〔非公開〕

ＴＲ－Ｃ－００９８

３次元顔画像生成に関する研究

小　森　正　美　　大　谷　　淳
Masami　KOMORI　　Jun　OHYA

１９９４　２．２５

ＡＴＲ通信システム研究所

3次元顔画像生成に関する研究

小森正美　　大谷淳
MASAMI KOMORI　JUN OHYA

1994.2.25

ATR通信システム研究所

目次

図目次

Chapter 1

はじめに

異なる場所にいる人間同士がコミュニケーションを図る通信手段として，従来は音声による電話が主体であったが，人間同士の意思疎通には視覚も重要な働きをしていると考えられることから，現在は，テレビ電話やテレビ会議の研究開発が進められ，普及しつつある．ATR 通信システム研究所では互いに異なる複数の場所にいる複数の人々があたかも一堂に会している感覚で会議を行なうことができる通信を，臨場感通信会議と呼び，検討が進められている．臨場感通信会議の目的は，生成された仮想空間に，異なる場所にいる会議参加者の像を配置し，参加者全員が同じ空間を共有する感覚で，打ち合せや協調作業を行なえる環境を提供することにある．

臨場感通信会議システムでは，送信側の人物像の受信側における表示は，3 次元ワイヤーフレームモデル（以下，ワイヤーフレームモデルと呼ぶ）にカラーテクスチャマッピングしたものを立体表示することにより行なわれる [1]．従って，受信側で人物像の表情変化や体の動きを実時間で表現するためには，送信側で人物の時間的変化情報を実時間で検出し，これらを受信側に送り，ワイヤーフレームモデルを駆動して変形させる．この 3 次元の人物像モデルは，3 次元環境データから構成される仮想空間に配置され，受信側の立体ディスプレイに 3 次元表示される．このようなシステム構成にすることにより，従来のＴＶ会議では困難であった運動視差の再現（例えば，参加者が横に動けば，見ているものの横が見えてくる）や，会議参加者同士の視線一致の実現が可能となる．すなわち，互いに離れた場所にいる会議参加者は，あたかも一堂に介したような臨場感をもって会議を行なうことができる．

本報告は 3 次元顔モデルにおける表情生成に関するものである．人間の表情は，顔に存在する表情筋と呼ばれる筋肉の動きにより発生する．従って，3 次元顔モデルにおいて表情を生成するためには，各表情筋の時間的な 3 次元変形の情報が有効である．すなわち，Ekman らが，表情筋の場所と変形の仕方について FACS[5] と呼ばれる分類学的なマニュアルを作成しているが，各表情筋の定性的な動作が主にまとめられているだけで，定量的な情報は乏しい．そこで本研究所では，マーカを顔皮膚表面に貼付し，ステレオカメラを用いて，直接的に表情筋の変形を測定し，顔モデルにおいて表情を再現する試みを開始している [6]．この検討では，マーカからの距離に応じた重みを，ワイヤーフレームモデルの各ノードに与え，適宜 3 次元動きベクトルを求めていたが，各ノードの各時刻における位置計測（推定）という観点からは不十分であった．

そこで，本報告では，マーカの位置に基づき，顔の皮膚表面を B-スプライン曲面近似することにより，ワイヤーフレームモデルの各ノードの位置推定を行なう手法を検討する．

本報告は 6 章から構成される．まず，2 章では，臨場感通信会議における人物像処理について概説し，3 章では，表情変化の計測すなわち人物の顔に貼付されたマーカの計測方法について説明し，4 章では，ワイヤーフレームモデルの各ノードの各時刻における位置推定に用いられる B-スプライン曲面近似について述べる．次に，5 章では，実験結果と考察を示し，最後に，6 章で，全体のまとめと今後の課題について述べる．

Chapter 2

臨場感通信会議における人物像処理

2.1 臨場感通信会議

図2.1は臨場感通信会議のイメージを示したものである．3次元コンピュータグラフィックス技術を用いて，仮想的な空間を実時間で生成し，この仮想空間を会議の“場”とし，この中に遠隔地の人々を合成表示する．この時，離れた場所にいる3人があたかも一箇所にいるような感覚をもつ環境を提供するために，仮想空間に含まれる会議室レイアウトやインテリア，協調作業物体，および他の場所にいる会議参加者は立体表示される．図2.1は，例えば，京都，大阪，東京のように，離れて存在する人々がある一つの場所に集まったかのような感覚をもちつつ，互いに協調して車のデザインをする様子を示している．

