JAERI-Tech
95-014

動画像処理システムの開発

1995年3月

加藤克海*・渡辺　正・町田昌彦**・蕪木英雄

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所技術情報部情報資料課（〒319－11 茨城県那珂郡東海村）あて，お申し越しください。なお，このほかに財団法人原子力弘済会資料センター（〒319－11 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Information Division, Department of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.

編集兼発行　日本原子力研究所
印　　刷　日立高速印刷株式会社

JAERI-Tech 95-014

動画像処理システムの開発

日本原子力研究所東海研究所情報システムセンター
加藤　克海*・渡辺　　正・町田　昌彦**・蕪木　英雄

(1995年2月3日受理)

情報システムセンターでは，分散処理環境の整備作業の一つとして画像処理環境の整備を進めており，画像処理サーバ用ワークステーションを導入している。このサーバ用ワークステーション上でシミュレーション結果を連続的に描画し，それを自動的にビデオ録画する動画像処理システムの開発を行なった。本報告書では，システム構成，可視化ツールによるワークステーション上への描画，ビデオ及びワークステーションの制御方法，アニメーションビデオの作成について，使用例をもとに解説する。

東海研究所：〒319－11　茨城県那珂郡東海村白方白根2－4
* 原子力データセンター
** 外来研究員富士通

Development of Animation Processing System

Katsumi KATO*, Tadashi WATANABE
Masahiko MACHIDA** and Hideo KABURAKI

Computing and Information Systems Center
Tokai Research Establishment
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken

(Received February 3, 1995)

Computer and network environment for image processing has been developed and maintained by the computing and information systems center under the course of establishing a disributed processing environment. A server workstation was introduced for image processing. An animation processing system, in which simulation results are consecutively visualized on the server workstation and automatically recorded on a video tape, has been developed. In this report, a system construction, visualization in the workstation using a visualization tool, control of video tape recorder and workstation, and production of animation video are described with some examples.

Keywords : Simulation, Visualization, AVS, Animation, Video

*NEDAC

**On leave from FUJITSU, Ltd

目　　次

Contents

1. はじめに

従来，大型計算機によるシミュレーション結果からは，大型計算機上の作図用カルコンプ・ライブラリ機能により作成されるペンプロット図のようなイメージ作画しか得られなかった．そのため，例えば等高線図の作画においては，プロットする線種に制約があったり，単色あるいは数種類の配色しか使用できなかったり，また動的表示（アニメーション化）がほとんど不可能であるなど，作画表示によるシミュレーション結果の把握が困難であった．スーパーコンピュータの出現によりシミュレーション結果は膨大なものとなってきており，正確かつ効率的な結果の理解のため，可視化技術はシミュレーションに基づく研究に必要不可欠なものとなっている．

近年，ネットワーク技術の高度化とワークステーションやパーソナルコンピュータの高性能化により，このようなシミュレーション結果の作画表示は，高度の作画機能を持つワークステーション上において可視化ツールやユーティリティを用いて行なうということが普及してきた．情報システムセンターでは，従来大型計算機で行なっていた画像処理やファイル編集などを，ワークステーションなどの小型の計算機へ移行させ，作業の効率化を図る分散処理環境の整備を進めており，シミュレーション結果の可視化のため，画像処理サーバ，カラープリンタサーバなどの整備を行なっている [1]．この画像処理環境の整備作業の一つとして，汎用可視化ツールＡＶＳが画像処理サーバに導入されている [2][3]．ＡＶＳは，シミュレーション結果や実験結果などの数値データを，直接プログラミングすること無く容易に可視化することができるソフトウェアであり，広く利用されている．

可視化の結果は，ワークステーションの画面で直接見たり，あるいは論文等への使用のためにプリントされるばかりではなく，過渡現象を表す時系列データの動的な理解や効果的なプレゼンテーションのためにも利用されている．このため，時系列データの連続的表示とその記録，いわゆるアニメーション作成などの動画像処理技術も，シミュレーションに欠くことのできない重要な技術となっている．

本報告書では，シミュレーション結果を画像処理サーバ用ワークステーション上でＡＶＳを用いて連続的に描画し，それを自動的にビデオ録画する動画像処理システムの開発，および使用方法について解説する．システムの構成，ＡＶＳによるワークステーション上への描画，ビデオテープレコーダー（ＶＴＲ）及びワークステーションの制御方法，アニメーションビデオの作成について，使用例をもとに説明する．

2. システム構成

アニメーションビデオの作成を行なった動画像処理システムの構成をFig.2.1に示す.

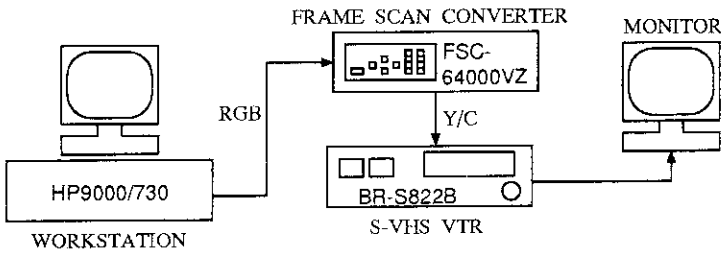

Fig. 2.1　システム構成図

Fig.2.1に示されているように，画像処理サーバ用として導入整備されているワークステーションはＨＰ９０００／７３０ ＣＲＸ－２４ｚ（ＯＳ：ＨＰ－ＵＸ，主メモリ：１２８ＭＢ，ディスク容量：３．７ＧＢ）である[1]．可視化ツールＡＶＳは，このワークステーション上で作動する．

フレームスキャンコンバータは，ワークステーションからの映像用ＲＧＢ信号を標準のＴＶ信号に変換するもので，ディスプレイ上の録画したい部分（ウィンドウ）の選択や，画面の拡大，縮小などの調整もここで行なう．これは，（株）フォトロン製，ＦＳＣ－６４０００ｖｚである．

ＶＴＲは，Victor製ＢＲ－Ｓ８２２Ｂである．モニターは，おなじくVictor製カラービデオモニターＶＭ－Ｒ１５０Ｓである．

このシステムにおいて，ワークステーション上でＡＶＳを用いてシミュレーション結果を描画し，同時にビデオ制御（サーバ）プログラム，可視化制御（クライアント）プログラムを実行することによって，連続的な描画をアニメーションとしてＶＴＲに録画する．このとき，ワークステーションの画面上には，シミュレーション結果描画用のウィンドウ，ＡＶＳ制御用のウインドウ，ビデオ制御用のウインドウの３つが表示される．モニター上には，フレームスキャンコンバーターを通して実際にビデオに録画されている画面が表示される．

3. ワークステーション上での描画

ＡＶＳを用いたワークステーション上での描画についての概略を記述する．また，アニメーション作成例として，２次元コンタ図，３次元コンタ図，ベクトル場，トレーサー粒子表示を取り上げ，それぞれの単一画面の描画について説明する．すなわち，ここで示した単一画面は，連続したアニメーションのある時間でのスナップショットである．

3.1 可視化ツールＡＶＳ

可視化ツールＡＶＳ（Application Visualization System）は米国ＡＶＳ（Advanced Visual Systems）社が開発したコンピュータグラフィックス用ソフトウェアである．ＡＶＳはシミュレーション結果や実験データなどの数値データを，直接プログラミングすることなく容易に可視化することができる．ここでは，モジュールと呼ばれる処理単位を組み合わせて必要な画像処理の一連の作業を構成し，そのモジュールの組合せ（ネットワークと呼ばれる）を順次実行することによって，データの読み込み，処理，軸やラベルの設定，色指定，画像ファイルの作成などを行なう．また，連続的な描画や自動的な画像処理などを行なうために，ＡＶＳの実行をＣＬＩ（Command Language Interpreter）と呼ばれる機能により制御することができる．

3.1.1 モジュール

モジュールは従来の作画プログラムではサブルーチン，あるいは関数に相当するもので，機能により以下の４種類に分けられる．

- データ入力モジュール
  各種データファイルの読み込み，またはデータ作成．
- フィルタモジュール
  データを変換し，別のデータを作成，ある値の抽出．
- マッパーモジュール
  数値から表示可能な形式に変換．
- データ出力モジュール
  画面に処理結果を表示，ファイルに出力．

