta Script による映像の構造の記述とそれに基づいた合成メカニズムに関しては問題はなく，対位法構造の時間変換機能としての基本的な要求条件を Meta Script の仕様が満たしている事が確認された（図 6）.

4.1.2 INVENTION-Happy+Sad-[4],[7],[11]

“ INVENTION-Happy+Sad- ”は，Counterpoint Template（対位法テンプレート）の実例として作成した映像作品である．Counterpoint Template とは対位法的映像を作成するための構造の雛形となるテンプレートであり，典型的な対位法楽曲の構造を Meta Script により記述する．同一の構造の下で，どのような映像素材を適用するかによって，多様な映像表現が可能である．また，その構造の一部を変更する事によって，多様な構成を実現できる.

“ INVENTION-Happy+Sad- ”では，バッハの 3 声のインベンションの第 4 番[17]を元に，その構造を Meta Script で記述し（図 7），その構成要素となる旋律をダンスの基本動作（歩行，跳躍，回転）の映像によって置き換えた．各声部の映像において，主題とダンスの基本動作とが結び付けられ，それが模倣される．これをさらに「楽しい」,「普通」,「悲しい」という感情表現と結び付けて，映像素材を組み合わせる事により，具体的な感情表現によって構成されるイメージの構築を行った（図 8）.

今回使用した楽曲の形式はフーガ（フゲッタ）に属し，主題提示部，展開部，終局部の 3 部から構成される．特にこの楽曲の場合は，まず高音部と低音部が同期して始まり，次に展開部の始まりでは中音部と低音部が同期し，終結部の始まりでまた高音部と低音部が同期するという特徴を持つ．各声部がぴったり同期したり微妙にずれ合ったりしながら楽曲が進行し，一番最後に楽曲が終了する時になって始めて 3 声全部が完全に同期する．このような構造を表現するために，Define Node の offset 属性を利用してグローバルな同期タイミングを定義し，それを各声部の Scene Node で引用するように Meta Script を記述した．この offset 属性の機能は，対位法楽曲の分析に基づいて新たに追加したものである．この機能によって，独立した声部間の自由な同期関係を記述できるようになった.

本映像作品の市販ソフトウェアによる合成には，Adobe 社の After Effect の輝度合成機能を使用した．一方，プロトタイプでは，“ Dance Canonica ”と同様に OVERLAY モードによる輝度合成を行った．結果は，テクスチャ的にもほぼ同様であり，目立った違いはなかった. “ INVENTION ” の映像合成では Meta Script の記述通りに正確にオフセット時間と再生速度の変換が行われる事が重要であるが，O 2 版の Movie Renderer はその精度が充分ではない．この点についても，QuickTime 版では改善される予定である.

4.1.3 家族ゲーム・ゲーム[6]