このように，臨場感通信会議では，協調作業物体や会議参加者は立体表示されるため，3次元的にモデリングされている必要がある．特に，人物像モデルは，表情変化や体の動きを実時間で表現できなければならない．そこで，臨場感通信会議システムでは，送信側の人物像の受信側における表示は，ワイヤーフレームモデルにカラーテクスチャマッピングしたものを立体表示することにより行なわれる．また，表情変化を実時間で再現することに関しては，送信側で，人物の時間的変化情報を実時間で検出し，これらを受信側に送信し，ワイヤーフレームモデルを駆動することで実現される．このようにして生成された3次元の人物像モデルは，3次元環境データから構成される仮想空間に配置され，受信側の立体ディスプレイに3次元表示される．このようなシステム構成にすることにより，従来のテレビ会議で困難であった運動視差の再現や，会議参加者同士の視線一致の実現が可能となる．

以下，臨場感通信会議において重要な参加者の人物像の実時間表示方法について述べる．

2.2 3次元人物像の実時間表示

人物の3次元モデルに関する処理の流れを図2.2に示す．同図に示すように，このシステムは人物像の3次元モデル生成部，人物像の動きの検出部，人物像の生成部から構成されている．なお，同図は，人物の動き検出部と生成を実時間で行なう実験システム [2] のブロック図を示している．以下に各々の処理を説明する．

2.2.1 人物像の3次元モデリング

会議参加者の人物像は，あらかじめ3次元モデリングしておく必要がある．このモデリングは，人体の各パーツ毎に行なわれる．

レーザレージスキャナ等を用いて，人体のパーツの表面の3次元点データの集合を取得した後，

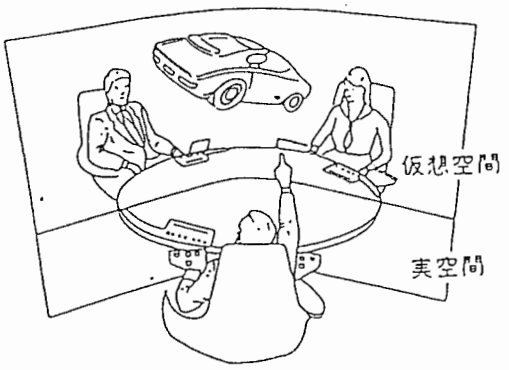


Figure 2.1: 臨場感通信会議のイメージ

utility software によりこれを三角パッチの集合体であるワイヤーフレームモデルに変換する [1]. 同時に，人体パーツ表面のカラーテクスチャも獲得しておく．カラーテクスチャ情報は，対応する場所の三角パッチに貼り付けられ，三角パッチの変形に応じて，色彩の補間や間引きが行なわれる [1].

2.2.2 人物像の動きの実時間検出

図 2.2に示すように，人物の動きとして，表情変化および頭，指，体のそれぞれの動きを検出する．

顔の表情変化の検出は，図 2.2に示すように，実際の人物の顔に存在する表情筋に対応する皮膚表面に青色のマーカを貼り，顔画像中において，これらのマーカを追跡することにより行なう．なお，顔画像を常に顔に対して一定の位置から入力するために，顔画像用のＴＶカメラは，会議参加者が装着する図 2.2のヘルメットに固定されている．もちろん，マーカやヘルメットは，自然な人間同士のコミュニケーションのためには望ましくないが，現時点では実時間性を追求するために使用している．

人物の頭部と体の 3 次元空間における移動と回転は，それぞれの部分に装着した磁気センサにより検出する．例えば，指の動き（曲がり具合）は，データグローブを装着することにより検出する．

2.2.3 人物像の実時間生成

3 次元人物像における表情の再現は，2.2.2節で示したマーカの追跡結果を利用して行なわれる．本来顔は 3 次元構造をもつが，ＴＶカメラにより獲得される画像中ではマーカの移動が 2 次元の動きとして検出しされるため，3 次元モデルを駆動するためには，拘束条件が必要である．顔に貼付された複数のマーカのうち，基準となるマーカを決定する．そのマーカを不動の基準点とし，無表情時における各マーカとの距離を求めておき，表情変化にともない距離変化が生じたマーカの動き情報に基づき，いくつかの motion rule[1] を用意しておき，これより 3 次元モデルを駆動する．本報告の検討は，現状の motion rule の高度化にも有効であると考えられる．