この４種類の機能のモジュール群から，必要なモジュールをとりだし，組み合わせてネットワークを作成する．ただし，データ出力モジュールのうち，イメージビューワ，グラフビューワ，ジオメトリビューワと呼ばれる３つのモジュールは，ネットワークに組み込まず，単体で使用することも可能である．

ＡＶＳに用意されているモジュールに適当なものがない場合はユーザがモジュールを作成することが可能である．この場合，モジュールとして受け持たせる機能を定義する原始プログラム（モジュールプログラム）をまず作成し，これをコンパイルすることにより使用可能なモジュールを作成する．

ユーザが独自に作成するモジュールのプログラムは，vi等の通常のエディタ上で作成する．例えば，後述の2次元コンタを描画する例では，データ入力モジュールとして，read_con253253という名称のモジュールを使用しているが，まずread_con253253_f.fというモジュールプログラムを作成し，次に，このプログラムをコンパイルして，read_con253253_fを作成する．このモジュールは，ＡＶＳ上では，read_con253253という名称となるようにプログラム中で定義してある．モジュールプログラムはFORTRANで記述しても，Ｃ言語を用いても良い．モジュールの作成の詳細については，参考文献[3][4]を参照のこと．

3.1.2 ネットワーク

ネットワークは，従来の作画プログラムでは主プログラムに相当するもので，処理単位であるモジュールを組み合わせたものとなっている．ネットワークは，ネットワークエディタと呼ばれるエディタ上で，入力，データ変換，出力などの各種モジュールを組合せ，実行順序をモジュール間の接続で指定することにより構築する．このネットワークを実行させることによりデータの可視化，ファイルの作成などを行なう．ネットワークの修正，保存等もネットワークエディタ上で，マウス操作により容易に行なうことができる．ネットワークの作成，修正，保存，実行についての詳細は参考文献[3] [4]参照のこと．

3.1.3 ＣＬＩ

ＡＶＳにはＣＬＩ（Command Language Interpreter）と呼ばれる機能がある．これは，ＡＶＳを制御するためのコマンドスクリプトの実行機能である．コマンドスクリプトはＣＬＩ言語と呼ばれるＡＳＣＩＩ言語で記述される．ＣＬＩ言語によって書かれた命令の集まりであるスクリプトを実行することにより，自動的に画像処理の作業を行なうことができる．このスクリプトは文法に従って作成するか，ＡＶＳの起動時にオプションをつけることによって一連の操作を記録し，それをスクリプトとすることも可能である．本報告書で取り上げたアニメーション化の例では後者の方法によって作成し，これにより可視化結果を自動的にＶＴＲに収録した．

ＡＶＳにはデモンストレーション機能も付いており，様々な機能を見ることができ，ネットワークを作成する際のヒントとすることも可能である．このデモンストレーション機能はＣＬＩ言語で作成されている．

3.2　2次元コンタの描画

3.2.1　ネットワーク

2次元コンタ図の描画に使用したネットワークをFig.3.1に示す.

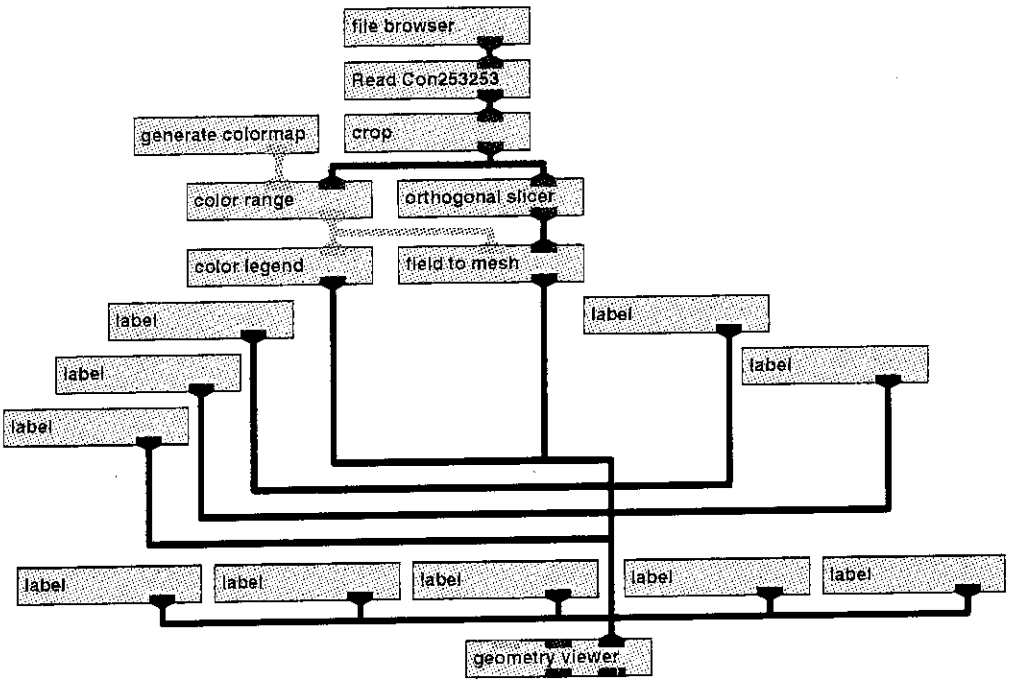

Fig. 3.1　2次元コンタ図用ネットワーク

Fig.3.1のネットワークで使用しているモジュールの機能の概略は，以下の通りである.

- file browser （ファイルを選択する）

  描画するファイルを選択する.

- Read Con253253 （描画データを読み込む）

  選択されたファイルから描画データを読み込む．このモジュールは2次元描画用に独自に作成した.

- crop （任意の範囲のデータを抽出する）

  読み込んだデータの内，描画に用いるデータを抽出する．通常は読み込んだデータを全て描画するが，このモジュールで範囲を指定することにより，ある一部分のみを描画することも可能である.

- orthogonal slicer （座標軸に垂直な平面でボリュームデータを切断する）

  描画データは253 × 253の2次元であるが，便宜上Read Con253253モジュールで253 × 253 × 3の3次元化している．この3次元の配列から2次元の断面を取り出している.

- field to mesh （2次元のスカラーフィールドを3次元の面に変換する．）

  2次元のデータを geometry viewer モジュールで描画可能な3次元の面に変換する．

- generate colormap （ＡＶＳカラーマップを出力する）

  カラー表示のためのカラーマップデータ構造を生成し出力する．

- color range （カラーマップの最大値，最小値を合わせる）

  読み込んだデータの最大値，最小値を見つけ，カラーマップの最大値，最小値とする．

- color legend （描画ウィンドウ内に色とデータの対応関係を表示する）

  描画ウィンドウ内にカラーマップとデータ値を表示する．これにより，色と数値の対応が一目でわかる．

- label （描画ウィンドウに注釈を表示する）

  モジュールで入力した文字列，数値を描画ウィンドウ内に注釈として指定した場所に表示する

- geometry viewer （データを表示する）

  上段モジュールからのデータをもとにディスプレイ上に描画を行なう．

3.2.2　2次元コンタ図

Fig.3.1に示されている2次元コンタ図用ネットワークを使用した描画例をFig.3.2に示す．これは，2次元超伝導体のシミュレーションにより得られたもので，超伝導状態中に発生した渦電流の挙動を示したものである．

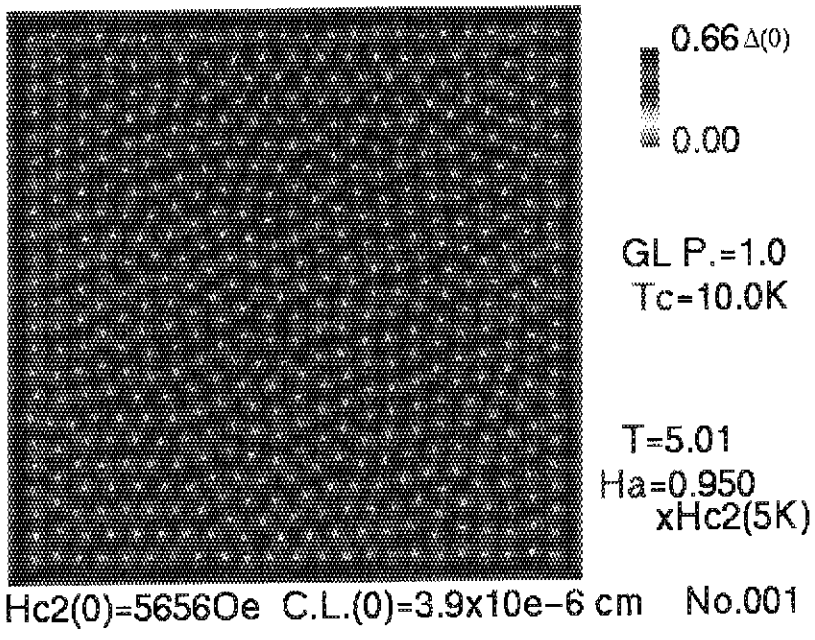

Fig. 3.2　2次元コンタ図描画例

3.3　3次元コンタの描画

3.3.1　ネットワーク

3次元コンタ図の描画に使用したネットワークをFig.3.3に示す.