対位法的映像の応用編として作成した実例である．映画「家族ゲーム」のパロディ・シーンを素材に，その様々な合成バリエーションが，対比，模倣，同期，反復，等の

```
#
#    Meta Script Sample
#    "Dance Canonica"
#
Movie(DanceCanonica) {
# Main Contents
  Define {
    Shot(Dance) {
      type(SILENTMOVIE)
      Symbol(dancer) {
        casting(MarikoFukushima)
      }
      Symbol(color) {
        casting(normal)
      }
    }
    Shot(DanceInverse) {
      type(SILENTMOVIE)
      Symbol(dancer) {
        casting(MarikoFukushima)
      }
      Symbol(color) {
        casting(inverse)
      }
    }
  }
# Dance Variations with Sound
  Scene(DanceAndSound) {
    timing(PARALLEL)
# Movie Part
    Scene(DanceMovie) {
# Main Title
      Shot(MainTitle) {
        file_name(MainTitle.tif)
        duration(4.0)
      }
# Parallel Canon Scene
      Shot(ParallelTitle) {
        file_name(ParallelTitle.tif)
        duration(2.0)
      }
      Scene(Parallel) {
        timing(PARALLEL)
        Shot(Dance) {
          type(SILENTMOVIE)
          duration(84.0)
        }
        Shot(Dance) {
          type(SILENTMOVIE)
          duration(84.0)
          start_time(1.0)
        }
      }
# Augmentation Canon Scene
      Shot(AugmentationTitle) {
        file_name(AugmentationTitle.tif)
        duration(2.0)
      }
      Scene(Augmentation) {
        timing(PARALLEL)
        Shot(Dance) {
          type(SILENTMOVIE)
          duration(84.0)
        }
        Shot(Dance) {
          type(SILENTMOVIE)
          speed(0.5)
          duration(84.0)
        }
      }
# Retrograde Canon Scene
      Shot(RetrogradeTitle) {
        file_name(RetrogradeTitle.tif)
        duration(2.0)
      }
      Scene(Retrograde) {
        timing(PARALLEL)
        Shot(Dance) {
          type(SILENTMOVIE)
          duration(84.0)
        }
        Shot(Dance) {
          type(SILENTMOVIE)
          duration(84.0)
          reverse(1)
        }
      }
# Diminution Canon Scene
      Shot(DiminutionTitle) {
        file_name(DiminutionTitle.tif)
        duration(2.0)
      }
      Scene(Diminution) {
        timing(PARALLEL)
        Shot(Dance) {
          type(SILENTMOVIE)
          duration(84.0)
        }
        Shot(Dance) {
          type(SILENTMOVIE)
          speed(2.0)
          duration(42.0)
          repeat(2)
        }
      }
# Retrograde&Inversion Canon Scene
      Shot(Retrograde&InversionTitle) {
        file_name(RetroInvTitle.tif)
        duration(2.0)
      }
      Scene(Retrograde&Inversion) {
        timing(PARALLEL)
        Shot(Dance) {
          type(SILENTMOVIE)
          duration(84.0)
        }
        Shot(DanceInverse) {
          type(SILENTMOVIE)
          duration(84.0)
          reverse(1)
        }
      }
    }
# BGM Part
    Shot(DanceBGM) {
      type(SOUND)
      duration(440.0)
      Symbol(DanceBGM)
    }
  }
}
```

図 6 “Dance Canonica”の Meta Script

対位法的概念に基づいて構成される．（図 9）“家族ゲーム・ゲーム”とは，固定的な一映像作品ではなく，これらの多様なバリエーションの総体である．

“Dance　Canonica”や“invention”に比べ，“家族ゲーム・ゲーム”では，より自

```
#
#  Counterpoint Template Sample
#  : Bach 3 Part Invention No.4
#
Movie(invention){
 Scene(Title){
 }
# Main Scene
 Scene(Main){
  timing(PARALLEL)
  Define{
   offset(Development, 45.0)
   offset(Coda, 75.0)
  }
#  Low Voice Part
  Scene(LowVoice){
   timing(SEQUENTIAL)
#   Low Voice Subject
   Scene(Low1){
    timing(SEQUENTIAL)
#    from measure 1 to 3
    Scene(Low1-1){
    }
#    from measure 4 to 8 former half
    Scene(Low1-2){
    }
   }
#   Development of Low Voice
   Scene(Low2){
    timing(SEQUENTIAL)
#    Starting time offset
#    of Development Part
    start_time(Development)
#    from measure 8 latter half
#    to 10 former half
    Scene(Low2-1){
    }
#    from measure 10 latter half
#    to 13 former half
    Scene(Low2-2){
    }
   }
#   Low Voice Coda
   Scene(Low3){
    timing(SEQUENTIAL)
#    Starting time offset of Coda Part
    start_time(Coda)
#    from measure 13 latter half
#    to 15 former half
    Scene(Low3-1){
    }
#    from Following High Voice
#    to Low Voice Subject
    Scene(Low3-2){
    }
#    from measure 18 to 19
    Scene(Low3-3){
    }
#    Low Voice Coda
#    (from Following High Voice)
    Scene(Low3-4){
    }
   }
  }
#  Middle Voice Part
  Scene(MiddleVoice){
   timing(SEQUENTIAL)
   start_time(6.0)
   Scene(Middle1){
    timing(SEQUENTIAL)
#    from 2 to 4
    Scene(Middle1-1){
    }
#    from measure 5 to 8 former half
    Scene(Middle1-2){
    }
   }
   Scene(Middle2){
    timing(SEQUENTIAL)
#             Starting time offset
#    of Development Part
    start_time(Development)
#    from measure 8 latter half to 13
    Scene(Middle2-1){
    }
#    from measure 14 to 16
    Scene(Middle2-2){
    }
#    from measure 17 to 20 former half
    Scene(Middle2-3){
    }
#    from measure 20 latter half to 23
    Scene(Middle2-4){
    }
   }
  }
#  High Voice Part
  Scene(HighVoice){
   timing(SEQUENTIAL)
   Scene(High1){
    timing(SEQUENTIAL)
#    from measure 1
#    to measure 3 head
    Scene(High1-1){
    }
#    from measure 3
#    to measure 5 head
    Scene(High1-2){
    }
#    from measure 5 to 7 former half
    Scene(High1-3){
    }
#    from measure 7 latter half to 8
    Scene(High1-4){
    }
#    from measure 9 to 13 former half
    Scene(High1-5){
    }
   }
   Scene(High2){
    timing(SEQUENTIAL)
#    Starting time offset of Coda Part
    start_time(Coda)
#    from measure 13 latter half
#    to 15 head
    Scene(High2-1){
    }
#    from measure 15 to 20 head
    Scene(High2-2){
    }
#    from measure 20 to 23
    Scene(High2-3){
    }
   }
  }
 }
}
```