頭および体に取り付けられた磁気センサから得られる 6 自由度の姿勢情報は，それぞれの人体パーツに与えられ，ワイヤーフレームモデルの変形に用いられる．同様に，データグローブからの指の曲がり度合の情報に基づき，指の 3 次元モデルの曲げが決定される．

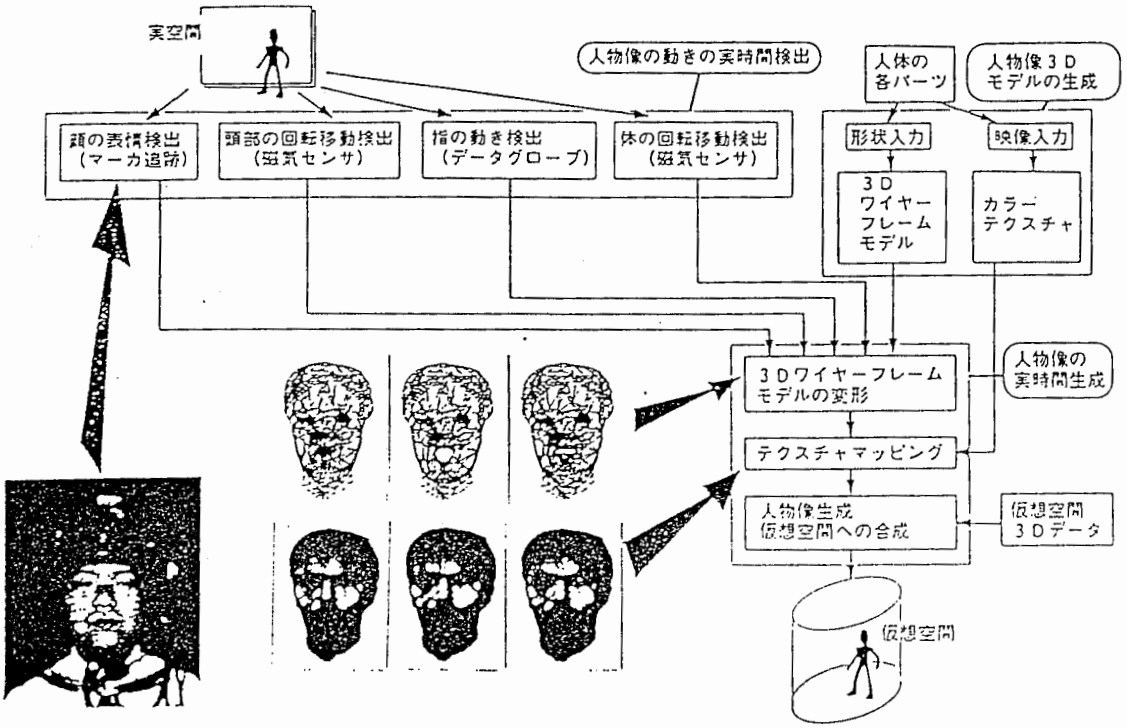


Figure 2.2: 人物の 3 次元モデルに関する処理の流れ

Chapter 3

表情変化の計測

3.1 表情筋

1章でも述べたように，より自然な表情を生成するためには，顔の筋肉の動作を 3 次元顔モデルに反映させることが考えられる．すなわち，3 次元顔モデルを駆動するためには，様々な表情変化時における皮膚表面の時間的位置を知る必要がある．そこで，顔の表情変化の検出方法として，実際の人物の顔の皮膚表面に青色のマーカを貼り，各時刻におけるマーカの位置を後述のように，ステレオカメラで計測することにした．マーカは人物の顔に存在する表情筋に対応する皮膚表面に貼付した．

人物の表情変化に関連する筋肉，すなわち表情筋は主として顔面の皮下に広く存在し，喜怒哀楽など様々な表情はこの働きによるものである．図 3.1 に示すように，表情筋は前頭筋 (p1) に代表される平行型と，眼輪筋 (c1) に代表される楕円型に大別できる．この表情筋の動きが 3 次元モデルに反映するようにマーカを皮膚表面に貼付した．