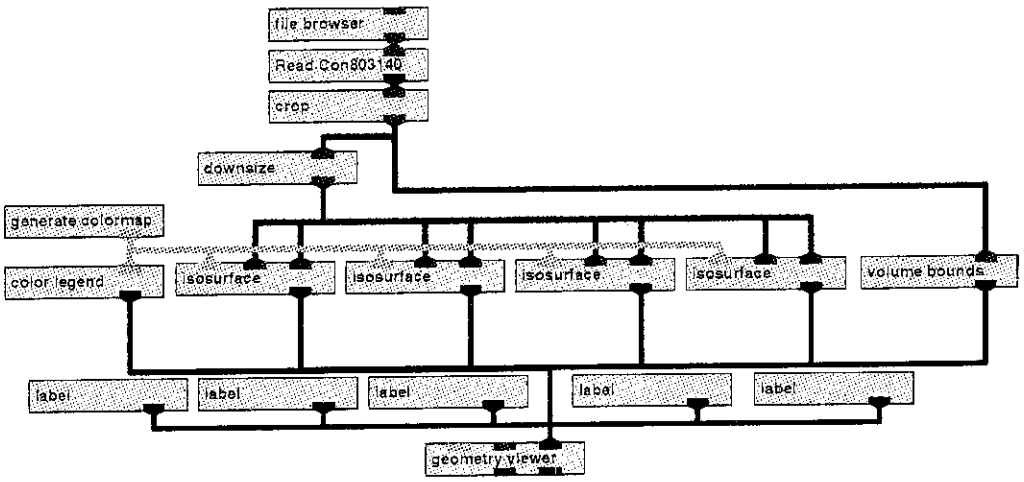

Fig. 3.3　3次元コンタ図用ネットワーク

Fig.3.3のネットワークで使用しているモジュールの機能の概略は，以下の通りである.

- file browser （ファイルを選択する）

  描画するファイルを選択する.

- Read Con804130 （描画データを読み込む）

  選択されたファイルから描画データを読み込む．このモジュールは3次元描画用に独自に作成した.

- crop （任意の範囲のデータを抽出する）

  読み込んだデータの内，描画に用いるデータを抽出する．通常は読み込んだデータを全て描画するが，このモジュールで範囲を指定することにより，ある一部分のみを描画することも可能である.

- downsize （サンプリングによりデータのサイズを小さくする）

  入力されたデータをサンプリングすることによってサイズを小さくする．パラメータで指定した数値分を飛ばしてデータを下段モジュールに出力する．小さくしたデータによる描画は，スピードは速いが，解像度は落ちる.

- volume bounds （3次元のフィールドを作成）

  描画するフィールドを作成し，3次元の枠を表示する.

- isosurface （データの等値面を作成する）

  等値面を表すオブジェクトを作成する．等値面とは，２次元の等高線を３次元に拡張したもので，パラメータで指定した値を持つフィールド要素をつなぐ．

- generate colormap （ＡＶＳカラーマップを出力する）

  カラー表示のためのカラーマップデータ構造を生成し出力する．

- color legend （描画ウィンドウ内に色とデータの対応関係を表示する）

  描画ウィンドウ内にカラーマップとデータ値を表示する．これにより，色と数値の対応が一目でわかる．

- label （描画ウィンドウに注釈を表示する）

  モジュールで入力した文字列，数値を描画ウィンドウ内に注釈として指定した場所に表示する．

- geometry viewer （データを表示する）

  上段モジュールからのデータを元にディスプレイ上に描画を行なう．

3.3.2　3次元コンタ図

Fig.3.3に示される3次元コンタ図用ネットワークを使用した描画例をFig.3.4に示す．これは，構造欠陥を含む3次元の層状超伝導体のシミュレーションにより得られたもので，超伝導状態中に発生した渦電流の挙動を示したものである．

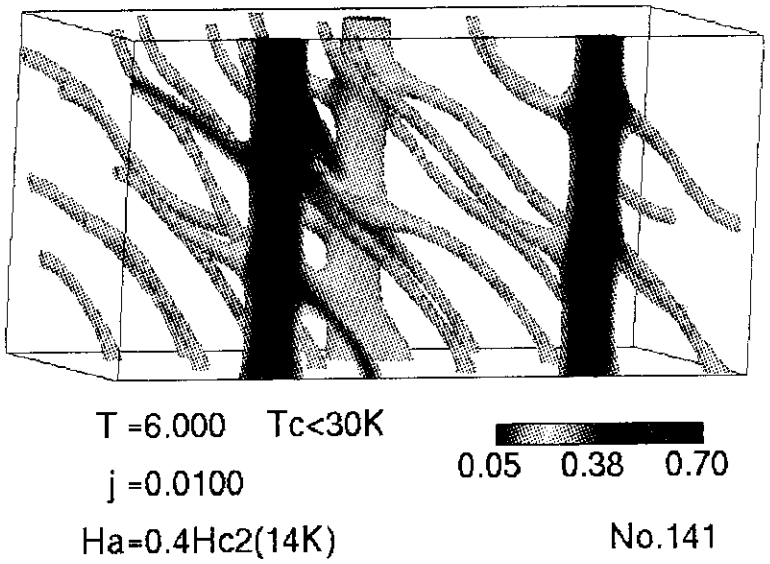

Fig. 3.4　3次元コンタ図描画例

3.4 ベクトル場の描画

3.4.1 ネットワーク

流れ場の描画に使用したネットワークを Fig.3.5に示す.

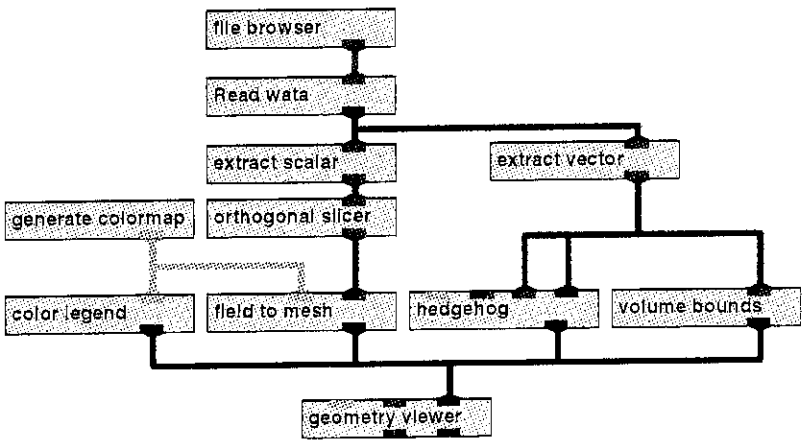

Fig. 3.5 ベクトル場用ネットワーク

Fig.3.5に示したネットワークで使用したモジュールの機能の概略を以下に示す.

- file browser （ファイルを選択する）

  描画するファイルを選択する.

- Read wata （描画データを読み込む）

  選択されたファイルから描画データを読み込む. このモジュールはベクトル場描画用に独自に作成した.

- extract scalar (入力されたデータからスカラー・フィールドをとりだす)

  上段モジュールからのデータの中から選択したスカラーデータを取り出す.

- extract vector (入力されたデータからベクトルフィールドを取り出す)

  上段モジュールからのデータの中から選択したベクトルデータを取り出す.

- orthogonal slicer （座標軸に垂直な平面でボリュームデータを切断する）

  描画データは 80×40 の 2 次元であるが, 便宜上 Read wata モジュールで $80 \times 40 \times 3$ の 3 次元化している. この 3 次元の配列から 2 次元の断面を取り出している.