図 7　対位法テンプレート“INVENTION”

由な引用と反復が展開される．全体を通して食事のシーンが繰り返されるが，それに使われている映像は必ずしも同一ではない．さらに，各シーンの内部においても，前景と背景との関係は直接的な模倣ではなく，例えば前景の会話の内容に対応して背景

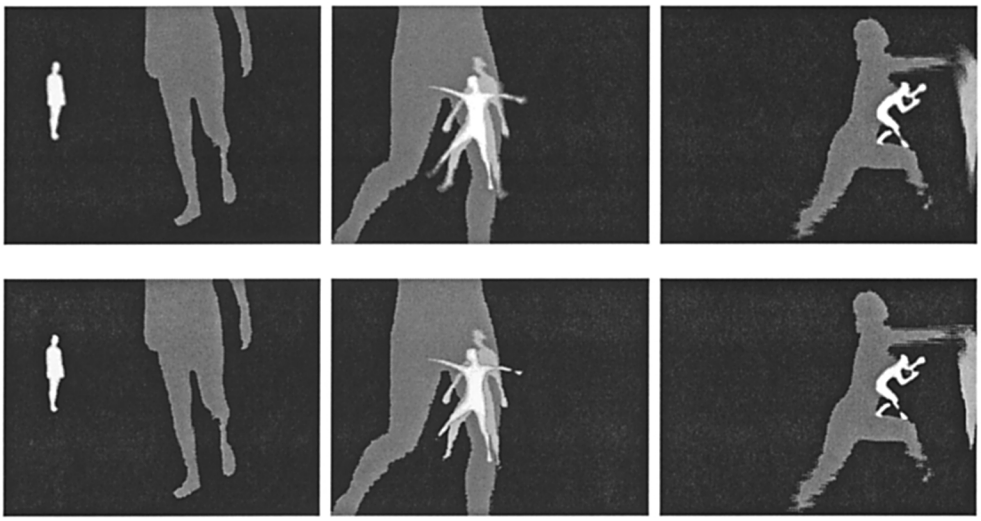

図 8　INVENTION–Happy＋Sad–（下段がプロトタイプの出力）

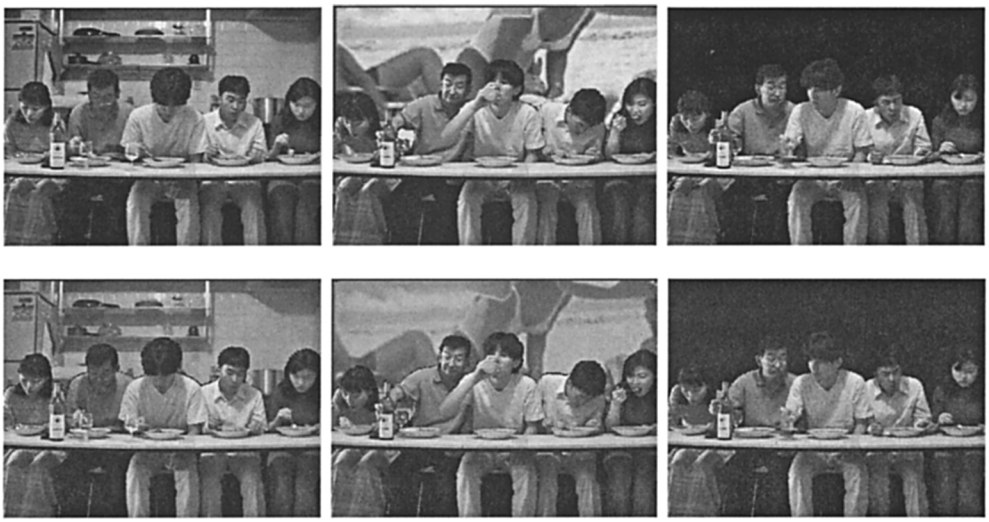

図 9　家族ゲーム・ゲーム（下段がプロトタイプ出力）

の映像が変化するという関係によって，主題と応答の関係が実現される．このように，“家族ゲーム・ゲーム”における模倣関係は極めて抽象性が高く，その構造が Meta Script によって直接的に記述されている訳ではない．

その一方で，“家族ゲーム・ゲーム”の映像は，“Dance Canonica”や“invention”に比べて，より通常の映画やテレビドラマのシーンに近い性格を持っている．もともと扱った素材が一種のホームコメディであったため，それに適した合成手法としてクロマキー合成による背景映像の書き割り的な合成を行った．結果的には，構成手法として対位法を用いながらも，合成された映像が与える印象は映像素材の持つドラマ的な傾向の範囲内に収まっている．このように，“家族ゲーム・ゲーム”は，一般のドラマ的な映像に対位法を適用する方法の一例でもある．

クロマキー合成の結果は，Premiere でもプロトタイプの CHROMAKEY モードでもほぼ同一であった．但し，“ Dance Canonica ”と同様な画像圧縮の問題により，画質的にはプロトタイプの鮮明度が劣っている．また，具体的な素材構成の構造は“ invention ”よりもはるかにシンプルであり，Meta Script による記述は階層的にも同期関係的にも分かり易いものである．

4.2 プロトタイプシステムの操作性

前節では，Meta Script の記述能力とプロトタイプの映像合成能力を中心に考察した．映像合成上の全般的な問題として，現状の Movie Renderer では画質や映像素材の時間変換の精度が不充分な点が挙げられるが，これに関しては QuickTime 版では改善されている．プロトタイプの操作性に関しては，それを Premiere に代表されるノンリニア映像編集システムと比較した時，次の事が言える．