3.2 顔に貼付したマーカ位置とワイヤーフレームモデルのノードとの関係

本報告におけるマーカの貼付位置を図 3.2に示す．例えば，図 3.1に示したように，前頭筋は平行型であるため，額部分のマーカは格子状に配列した．また，眼輪筋は楕円型であるため，眼周辺部分のマーカは楕円状に配置した．他の部分のマーカについても同様に，表情筋に沿うようにマーカを貼付した．また，マーカは顔片面（左半面）で 56 箇所に貼付し，ここでは顔の表情は左

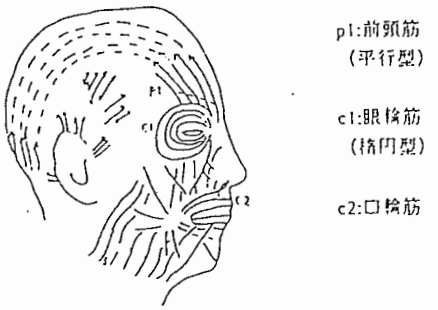


Figure 3.1: 表情筋 [3]

右対称であると仮定ししたため，顔片面の測定のみ行なっている．

Figure 3.2: マーカ貼付位置

顔の表情変化を実時間で検出するために，マーカの位置はカラーＴＶカメラにより，各時刻において計測される．一方，ワイヤーフレームモデルは2章で述べたように，レーザレージスキャナ等を用いて，あらかじめ取得された顔表面の3次元データから生成される．従って，マーカとワイヤーフレームモデルは独立であり，一般的には，マーカは，ワイヤーフレームモデルのノードとは必ずしも対応していない．すなわち，表情を生成するために，ワイヤーフレームモデルのノードを駆動するには，各時刻におけるマーカの位置情報からワイヤーフレームモデルのノードの位置を推定する必要がある．

さて，マーカの位置はカラーＴＶカメラにより計測されると述べたが，3次元上のマーカ位置を，2次元座標系のＴＶカメラを用いてどのように計測するかを次章に述べる．

3.3 マーカ位置の3次元計測

3.3.1 測定原理

ワイヤーフレームモデルを駆動するには，各時刻のマーカの位置を知る必要がある．そこで，各表情発生時の皮膚表面の動きを，3次元的に捉えなければならない．ここでは，一般的に，対象の3次元形状を得るために用いられる2台のＴＶカメラを用いるステレオカメラによる手法を用いる．その原理を図3.3に示す．なお，同図に示すように本報告では，正面と側面の画像を用いる．

点Ｐを顔の皮膚表面上の点とすると，Ｐは2台のカメラO_fとO_sにより，それぞれ画像面上の点P_fとP_sとして観測される．従って，O_fとP_fを結ぶ直線と，O_sとP_sを結ぶ直線の交点から，Ｐの3次元座標が求められる．3次元座標を求める計算式等の詳細は文献[6]を参考されたい．

カメラO_f，O_sから得られる画像中において，P_fとP_sが互いに対応付けられる点であることを見い出す必要があるが，これを自動的に行なうのは困難である．すなわち，正面と側面画像にマーカが多数存在するので，誤対応付けが発生する確立が高い．そこで，本報告では，以下に述べる半自動的対応付けを行なう．

3.3.2 マーカ位置の半自動マッチング

まず，初期フレームに対するマーカ位置の設定を手動で行なう．すなわち，図3.4に示すような各マーカの位置を番号順に指定する．指定はマウスカーソルをマーカの位置に持っていき，ボタンをクリックすることで行なわれる．設定された座標には×印とそのマーカの番号が表示される．