- field to mesh （2 次元のスカラーフィールドを 3 次元の面に変換する. ）

  2 次元のデータを geometry viewer モジュールで描画可能な 3 次元の面に変換する.

- hedgehog （3次元3成分のベクトルフィールドにベクトルを配置する）

  ベクトルフィールドに読み込んだデータのベクトルを配置する.

- volume bounds （3次元のフィールドを作成）

  描画するフィールドを作成し，3次元の枠を表示する.

- generate colormap （ＡＶＳカラーマップを出力する）

  カラー表示のためのカラーマップデータ構造を生成し出力する.

- color legend （描画ウィンドウ内に色とデータの対応関係を表示する）

  描画ウィンドウ内にカラーマップとデータ値を表示する．これにより，色と数値の対応が一目でわかる.

- geometry viewer （データを表示する）

  上段モジュールからのデータを元にディスプレイ上に描画を行なう.

3.4.2　ベクトル場

Fig.3.5に示された流れ場用ネットワークを使用した描画例を図3.6示す．これは，2次元熱伝導／対流系のシミュレーションにより得られたもので，対流状態におけるベクトル場と対応する温度分布を示したものである．

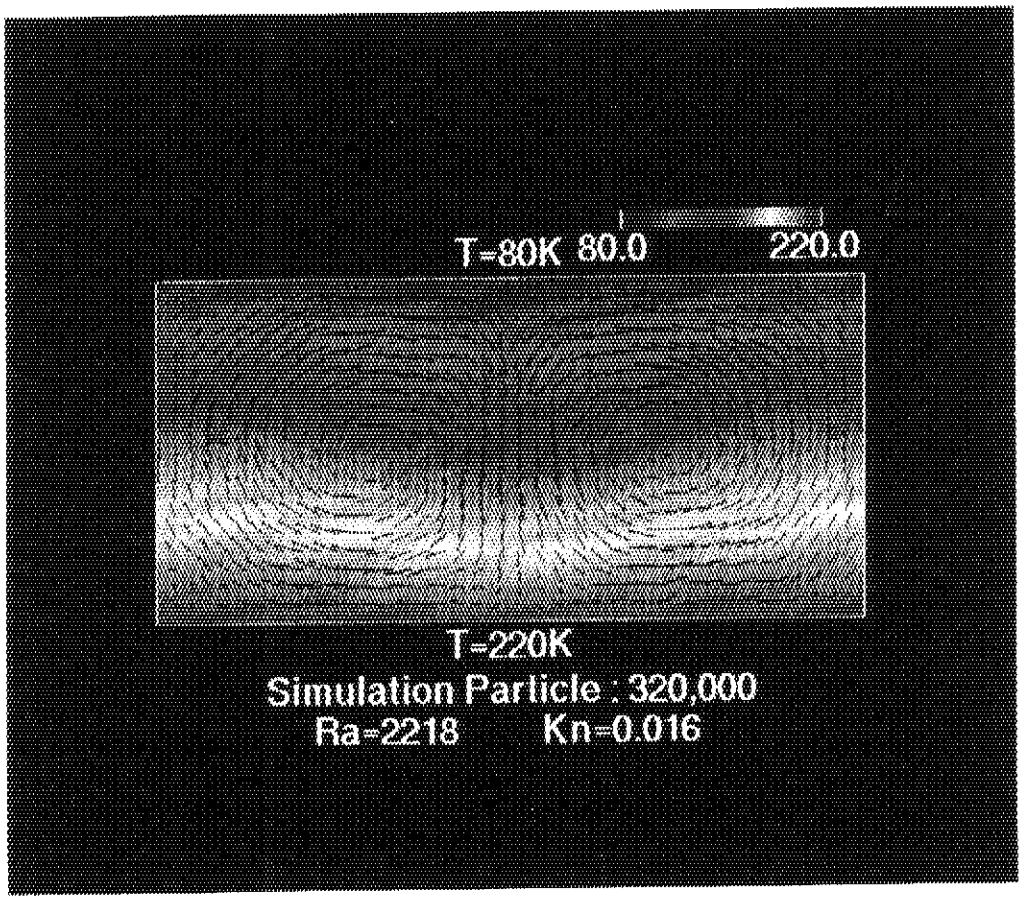

Fig. 3.6　ベクトル場描画例

3.5 トレーサー粒子の描画

3.5.1 ネットワーク

トレーサー粒子の描画に使用したネットワークを Fig.3.7に示す.

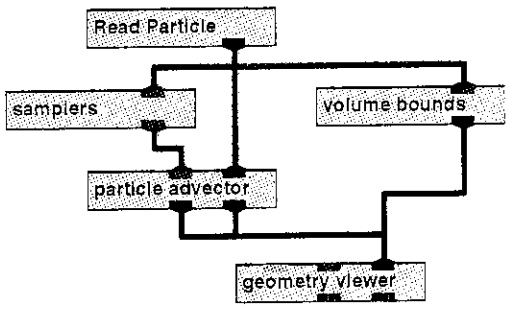

Fig. 3.7 トレーサー粒子描画用ネットワーク

Fig.3.7に示したネットワークで使用したモジュールの機能の概略を以下に示す.

- Read Particle (トレーサー粒子描画用データを読み込む)

  ファイルを選択しデータを読み込む. このモジュールも独自に作成.

- volume bounds (3次元のフィールドを作成)

  描画するフィールドを作成し, 3次元の枠を表示する.

- samplers (上流モジュールからのデータから位置を示すデータのサブセットを抽出する)

  上流のモジュールから流れてきたデータの中から位置を示すデータを抽出する.

- particle advector (パーティクルを速度フィールドにリリースする)

  上流から流れてきたベクトルデータを速度フィールドデータとして扱い, 位置データの示す位置にパーティクルを配置する. またボタン操作により, 配置したパーティクルを速度フィールドデータに従って動かすことができる.

- geometry viewer (データを表示する)

  上段モジュールからのデータを元にディスプレイ上に描画を行なう.

3.5.2　トレーサー粒子

Fig.3.7に示されたトレーサー粒子描画用ネットワークを使用した描画例を図3.8示す．これは，Fig.3.6に示された対流状態における流れ場にトレーサー粒子を流した場合の粒子の移動を示したものである．