映像編集の初期段階において, 様々な映像合成の効果を試行錯誤するためには, Premiere の GUI の方が操作性が良い．この一つの理由は，現在のプロトタイプがバッチ処理的で Movie Graph GUI から直接レンダリング結果を確認できないためである．もう一つの理由は，Movie Graph GUI においては Premiere のようなタイムラインによる直接的な時間関係の確認ができないためである．

一方，対位法に代表される上位構造の定義，編集や，それに対する映像要素の割付の操作に関しては，プロトタイプの方が効率的である．従って，Meta Script と Movie Graph GUI に基づいた本プロトタイプの機能とノンリニア映像編集システムの機能とを統合化したシステムを構築できれば，それが理想的である．

4.3 Meta Script と他のスプリプト言語との比較

以下に，映像やマルチメディアの分野における既存の代表的なスクリプト言語を取り上げ，それらと Meta Script とを比較する．

4.3.1 EDL

EDL（Edit Decision List）[19]は，映像系の編集制御記述形式として最も基本的なものであり，タイムコードベースで逐次的に映像編集機器の制御情報を記述する．Meta Script よりも Raw Script に相当するレベルのものと言える．“ 家族ゲーム・ゲーム ”の映像合成実験では，Meta Script から EDL レベルの言語に変換して，ビデオテープベースのリアルタイム映像合成を行う実験も行った．Premiere は EDL 形式の出力機能を持つ．Premiere で EDL の入力も可能ならば，EDL を Premiere とのインタフェースに利用する可能性が考えられるが，残念ながら Premiere がサポートしているのは出力のみである．何れにしても，ビデオテープ装置の制御言語とファイル合成の記述とでは明らかに要求される仕様が異なると考えられる．

4.3.2 VRML

VRML（Virtual Reality Modelling Language）[20]は，SGI 社の Open Inventor[21]のファイルフォーマットに基づいて考案された Web 用の 3 D 空間の記述言語であり，現在は ISO 標準となっている．Meta Script，Movie Graph，Movie Graph GUI のアイデアは，VRML や Open Inventor を参考にして考案したものであり，データ構造的にも類似点がある．従って，Meta Script の仕様も，表記方法としての違いはあっても，内容的にはそれらとの類似点が多い．