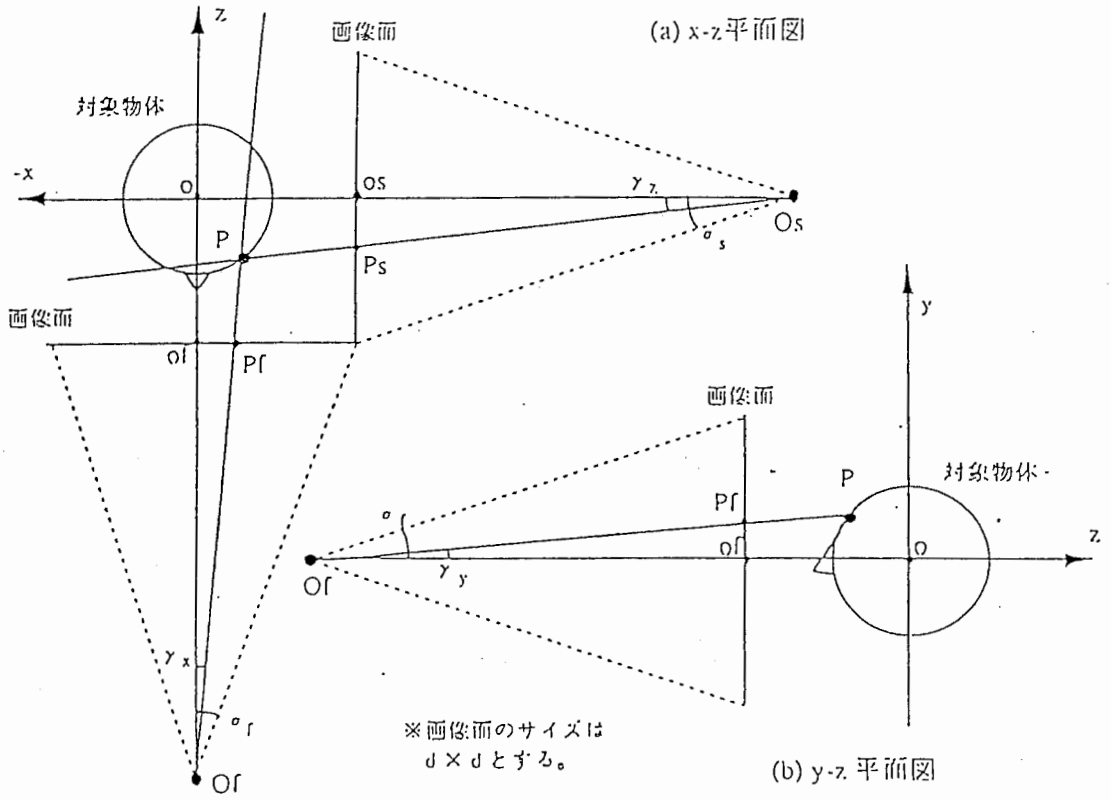


Figure 3.3: ステレオ計測の原理

この作業をすべてのマーカに対して行なうことで，正面画像と側面画像のマーカの対応付けが行なわれる．第2フレーム以降については，一般にはマーカが画像中で移動するため，マーカのもつ色の情報を用いて追跡し，自動的にマーカの重心を求める．すなわち，直前のフレームで指定された座標の周辺の画素について，色が類似した画素の重心を計算する．従って，直前のフレームでの対応付け情報は引き継がれる．しかし，光の反射等によりマーカの色が著しく変化した場合，自動的に設定された座標は実際のマーカ位置と大きく異なる場合がある．このような場合には自動的に設定された×印をマウスでクリックし，また移動先をクリックすることで，正確なマーカの位置が手動で設定される．これをすべてのフレームに対し行なうことで，マーカの3次元座標を取得することができる．