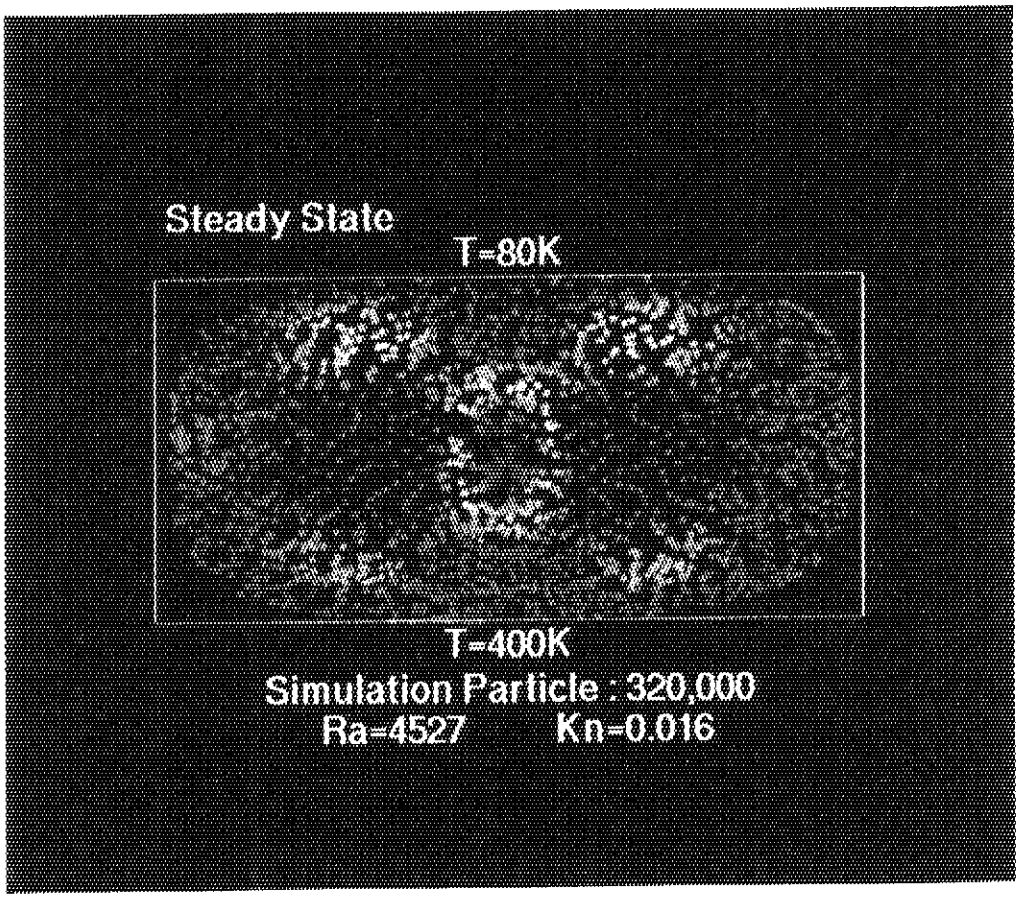

Fig. 3.8　トレーサー粒子描画例

4. アニメーション化とビデオ録画

ＡＶＳによる描画を連続して行ない，ビデオ録画することによりアニメーションビデオを作成することができる．このとき，ＡＶＳによる可視化部分とＶＴＲの録画を同時に制御する必要がある．ＡＶＳの可視化部分とＶＴＲの制御部分はプロセス間通信によって連動させており，ＶＴＲ側をサーバプログラム，可視化側をクライアントプログラムがそれぞれ受け持っている．以下では，ＡＶＳによる連続描画，ＶＴＲの制御，可視化の制御，及び実際のアニメーションビデオの作成手順について述べる．

4.1 連続描画

ＡＶＳによる可視化結果を連続してビデオテープにコマ撮りしていくとアニメーション化が実現でき，これは時系列データの解析には効果的である．ＡＶＳの操作はマウスを使って行なうが，連続して数十〜数百枚の描画を行ないコマ撮りするといった場合，１枚１枚マウスで操作することはかなりの労力を要する．そこでＡＶＳのＣＬＩ機能を用いることによりこれを自動化した．以下にＣＬＩスクリプトの作成及び実行方法を示す．

4.1.1 ＣＬＩスクリプトの作成

通常ＣＬＩスクリプトは"xxxx.scr" というファイルの形で作成する．

まずＡＶＳを起動する．通常"avs" とタイプすれば良いがＣＬＩモードで起動するため"avs␣-cli" とタイプして起動を行なう．

起動後，起動を行なったウィンドウでEnterするとプロンプトが" avs>" となる．この状態で次の命令を実行する．

```
avs>script␣-open␣xxxx.scr
```

この命令によりスクリプトファイルが定義されＡＶＳ終了までの全てのマウス操作がこのファイルにＣＬＩ言語で記録される．従って描画１枚分の操作を記録し，後にこのファイルをエディタで１００枚分の操作に編集（描画１枚分の命令を枚数分コピーする）することにより１００枚描画するスクリプトファイルを作成する．今回作成したスクリプトファイルを以下のFig.4.1に示す．

```
/**** ネットワークの読み込み ****/
net_read /home/sigcs/kato/sc/high-Tc2dvideo.net

net_flow off
parm_set label.user.11:"Title String" 001
parm_set "file browser.user.7":"File Browser"  (継続)
          /tmp/Work/kato/video/baAF00T80T80K6.VIS001
net_flow on

/**** avs_exist ファイル（描画完了）の書き込み ****/
sh touch /home/sigcs/kato/VLAN/avs_exist

/**** 1秒間スリープ ****/
script_sleep 1

/**** 次のデータの読み込み ****/
net_flow off
parm_set label.user.11:"Title String" 002
parm_set "file browser.user.7":"File Browser"  (継続)
         /tmp/Work/kato/video/baAF00T80T80K6.VIS002
net_flow on
sh touch /home/sigcs/kato/VLAN/avs_exist
script_sleep 1
                            :                    :
                            : /**** 中  略 ****/ :
                            :                    :
net_flow off
parm_set label.user.11:"Title String" 100
parm_set "file browser.user.7":"File Browser"  (継続)
         /tmp/Work/kato/video/baAF00T80T80K6.VIS100
net_flow on
sh touch /home/sigcs/kato/VLAN/avs_exist

/**** avs_finish ファイル（全描画終了）の書き込み ****/
sh touch /home/sigcs/kato/VLAN/avs_finish
```

Fig. 4.1　スクリプトファイル

4.1.2 ＣＬＩスクリプトの実行

スクリプトファイルの実行は作成と同じように"avs␣-cli"で起動し，"avs>"の状態で次の命令を実行する．

```
avs>script␣-play␣xxxx.scr
```

これを実行すると記録した時と同じ操作をＡＶＳが自動的に行なう．
またシェル上で，

```
avs␣-cli␣"script␣-play␣xxxx.scr"
```

とするとＡＶＳの起動からスクリプトの実行までを自動で行なうことができる．詳しくは，ＡＶＳのユーザーズマニュアル[4]を参照のこと．

なお，今回作成したスクリプトファイルは，ＡＶＳのマウス操作の他に外部シェルにコマンドを送る"sh"命令およびスクリプトの実行を小休止させる"script_sleep"命令を含んでいる．いずれもビデオ制御のために付加したものである．

4.2 ビデオの制御

サーバプログラムが起動されると，サーバプログラムからオペレーティングシステムに１つのソケットを要求してくる．サーバがソケットを受けとると，サーバはそのソケットに，公に知られた名前を割り当てる（今回は mysoket= とした）これによって他のプログラムは，その名前に対する通話をオペレーティングシステムに要求することができる．ソケットに名前をつけた後で，サーバはクライアントからの接続要求を受けとるために，このソケットを常にモニタする．接続要求が到着し，サーバがその要求を受けるとオペレーティングシステムは，そのクライアントとサーバをソケットで結合し，通信を行なうことができる．ここで作成したサーバプログラムを以下の Fig.4.2～ Fig.4.17に示す．サーバプログラムの作成にあたっては，Ｃ言語に関して参考文献[5]を参照した．

```
/*********************************************************/
/*  server.c (V-LAN frame-by-frame recording program)     */
/*                                    for HP9000/730     */
/*********************************************************/

#include <sys/types.h>  /* this is headed on v_lan.demo program */
#include <sys/socket.h> /*                  - - -              */
#include <sys/un.h>     /*                  - - -              */
#include <sys/fcntl.h>

#include <stdio.h>
#include <string.h>

#include "std_defs.h"
#include "vlan_defs.h"

/* this is headed on v_lan.demo program */
#define ADDRESS         "mysocket"      /* addr to connect */

extern int errno;


/*
session_setup:  set-up frame-by-frame recording session

Parameters:
sio: serial I/O port device descriptor
smpte_start: frame to start recording, in SMPTE format

Return:
Returns FALSE if setup was successful.
Returns TRUE on error.
*/
```

Fig. 4.2 サーバプログラム

```
boolean session_setup(sio, smpte_start)
int sio;
char *smpte_start;
{
vlan_error_t vlan_error;
char buf[256];
char start[256];
int i,j,k;

static char *module = "session_setup";


/* Communicate with Node 1 on V-LAN Network */
        /* VTRのノード番号   1台の時は ND1          */

vlan_error = vlan_io(sio, "ND 1\r", buf);
if (vlan_error != VLAN_OK) {
return TRUE;
}

/* Stop VTR and wait for VTR to stop */

vlan_error = vlan_io(sio, "ST\r", buf);
if (vlan_error != VLAN_OK) {
print_vlan_error(vlan_error);
exit(1);
}

wait_for_vtr_stop(sio);
```

Fig. 4.3　サーバプログラム（続き）

```
/* Track Select:  Record on Video Track only */
        /* テープのトラックの設定                    */

/* vlan_error = vlan_io(sio, "TS V\r", buf); */
vlan_error = vlan_io(sio, "TSV\r", buf);
if (vlan_error != VLAN_OK) {
return TRUE;
}

/* Set Preroll time to 3 seconds */
        /* テープの「助走」量            */

/* vlan_error = vlan_io(sio, "PR 3:00\r", buf); */
vlan_error = vlan_io(sio, "PR3:00\r", buf);
if (vlan_error != VLAN_OK) {
return TRUE;
}

/* Set Postroll time to 1 second */
        /* 録画した後行き過ぎる量        */

/* vlan_error = vlan_io(sio, "PT 1:00\r", buf); */
vlan_error = vlan_io(sio, "PT1:00\r", buf);
if (vlan_error != VLAN_OK) {
return TRUE;
}

/* Set Auto-Increment to 1 frame */
/* 録画後のインクリメントするフレーム数  */
/* 録画フレーム数と一致すること          */

vlan_error = vlan_io(sio, "AI15\r", buf);
if (vlan_error != VLAN_OK) {
return TRUE;
}
```