但し，特に時空間構造の扱いには大きな違いがある．VRML では 3 D グラフィックスがベースとなり，その構成要素が時間軸に対応付けられる．しかし，このような時空間構造は，それ自体が二次元である映像メディアを用いた多重構造的イメージ表現には整合しない．そこで，Meta Script では，二次元のイメージ空間と時間軸との組み合わせとして時空間を構成するように考えた．この点に関しては，Meta Script の構造は，むしろノンリニア映像編集ソフトウェアや Apple 社の QuickTime[18]における構成に近い．但し，これらのソフトウェア自体はスクリプト言語を持たず，そこにも Meta Script の存在意義がある．

4.3.3 SMIL

SMIL（Synchronized Multimedia Integration Language）[22]は，W 3 C が仕様策定した XML 準拠の Web 用マルチメディア記述言語であり，動画像の同期再生を記述する事が出来る．今日存在するスクリプト言語の中では最も Meta Script に近いものと言えるが，Meta Script の映像合成と SMIL の映像再生とは明らかに目的が異なり，その違いに準じた仕様の違いが存在する．即ち，SMIL はマルチウィンドウ的な映像の並列再生を想定しているため，空間配置情報の記述機能を持つが，その一方で，映像合成に関する記述機能は持たない．

順列/並列の組み合わせに基づいた映像の構造の記述に関しては，具体的な表記方法を別とすれば，Meta Script と全く同じ考え方である．SMIL では，さらにこれとイベントの記述を組み合わせる事ができる．一方，Meta Script には，通常の意味でのイベントの概念はなく，構成上の同期関係に特化した記述機能として Define Node の offset 属性が用意されている．また，Define Node に類似したものとしては SMIL の head element があるが，これは所謂ヘッダー部であり，構造的に任意の階層に置ける Define Node 方が自由度が高い．さらに，Meta Script には Symbol Node の利用による抽象度の高い記述能力がある．SMIL の meta element は Symbol Node との類似性を持つが，head element にのみ含まれるものであり，また，Meta Script のようなデータベースとの関連付けはない．

以上，ごく代表的なもののみを取り上げた．この他に注目されるものとして，TVML，LINGO，MPEG 4 等が挙げられるが，それぞれ次のようにその位置付けや目的が Meta Script とは異なっている．TVML は，NHK 技研が開発した映像番組制作用スクリプト言語である．CG による番組を自動生成する事を目的とする言語であるため，Meta Script のような既存映像素材の編集のための言語とは記述内容もシステムに要求される機能も異なっている．LINGO は，Macromedia 社のマルチメディアオーサリングシステムである Director 用の専用スクリプト言語である．LINGO はマルチメディア素材の木目細かい制御を可能とするが，Meta Script のような構造記述言語ではなく，より一般的なプログラミング言語として位置付けられる．MPEG 4[23]は，ISO の動画像圧縮の国際標準規格であり，これらのスクリプト言語とは全く異なるが，その仕様が要求する機能は，映像合成を含めて Meta Script の機能のほとんどをカバーしている．MPEG 4 に特化した標準スクリプト言語が存在する訳ではないが，MPEG 4 のシーン記述には VRML の仕様を 2 D 拡張したバイナリーフォーマット（BIFS）が採用されている．2 D 拡張されている点においては，VRML から Meta Script

により近づいていると言える．だが，構造の記述そのものよりは再生のリアルタイム性を重視したものであるため，やはり時空間構造の考え方はMeta Scriptとは異なり，VRMLのScene Graphを映像の時系列変化に合わせて更新して行くという方式が採用されている．

SMILに代表される時間制御の構造形式は，今日のマルチメディア・フレームワークの考え方として一般化しつつあり[24][25]，それはMeta Scriptのそれとも基本的に同等のものである．その一方で，映像合成という分野においては，Meta Scriptに相当するスクリプト言語はまだ存在しない．対位法的映像とは，決して特殊な映像ではなく，通常の映像形式も含めて，より汎用性の高いものである．Meta Scriptを使用する事により，今後の一層のマルチメディアの普及を踏まえた高度な映像記述が可能となる．