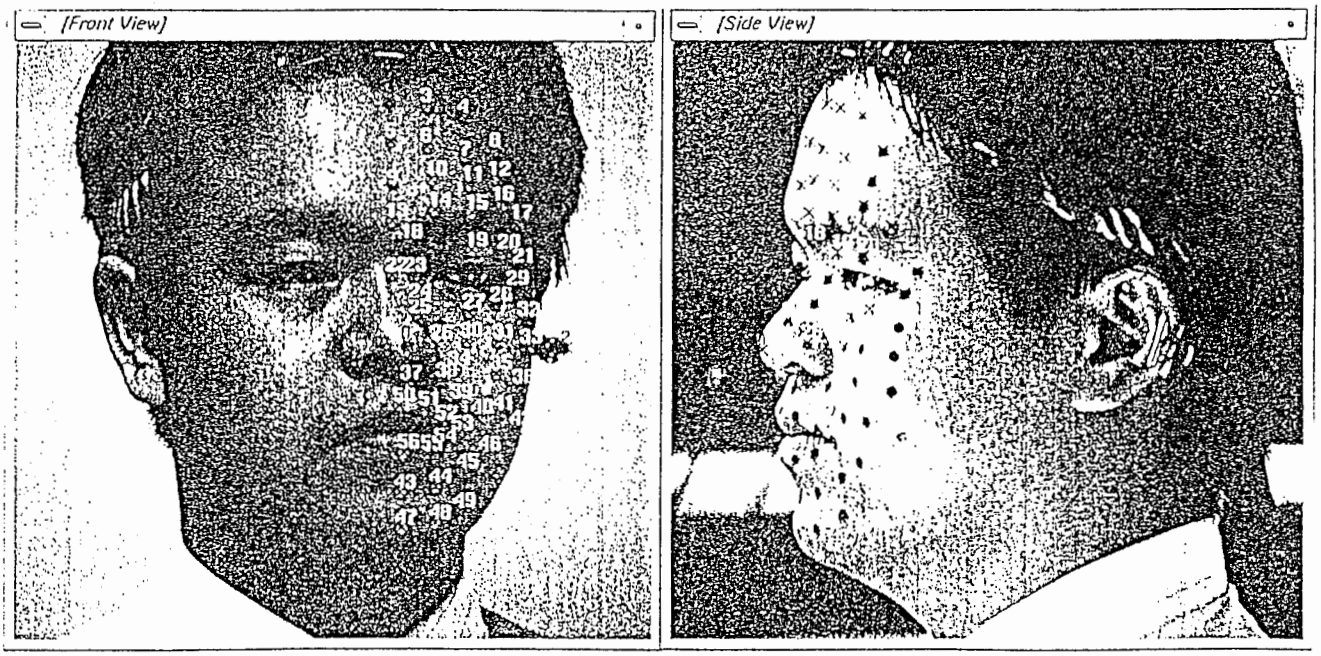


Figure 3.4: マーカ位置の設定

Chapter 4

B–スプライン曲面近似

4.1 はじめに

3章でも述べたように，マーカの位置とワイヤーフレームモデルのノードの位置は必ずしも対応していない．そこで，各時刻におけるマーカ位置の情報からワイヤーフレームモデルのノードの位置を推定する必要がある．本研究では，マーカを通過する曲面を用いて，皮膚表面を近似し，この曲面上にワイヤーフレームモデルのノードが存在すると仮定して，各ノードの位置を推定する．ここで，マーカを通過する曲面近似には，以下に述べるような B–スプライン近似を用いる．

4.2 B–スプライン

B–スプライン近似とは制御点と呼ばれる点の集合を用いて，滑らかな曲線を形成する近似手法である．図 4.1に示すように，制御点は磁石のように，曲線をある方向へ引っ張る役割を果たす．

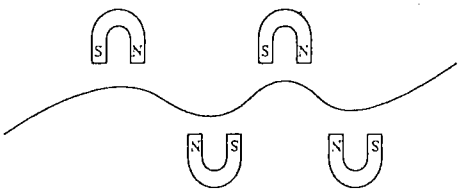


Figure 4.1: 制御点の曲線に対する影響

4.2.1 B–スプライン曲線

順序付けられた$n+1$個の位置ベクトルQ_0，Q_1，...，Q_{n-1}，Q_nが与えられた場合（図 4.2），それらの 4 つの点を順次一組としてできる$n-2$個の 1 次結合

$$P_i(t) = X_0(t)Q_{i-1} + X_1(t)Q_{i+1} + X_3(t)Q_{i+2} \quad (i = 1, 2, \ldots, n-2) \tag{4.1}$$

について考える．

$X_0(t)$，$X_1(t)$，$X_2(t)$，$X_3(t)$ をパラメータ $t(0 \leq t \leq 1)$ の多項式とする．$P_i(t)$ はパラメータの変化に対し一つの曲線セグメントを表す．隣合う曲線セグメント$P_i(t)$，$P_{i+1}(t)$ が，それぞれ