Fig. 4.4　サーバプログラム（続き）

```
/* Build Set Inpoint string */
        /* 録画開始タイムコードの設定 */
/* "SI(smpte_start)\r"となる  */

strcpy(start,"SI ");
strcat(start, smpte_start);
strcat(start, "\r");

/* Set Inpoint */
vlan_error = vlan_io(sio, start, buf);
if (vlan_error != VLAN_OK) {
fprintf(stderr,"Bad Starting Timecode\n");
return TRUE;
}

/* Set Duration to 1 frame */
/* 1回で録画するフレーム数 */

vlan_error = vlan_io(sio, "SD15\r", buf);
if (vlan_error != VLAN_OK) {
return TRUE;
}

return FALSE;
}



/*
record_frame:  record one frame to videotape

Parameters:
sio: serial I/O port device descriptor
```

Fig. 4.5　サーバプログラム（続き）

```
Return:
Returns FALSE if frame record was successful.
Returns TRUE on error.
*/

boolean record_frame(sio)
int sio;
{
char buf[256];
vlan_error_t vlan_error;

static char *module = "record_frame";


/* Perform Insert Edit */
vlan_error = vlan_io(sio, "PF\r", buf);
if (vlan_error != VLAN_OK) {
return TRUE;
}

sleep(1);

/* Wait until VTR is in editing mode, then fall-through... */
if (wait_for_vtr_edit(sio)) {
fprintf(stderr,"Error during wait_for_vtr_edit.\n");
return TRUE;
}

/* Wait unitl VTR is paused (editing complete)... */
if (wait_for_vtr_pause(sio)) {
fprintf(stderr,"Error during wait_for_vtr_pause.\n");
return TRUE;
}

return FALSE;
}
```

Fig. 4.6 サーバプログラム（続き）

```
/*
        <Input parameter>
        function_type : switch specifing open or close
                        (1 : open, 2 : close)

        <Return>
        the descriptor of the socket (integer)

*/

int socket_server(function_type)
int function_type ;
{
        int fromlen;
        register int i, s, ns, len, len1, len2;
        struct sockaddr_un saun, fsaun;

        /* switching (1: open, 2: close) */
        switch(function_type){

        case 1:
        /*
        * Get a socket to work with.  This socket will
        * be in the UNIX domain, and will be a
        * stream socket.
        */
        if((s = socket(AF_UNIX, SOCK_STREAM, 0)) < 0){
           perror("server: socket");
        exit(1);
        }
```

Fig. 4.7 サーバプログラム（続き）

```
        /*
        * Create the address we will be bindings to.
        */
saun.sun_family = AF_UNIX;
        strcpy(saun.sun_path, ADDRESS);

        /*
        * Try to bind the address to socket.  We
        * unlink the name first so that the bind won't fail.
        *
        * The third argument indicates the "length" of
        * the structure, not just the length of the
        * socket name.
        */
        unlink(ADDRESS);
len1 = sizeof(saun.sun_family);
len2 = strlen(saun.sun_path);

        len = (sizeof(saun.sun_family) + strlen(saun.sun_path));
/* <<<<<<<これがないとエラーが出る. >>>>>>>*/
        printf("%d\n",len);
        printf("%d\n",len1);
        printf("%d\n",len2);
printf("\n <<<< Please run the client program !! >>>> \n");

        if (bind(s, &saun, len) < 0) {
            perror("server: bind");
        exit(1);
        }
        /*
        * Listen on the socket.
        */
        if (listen(s, 5) < 0){
            perror("server: listen");
        exit(1);
        }
```

Fig. 4.8　サーバプログラム（続き）

```
        /*
        * Accept connections.  When we accept one,  ns
        * will be connected to the client.  fsaun will
        * contain the address of the client.
        */
        if ((ns = accept(s, &fsaun, &fromlen)) < 0){
            perror("server: accept");
            printf("%d\n",errno);
    exit(1);
        }

        break ;

        case 2 :
        /*
        * We can simply use close() to terminate the
        * connection, since we're done with both sides.
        */
        close(s);

        break ;
        }
        return(ns) ;
}


/*
        the funcion of sending signal :
        send_signal(descriptor_ns,signal)

        input : desctiptor_ns (int)  the descriptor of the target
                signal (int)  the sent signal (signal <= 9 * 1 byte)

        return: void
```

Fig. 4.9 サーバプログラム（続き）

```
*/
send_signal(descriptor_ns,signal)
int descriptor_ns ;
int signal ;
{
        char signal_string[10] ;

        /* transforming the frame number to a character */
        sprintf(signal_string, "%d\n", signal);

        /* sending frame number to the client. */
        if((send(descriptor_ns, signal_string, strlen(signal_string), 0))==-1)
           exit(1) ;

        return ;
}


/*
        the funcion of getting signal :
        int get_signal(file_descriptor)

        input : file_descriptor (FILE)  the descriptor of the open file

        return : signal (int)

*/
int get_signal(file_descriptor)
FILE *file_descriptor ;
{
        int response_number_s ;
        char response_string_s[10] ;

        /* receiving response from the client. */
        fgets(response_string_s,sizeof(response_string_s),file_descriptor);
```

Fig. 4.10　サーバプログラム（続き）

```
        /* transforming the character to a response number*/
        sscanf(response_string_s,"%d",&response_number_s);

        return(response_number_s) ;
}




/*
main:  main routine of V-LAN example program
*/

main()
{

int sio; /* serial I/O device descriptor */
char buf[256]; /* line buffer */
char smpte_start[256]; /* starting frame string */
vlan_error_t vlan_error; /* vlan error flag */
int number_of_frames = 0; /* number of frames to record */
int frame; /* current frame being recorded */
        int signal ;            /*              - - -           */

int type_open ;         /* defining in v_lan.demo program */
        int type_close ;        /*              - - -           */

        int descript_ns ;       /*              - - -           */

        int response_number_s ; /*              - - -           */
int repeat_number ;    /*              - - -           */

        FILE *fp;               /*              - - -           */

boolean error = FALSE; /* error flag */
```

Fig. 4.11　サーバプログラム（続き）

```
char inpoint2[256]; /* (addition) */

        type_open=1 ;
        type_close=2 ;


printf("V-LAN frame-by-frame recording server program\n\n");
        printf("*********************************************\n");
        printf("*                                           *\n");
        printf("*        1 frame = 0.5 sec (SD15)           *\n");
        printf("*                                           *\n");
        printf("*********************************************\n");

/* open serial I/O port for use */
sio = sio_open(VLAN_SIO_PORT);

/* check for error during sio_open */
if (sio <= 0) {
fprintf("Can't open %s\n",VLAN_SIO_PORT);
exit(1);
}


/* get starting frame */
printf("Begin recording at SMPTE timecode? ");
fflush(stdout);
scanf("%s",smpte_start);
fflush(stdin);

/* get number of frames to record */
printf("Number of frames to record? ");
fflush(stdout);
scanf("%d",&number_of_frames);
fflush(stdin);
```

Fig. 4.12　サーバプログラム（続き）

```
/* check number of frames */
if (number_of_frames <= 0) {
fprintf(stderr,"Bad number of frames.\n");
exit(1);
}

/* setup VTR for frame-by-frame recording */
if (session_setup(sio,smpte_start))
exit(1);

printf("\n");
printf("Please insert a pre-blacked tape with timecode into the VTR.\n");
printf("Press <return> when ready... \n");
while (getchar() != '\n');

printf("Rewinding tape.  Please wait...\n");

/* rewind tape to the beginning */
/* vlan_error = vlan_io(sio, "RW\r", buf) */;
vlan_error = vlan_io(sio, "GI\r", buf);
if (vlan_error != VLAN_OK) {
print_vlan_error(vlan_error);
exit(1);
}

/* wait until VTR is stopped (tape completely rewound) */
sleep(2);
/* wait_for_vtr_stop(sio); */
wait_for_vtr_pause(sio);

printf("\nChecking for time code on tape.  Please wait...\n");
```