5. Image Wave[26]～[29]

Meta Scriptに基づいて対位法的映像を構成する上での残された課題として，対位法の和声ルールに相当するような映像要素間の整合関係をいかに成立させるかという問題がある．そのような整合関係の構成要素としては，色調や構図等の様々な構成要素が考えられるが，今までの映像合成実験において，整合関係を最も左右する要因となっているのは，映像要素間の時間的同期関係であった．特に興味深いのは，互いに全く独立した別個の映像や音声を組み合わせてみると，適当な線形変換により調整する事により，それらがぴたりと同期しているように感じられるという現象が何度も観察された事である．そこで，この同期の問題に対象を絞り込んだ研究“Image Wave”を99年度より新たに開始した．

Image Waveとは，以下の仮説及びそれに基づいた研究コンセプトを指す総称である．

1) 映像のように時間的に存在するイメージは，内在するリズムを持っている．そのリズムは，それぞれが位相，周波数，振幅，ゆらぎを持った波の合成波として形作られている．
2) 人間の心の中のイメージも合成波である．

物理的に存在する映像や音とその脳内イメージとを波動という観点から統一的に捉える事により，それらの同期関係を効果的に制御する事が可能になるだろう．また，そのようなシステムを実際に実現できれば，イメージというものに対するそのような捉え方が適切である事が間接的に証明される事になる．

Image Waveでは，映像の時系列変化の情報を波動情報として捉え，その時間周波数解析や時系列特徴点抽出によって得られた情報を元に，複数の映像間の同期合成を行う．解析の対象となる時系列情報としては，フレーム間の輝度分布や空間周波数の変位等の統計量を用いる．一般に，映像には，固定的なリズムを持った映像と，より自由なリズムを持った映像とがあると考えられる．前者の例は，曲に合わせて踊っている映像や同一の動きをループ再生した映像であり，後者の例は，それ以外のより一般的な映像である．例えば，ダンスの映像でも，ポップス系と創作ダンスとでは異なる．ポップス系のダンス映像からは直接的に周波数を求め易いが，創作ダンスの映像

から周波数を求める事は困難であり，むしろフレージングの特徴点的な情報を元に，そのリズムを把握する事になる．このようなリズムの特質の違いに基づいて，同期合成の技法も，固定リズムの位相と周波数を一致させる同期合成と，時系列特徴点を一致させる同期合成とに分類される．Image Wave では，両者（及びその組み合わせ）を試みるが，中でも特に後者を重視し，これを対位法的同期（Counterpoint Synchronization）と呼ぶ．

現在は，この考え方に基づいて，映像の解析と同期合成プログラムのプロトタイピングを進めているところである．実用的なレベルでそれを実現する事やその有効性を実証する事には困難な面も多いが，時間的同期の問題は AI や脳研究においても特に最近になって注目されてきた分野であり，認知科学的アプローチによりマルチメディア処理技術の新たな可能性を切り開く事を目指したいと考えている．

6. おわりに

映像表現を対象とするマルチメディア処理技術の研究について紹介した．筆者はもともと建築設計の出身であり，その長らく CAD システムの研究開発に従事していた．映像系の研究に関わったのは，ごく最近の事である．建築設計とシステム設計と映像編集とでは，まるで違うという印象を持つ人も多いかも知れない．しかし，“ composition ”という観点から俯瞰すると，これらはデザイン的な共通点が非常に多く，私はそのような観点から映像の問題にアプローチしていった．

映像に代表されるマルチメディアの処理が建築や CAD と大きく異なるのは，時間情報を直接的に制御するという点である．これは極めて興味深い問題であり，この研究を始めてからあらためて「時間」というものに関心を持った．時間を空間と同等に扱う事が可能か，そのように扱うべきか，あるいは時間固有の問題としてどのような問題があるのか，それらを踏まえた上で，対象を適切に処理するためには時空間をどのように構成すれば良いのか，というのは尽きない問題である．今回の研究を機会に，今後も時間をかけて考えて行きたい．

謝辞　対位法的映像の事例作品の制作に当たって御協力頂いたダンスグループ Dance Rhizome，神戸大学発達科学部柴研究室，同志社大学演劇サークル有志に深謝する．