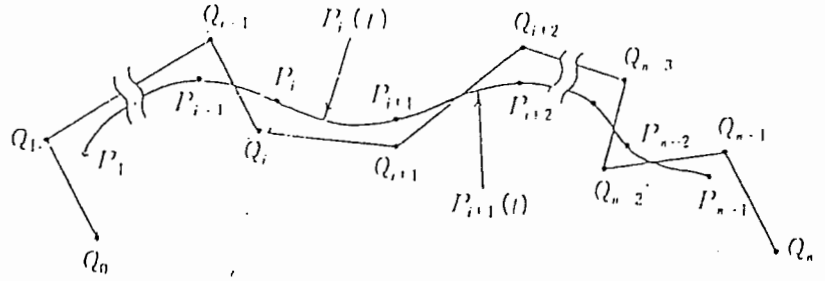


Figure 4.2: B–スプライン曲線の誘導 (M=4 の場合)

$t=1$, $t=0$ で連続であるための条件，すなわち，$P_i(1)=P_{i+1}(0)$ が$Q_j(j=i-1,i,\ldots,i+3)$ にかかわらず成立するための条件は，

$$\begin{aligned} X_0(1) &= X_3(0) = 0 \\ X_1(1) &= X_0(0) \\ X_2(1) &= X_1(0) \\ X_3(1) &= X_2(0) \end{aligned} \tag{4.2}$$

となる．同様にして 1 階の導関数（接線ベクトル），2 階の導関数ベクトルが連続である条件，すなわち$\dot{P}_i(1)=\dot{P}_{i+1}(0)$，$\ddot{P}_i(1)=\ddot{P}_{i+1}(0)$ が$Q_j(j=i-1,i,\ldots,i+3)$ にかかわらず成立するための条件は，それぞれ

$$\begin{aligned} \dot{X}_0(1) &= \dot{X}_3(0) = 0 \\ \dot{X}_1(1) &= \dot{X}_0(0) \\ \dot{X}_2(1) &= \dot{X}_1(0) \\ \dot{X}_3(1) &= \dot{X}_2(0) \end{aligned} \tag{4.3}$$

$$\begin{aligned} \ddot{X}_0(1) &= \ddot{X}_3(0) = 0 \\ \ddot{X}_1(1) &= \ddot{X}_0(0) \\ \ddot{X}_2(1) &= \ddot{X}_1(0) \\ \ddot{X}_3(1) &= \ddot{X}_2(0) \end{aligned} \tag{4.4}$$

である．さらに座標変換に対し形状が不変であるための条件

$$X_0(t)+X_1(t)+X_2(t)+X_3(t) \equiv 1 \tag{4.5}$$

を追加する．これらの条件から $X_0(t)$，$X_1(t)$，$X_2(t)$，$X_3(t)$ の関数形は

$$\begin{aligned} X_0(t) &= \tfrac{1}{6}(1-t)^3 \\ X_1(t) &= \tfrac{1}{2}t^3 - t^2 + \tfrac{2}{3} \\ X_2(t) &= -\tfrac{1}{2}t^3 + \tfrac{1}{2}t^2 + \tfrac{1}{2}t + \tfrac{1}{6} \\ X_3(t) &= \tfrac{1}{6}t^3 \end{aligned} \tag{4.6}$$

である．これらの関数を用いて，曲線セグメント式 (4.1) を表現すると

$$P_i(t) \;=\; X_0(t)Q_{i-1} + X_1(t)Q_{i+1} + X_3(t)Q_{i+2}$$

$$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} \\ \frac{1}{2} & -1 & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \end{bmatrix} \begin{bmatrix} Q_{i-1} \\ Q_i \\ Q_{i+1} \\ Q_{i+2} \end{bmatrix}$$

$$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} M_R \begin{bmatrix} Q_{i-1} \\ Q_i \\ Q_{i+1} \\ Q_{i+2} \end{bmatrix} \tag{4.7}$$

ここに，

$$M_R = \begin{bmatrix} -\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} \\ \frac{1}{2} & -1 & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \end{bmatrix} \tag{4.8}$$

となる．本報告ではこれらの関数を $N_{i,4}(t)$ で表している．すなわち，

$$X_0(t) \equiv N_{0,4}(t), X_1(t) \equiv N_{1,4}(t)$$
$$X_2(t) \equiv N_{2,4}(t), X_3(t) \equiv N_{3,4}(t) \tag{4.9}$$