Fig. 4.13　サーバプログラム（続き）

```
/* shuttle tape forward to sense timecode */
vlan_error = vlan_io(sio, "SH 8\r", buf);
if (vlan_error != VLAN_OK) {
print_vlan_error(vlan_error);
exit(1);
}

/* allow up to 30 secs. to check for valid timecode on tape */
if (! wait_for_valid_tc(sio, 30)) {
    fprintf(stderr,"Can't read time code on tape.\n");
    exit(1);
}

/* pauseing tape (addtion) */
vlan_error = vlan_io(sio, "SH 0\r", buf);
if (vlan_error != VLAN_OK) {
print_vlan_error(vlan_error);
exit(1);
}


        /* open socket */
        /* the following two steps are executed before for-loop in */
        /* v_lan_demo program                                      */

        descript_ns=socket_server(type_open);

        fp = fdopen(descript_ns, "r");

printf("\nBeginning recording session...\n");

/* for each frame... */
for (frame = 1; frame <= number_of_frames && (! error); frame++) {
```

Fig. 4.14 サーバプログラム（続き）

```
/*
   Note that the VTR is "forced" to position
   itself to the preroll point while the next
   image to be recorded is loaded.  Overlapping
   these two operations results in an improvement
   in the time required to record each frame
   to videotape.
*/

/* Position VTR to Pre-roll point. */
vlan_error = vlan_io(sio, "GP\r", buf);
if (vlan_error != VLAN_OK) {
print_vlan_error(vlan_error);
error = TRUE;
break;
}

/* printf("Starting to record No.%d frame ... \n",frame);*/
printf("\nRecording stand by for No.%d frame OK \n",frame);
fflush(stdout);

/* Load the next image to be recorded */

        /* the following three steps are executed                */
      /* at load_image(frame)                                   */
/*        printf("the sending frame number to the client : %d\n",frame);*/
        /* sending frame number to the client. */
send_signal(descript_ns,frame) ;
/*load_image(frame);*/


/* wait for VTR to pause... preroll is complete */
if (wait_for_vtr_pause(sio)) {
fprintf(stderr,"Error during wait_for_vtr_pause.\n");
error = TRUE;
break;
}
```

Fig. 4.15　サーバプログラム（続き）

```
        /* the following three steps are executed before          */
        /* the record_frame(sio)                                  */
        /* receiving response from the client. */
response_number_s=get_signal(fp) ;

        /* the judging the response from the client */
        if(response_number_s != 1)
           exit(1);
/* printf("\nRendering No.%d image is complete. \n",frame) ;*/



/* record the next frame onto videotape */
if (record_frame(sio)) {
error = TRUE;
break;
}

/* printf("\nNo.%d image has been recorded\n",frame); */

        /* transforming the recording end signal to a character */
        signal=0;
/*        printf("The sending the Recoding End Signal to the client \n\n");*/
/* sending the recoding end signal to the client. */
        send_signal(descript_ns,signal) ;
printf("\nRecording operation OK\n") ;


}

        /* close socket */
        socket_server(type_close);

if (error)
exit(1);
```

Fig. 4.16　サーバプログラム（続き）

```
printf("\nRecording complete.\n\n");

/* rewind tape to the beginning
vlan_error = vlan_io(sio, "RW\r", buf);
if (vlan_error != VLAN_OK) {
print_vlan_error(vlan_error);
exit(1);
} */

        /* Build Goto Inpoint string (addition) */
        strcpy(inpoint2,"GT ");
        strcat(inpoint2, smpte_start);
        strcat(inpoint2, "\r");

        /* Set Inpoint */
        vlan_error = vlan_io(sio, inpoint2, buf);
        if (vlan_error != VLAN_OK) {
                fprintf(stderr,"Bad Starting Timecode\n");
                return TRUE;
        }

/* wait until VTR is stopped (tape completely rewound) */
sleep(5);
/* wait_for_vtr_stop(sio); */
wait_for_vtr_pause(sio);
        printf(" \n");

/* close serial I/O line */
if (sio_close(sio)) {
fprintf("Can't close %s\n",VLAN_SIO_PORT);
exit(1);
}


     }
```

Fig. 4.17　サーバプログラム（続き）

4.3 可視化の制御

クライアントは，まずオペレーティングシステムに対して１つのソケットを要求する．次に，名前を与えられた別のソケットのいずれかにそのソケットを接続することを要求し，これによって相互通信の手順を開始する．オペレーティングシステムは，その名前のついたソケットを探し，発見すると，そのソケットをモニタしているプロセスに対して接続要求を送る．そのプロセス（サーバ）が接続要求を受けとると，オペレーティングシステムはこれらの２つのプロセスをそのソケットを介して接続する．クライアントプログラムの作成にあたっては，Ｃ言語に関して参考文献 [5] を参照した．

２つのプログラムはワークステーション上で互いに通信し合うことにより同期をとっている．クライアントはサーバからの録画するべきフレーム番号（何枚目の画像かを示す番号）を受信すると描画を開始し，完了すると描画完了をサーバに送信する．サーバはこれを受信するとビデオをリモート操作して録画を行ない，完了すると録画完了を送信する．クライアントはこれを受信すると録画が成功したとして次のフレーム番号待ちとなる．以上を繰り返して自動的にアニメーションを作成するが，このクライアントからＡＶＳを直接制御できないため，ＣＬＩスクリプトの **sh** コマンドを用いて **touch** された **avs_exist** ファイルがカレントディレクトリに存在していれば描画完了をサーバ側に送信し，サーバ側からの録画完了を受信するとファイルを消去するようにした．また **avs_finish** が存在していた場合は制御を終了（サーバ，クライアント側共）するようにした．ここで作成したクライアントプログラムを以下の Fig.4.18～ Fig.4.19に示す．

```
/*********************************************************/
/*  client.c (V-LAN frame-by-frame recording program)     */
/*                                for HP9000/730 & AVS     */
/*********************************************************/

#include <stdio.h>
#include "ipc_var.h"

#define  MAX_DRAW_TIME  1

main(){
    FILE *fp, *fp_fin ;

    /* open socket */
    descript_s_vlan=socket_client(type_open_vlan) ;
    fp_vlan = fdopen(descript_s_vlan, "r");


    while(1) {

while( (fp=fopen("avs_exist", "r" ))==NULL ) {
    sleep( MAX_DRAW_TIME ) ;
}

sleep(2);     /* 93.7/23 k.kato */

/* receiving a number of frame from server */
frame_vlan=get_signal(fp_vlan);
printf("AVS drawing completed No.= %d \n\n",frame_vlan) ;

/* printf("The frame number from the server : %d\n",frame_vlan); */


/* sending response to server */
send_signal(descript_s_vlan, response_number_c_vlan) ;
```

Fig. 4.18　クライアントプログラム

```
/* receiving the recording end signal from server */
/* printf("Sending the Rendering End Signal to the server ...\n"); */
signal_vlan=get_signal(fp_vlan);
if(signal_vlan != 0)
    exit(1);
/* printf("Recoding No.%d image is complete. \n\n",frame_vlan) ; */

fclose(fp) ;
unlink( "avs_exist" ) ;
        if( (fp_fin=fopen("avs_finish", "r"))!=NULL )
    break;

    }
       fclose( fp_fin ) ;
    unlink( "avs_finish" ) ;

    /* close socket */
    socket_client(type_close_vlan) ;

}
```

Fig. 4.19　クライアントプログラム（続き）

4.4 アニメーションビデオの作成

4.4.1 各装置の準備

4.4.1.1 ＶＴＲ

まず電源を投入し，テープの挿入を行ない巻き戻しをする．その後，以下の手順で新しいビデオテープ上にタイムコードを記録する．このタイムコードはテープ上の番地のようなもので，これが記録されていないと制御することができない．

1. ＶＴＲをストップの状態にする．
2. [REC] または [EDIT] を押す．
3. [INT/EXT] スイッチを "INT" にする．
4. [PRESET/REGEN] スイッチを "PRESET" にする．
5. [FREE/REC] スイッチを "FREE" にする．
6. [DF/NDF] スイッチを "DF" にする．
7. [VITC REC] スイッチを "ON" にする．
8. [COUNTER] スイッチを "TC/UB" にする．
9. [HOLD] ボタンを押す．

   現在のカウンターデータがホールドされ，カウンターの一番左の桁が点滅する．全ての桁を０にしたい時は [COUNTER RESET] ボタンを押す．
10. [SHIFT] ボタンを押してプリセットしたい桁へ点滅を移動させる．
11. [ADVANCE] ボタンを押して，点滅している桁の数値を設定する．
12. 10. 11. の操作を繰り返して，全ての桁に希望する数値を入力する．
13. [PRESET] ボタンを押す．

    カウンターの点滅が止まり，同時にタイムコードの歩進が始まる．タイムコードはアニメーションが全て収まるように十分に記録しておく．

4.4.1.2 ワークステーション

1. ログインを行なう．
2. 描画に必要なデータをＣＬＩで指定したディレクトリに用意する．
3. サーバ用，クライアント用，ＡＶＳのＣＬＩ実行用の３つのウィンドウを開く．

4.4.1.3　モニター

1. 電源を投入する.