参考文献　［1］　鈴木良太郎, 井上誠喜, スクリプトを用いたマルチメディア・モンタージュ, 情報処理学会第 55 回全国大会, 1997
［2］　鈴木良太郎，井上誠喜，スクリプトを用いたマルチメディア・モンタージュ―その 2. 対位法的映像の考察―, 情報処理学会第 56 回全国大会, 1998
［3］　鈴木良太郎，井上誠喜，スクリプトを用いたマルチメディア・モンタージュ―その 3. Meta Script の仕様―, 情報処理学会第 57 回全国大会, 1998
［4］　鈴木良太郎，岩舘祐一，スクリプトを用いたマルチメディア・モンタージュ―その 4. 対位法テンプレートによる映像合成―, 情報処理学会第 58 回全国大会, 1999
［5］　鈴木良太郎, 井上誠喜, Dance Canonica, 日本映像学会第 24 回大会映像作品展, 1998
［6］　鈴木良太郎，井上誠喜，家族ゲーム・ゲーム，日本映像学会第 24 回大会映像作品展, 1998
［7］　鈴木良太郎, INVENTION−Happy＋Sad−, 日本映像学会第 24 回大会映像作品展, 1999

［8］ Suzuki, R., Inoue, S., Dance Canonica, The 6th ACM International Multimedia Conference—Art Demos—Technical Demos—Poster Papers—, 1998, pp.37
［9］ 鈴木良太郎, 岩舘祐一, Multimedia Montage—対位法に基づいた映像合成の研究—, 映像情報メディア学会技術報告, Vol. 23, No. 33, 1999, pp. 31〜36
[10] Suzuki, R., Iwadate, Y., Multimedia Montage—Counterpoint Synthesis of Movies—, 1999 IEEE International Conference on Multimedia Computing & Systems, 1999, pp. 433-438
[11] Suzuki, R., Iwadate, Y., INVENTION—A Study of Counterpoint Structure Description—, 1999 IEEE Symposium on Visual Languages, 1999
[12] Eisenstein, S. M., エイゼンシュテイン全集 6, キネマ旬報社, 東京, 1980, p. 318
[13] Eisenstein, S. M., エイゼンシュテイン全集 7, キネマ旬報社, 東京, 1981, p. 378
[14] 池内友次郎（編), 新音楽辞典楽語, 音楽之友社, 東京, 1977
[15] 長谷川良夫, 対位法及びカノン・フーガ, 音楽之友社, 東京, 1955
[16] 吉崎清富, 対位法の泉, 音楽之友社, 東京, 1989
[17] ツェルニー編, バッハ　インベンション, 全音楽譜出版社, 東京, 1955
[18] Apple Computer Inc., QuickTime 3 Reference, Apple Computer Inc., 1998
[19] SMPTE, SMPTE 258 M, Television-Transfer of Edit Decision Lists, SMPTE, 1993
[20] ISO/SC 24, ISO/IEC 14772-1, The Virtual Reality Modeling Language, ISO, 1998
[21] Wernecke, J., The Inventor Mentor, Addison-Wesley, New York, 1994
[22] W 3 C, Synchronized Multimedia Integration Language（SMIL）1.0 Specification, W 3 C, 1998
[23] ISO/SC 29, ISO/IEC JTC 1/SC 29/WG 11 N 2725, Overview of the MPEG-4 Standard, ISO, 1999
[24] Gibbs, S., Composite Multimedia and Active Objects, OOPSLA'91, 1991, pp. 97-112
[25] ISO/SC 24, ISO/IEC 14478, Presentation Environments for Multimedia Objects (PREMO), ISO, 1998
[26] Csaky, M., 鈴木良太郎, Spatial Frequency Analysis of Movies, 情報処理学会第 58 回全国大会, 1999
[27] 鈴木良太郎, 岩舘祐一, Image Wave, 1999 年度人工知能学会全国大会, 1999
[28] 鈴木良太郎, 岩舘祐一, Image Wave—イメージの同期に関する研究—, 情報処理学会第 59 回全国大会, 1999
[29] Suzuki, R., Iwadate, Y., Image Wave—A study on image synchronization—, The 7 th ACM International Multimedia Conference—Poster Papers—, 1999

執筆者紹介　鈴　木　　良 太 郎（Ryotaro Suzuki）

昭和 29 年生．55 年東京大学大学院工学系研究科修士課程修了．平成 4 年日本ユニシス(株)入社．製造部門にて CAD/CAM システムの研究開発に従事．平成 9 年度より(株)ATR 知能映像通信研究所に出向．以後，マルチメディア情報処理の研究に従事．情報処理学会，人工知能学会，映像情報メディア学会，日本映像学会，IEEE，ACM 会員．ISO/SC 24 専門委員会委員．