2. [(Y/C)/VTR] スイッチを"VTR" にする.

   ワークステーションのディスプレイと同じ画面が出力される.

4.4.1.4　フレームスキャンコンバータ

1. 電源を投入する.

2. 録画画面を選択，調整する.

   ＡＶＳ用ウィンドウでＡＶＳを起動し１枚だけ描画させ，[SCROLL/ZOOM]，[FULL/WINDOW] ボタン，矢印ボタンを使って録画したい部分をモニタを見ながら選択，調整する.

   [SCROLL/ZOOM] ボタン

   SCROLL: 矢印ボタンがスクロール機能となる.

   ZOOM ：矢印ボタンがズーム機能となる.

   [FULL/WINDOW] ボタン

   FULL ：ディスプレイ全体を選択.

   WINDOW: ディスプレイ上の１ウィンドウ（全体の 1/4）を選択.

4.4.2　アニメーション作成操作

4.4.2.1　ＶＴＲ

1. タイムコードが記録されたテープをセットする.

2. [REMOTE] スイッチを"9pin" にする.

   これによりＶＴＲをワークステーションから制御するモードとなり，ＶＴＲのボタン類は機能しなくなる.（"LOCAL" にすると逆になる.）

3. [COUNTER] スイッチを"TC" にする.

   カウンター表示をタイムコード表示にする.

4.4.2.2　サーバウィンドウ

1. サーバプログラムを起動する.

   前回の録画を中途でやめた場合は"avs_exist"，正常終了した場合は"avs_finish" ファイルが作成されて残っているので消去しておく.

   起動

```
server
```

   録画開始点指定

```
Begin recording at SMPTE timecode? 00:01:30:00
```

録画フレーム数（枚数）指定（多めに指定しておいても支障はない）

```
Number␣of␣frames␣to␣record?␣1000
```

以上を実行するとＶＴＲは指定した録画開始点で停止し，クライアントプログラムの実行待ちとなる．

4.4.2.3　クライアントウィンドウ

1. クライアントプログラムを実行する．

   次の命令を実行する．

```
client
```

   実行するとサーバ側と通信を行ない，ＶＴＲは録画開始点より３秒前で停止し，録画待機状態となる．

4.4.2.4　ＡＶＳ・ＣＬＩウィンドウ

1. ＡＶＳを実行する．

   サーバプログラム，クライアントプログラムが実行されていることを確認したうえで次の命令を実行する．

```
avs␣-cli␣"script␣-play␣xxxx.scr"
```

   ＡＶＳのスクリプトファイルが実行され，ビデオ録画が自動的に枚数分行なわれる．

4.4.2.5　タイトル表示

タイトルや説明図は，作図ツールである”ｔｇｉｆ＋”，または LaTeX で作成した文章を表示する”ｘｄｖｉ”などを使って画面表示させ，アニメーション前に録画することによって可能である．

4.4.3　アニメーションの再生

4.4.3.1　直接ＶＴＲを操作する方法

1. 録画されたテープをＶＴＲにセットする．
2. [REMOTE] スイッチを”LOCAL”にする．
3. 録画開始点まで巻き戻す．
4. 再生ボタンを押して再生．

また，ＶＴＲの [JOG/SHTL] ボタンを押してジョグダイヤルを操作することによって，スローや倍速再生も可能である．

4.4.3.2 ワークステーションから操作する方法

1. 録画されたテープをＶＴＲにセットする.
2. [REMOTE] スイッチを"9pin" にする.
3. "vlan_talk" を起動して操作する.

   起動

   ```
   vlan_talk
   ```

   再生

   ```
   vlan:py
   ```

   録画開始点の頭出し

   ```
   vlan:gt␣00:01:30:00
   ```

   停止

   ```
   vlan:st
   ```

その他の詳しい制御コマンド一覧については, V-LAN テクニカルリファレンスマニュアル[5] を参照のこと.

5. おわりに

本報告書では，シミュレーション結果を画像処理サーバ用ワークステーション上で可視化ツールＡＶＳを用いて連続的に描画し，それを自動的にビデオ録画する動画像処理システムの開発と使用方法について解説した．本報告書が，膨大なシミュレーション結果の正確かつ効率的な理解，あるいは効果的なプレゼンテーションのための一助となることを期待する．

謝　辞

サーバ及びクライアントプログラムの作成に関して（株）フォトロンの協力を得た．ここに感謝の意を表す．

参考文献

[1] 日本原子力研究所情報システムセンター， Computer 情報 No. 58, 16–19, 平成６年５月.

[2] 日本原子力研究所情報システムセンター， Computer 情報 No. 58, 20–24, 平成６年５月.

[3] 加藤 克海，渡辺 正，蕪木 英雄，可視化ツールＡＶＳの導入整備， JAERI-memo （平成７年）．

[4] Advanced Visual Systems Inc., AVS User's Guide Release 4, 1992.

[5] David A. Curry, UNIX C プログラミング，アスキー出版局（1991）．

[6] （株）フォトロン， V-LAN テクニカルリファレンスマニュアル，１９９２年４月.

5. おわりに

本報告書では，シミュレーション結果を画像処理サーバ用ワークステーション上で可視化ツールＡＶＳを用いて連続的に描画し，それを自動的にビデオ録画する動画像処理システムの開発と使用方法について解説した．本報告書が，膨大なシミュレーション結果の正確かつ効率的な理解，あるいは効果的なプレゼンテーションのための一助となることを期待する．

謝　辞

サーバ及びクライアントプログラムの作成に関して（株）フォトロンの協力を得た．ここに感謝の意を表す．

参考文献

[1] 日本原子力研究所情報システムセンター， Computer 情報 No. 58, 16–19, 平成６年５月.

[2] 日本原子力研究所情報システムセンター， Computer 情報 No. 58, 20–24, 平成６年５月.

[3] 加藤 克海，渡辺 正，蕪木 英雄，可視化ツールＡＶＳの導入整備， JAERI-memo （平成７年）．

[4] Advanced Visual Systems Inc., AVS User's Guide Release 4, 1992.

[5] David A. Curry, UNIX C プログラミング，アスキー出版局（1991）．

[6] （株）フォトロン， V-LAN テクニカルリファレンスマニュアル，１９９２年４月.

5. おわりに

本報告書では，シミュレーション結果を画像処理サーバ用ワークステーション上で可視化ツールＡＶＳを用いて連続的に描画し，それを自動的にビデオ録画する動画像処理システムの開発と使用方法について解説した．本報告書が，膨大なシミュレーション結果の正確かつ効率的な理解，あるいは効果的なプレゼンテーションのための一助となることを期待する．

謝　辞

サーバ及びクライアントプログラムの作成に関して（株）フォトロンの協力を得た．ここに感謝の意を表す．

参考文献

[1] 日本原子力研究所情報システムセンター， Computer 情報 No. 58, 16–19, 平成６年５月.

[2] 日本原子力研究所情報システムセンター， Computer 情報 No. 58, 20–24, 平成６年５月.

[3] 加藤 克海，渡辺 正，蕪木 英雄，可視化ツールＡＶＳの導入整備， JAERI-memo （平成７年）．

[4] Advanced Visual Systems Inc., AVS User's Guide Release 4, 1992.

[5] David A. Curry, UNIX C プログラミング，アスキー出版局（1991）．

[6] （株）フォトロン， V-LAN テクニカルリファレンスマニュアル，１９９２年４月.