また，以上の誘導から明らかなように，この曲線では，$P_j(j = 0, 1, \ldots, n)$ にかかわらず接点でのセグメントの連続性が常に保証される．

4.2.2 曲線の逆変換

B-スプライン曲面のベクトル$P_i(i = 1, 2, \ldots, n-1)$が与えられて，曲面定義ベクトル$Q_j(j = 0, 1, \ldots, n)$を求める逆変換問題を考える（図4.2）．また閉じた曲線の場合には，$P_i(i = 0, 1, \ldots, n)$が与えられて，$Q_j(j = -1, 0, \ldots, n, n+1,$ だたし，実質的には$j = 0, 1, \ldots, n)$ を求める問題とする（図4.3）．

開いた曲線，閉じた曲線のいずれの場合においても，つぎの連立方程式が成立する．

$$\frac{1}{6}Q_{i-1} + \frac{2}{3}Q_i + \frac{1}{6}Q_{i+1} = P_i \left(\begin{array}{l} i = 1, 2, \ldots, n-1 \text{（開いた曲線）} \\ i = 0, 2, \ldots, n \text{（閉じた曲線）} \end{array} \right) \tag{4.10}$$

開いた曲線と閉じた曲線のそれぞれの場合につぎの条件設定を行なう．すなわち，

1. 開いた曲線発生の場合：$Q_0 = Q_1, Q_n = Q_{n-1}$ (4.11)
2. 閉じた曲線発生の場合：$Q_{-1} = Q_n, Q_{n+1} = Q_0$ (4.12)

条件（4.11）は，開いた曲線の両端において曲率を0とする条件である．この設定により，逆変換アルゴリズムが，閉じた曲線の場合と同様な形式に単純化する．また条件（4.12）は，曲線の始点と終点を曲率ベクトルまで連続に接続して閉じた曲線を作るためのものである．条件（4.11），(4.12）をそれぞれの場合に追加することにより，方程式（4.10）を解くことができる．以下に逆変換のアルゴリズムを示す（図4.4）．

x, y, zの各座標について以下の処理を実行する．

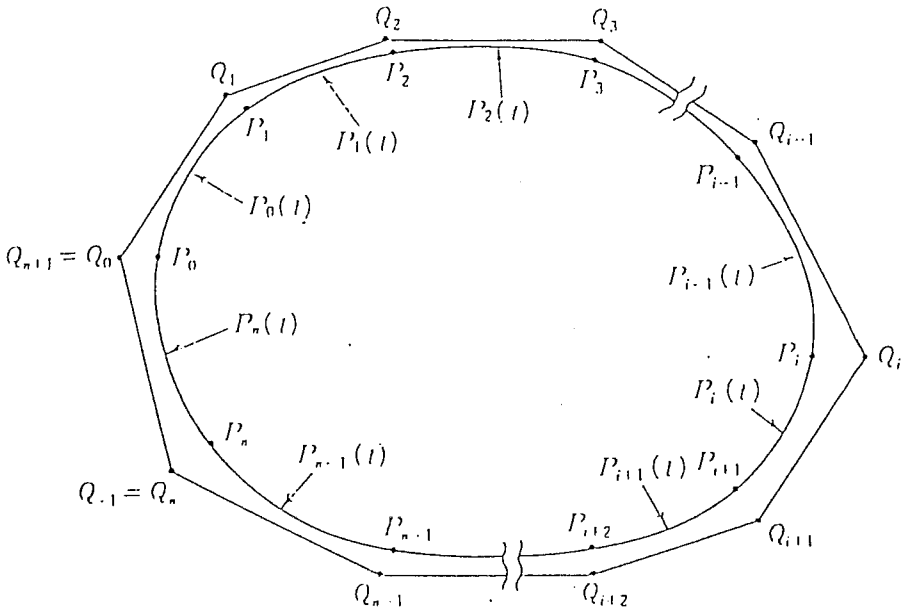


Figure 4.3: 曲線 (M = 4) の逆変換 (閉じた曲線の場合)

ステップ 1
$i = 1, 2, \ldots, n-1 (i = 0, 1, \ldots, n)$ に対し
$P_i \rightarrow Q_i$
そのあと $Q_1 \rightarrow Q_0, Q_{n-1} \rightarrow Q_0, Q_{n-1} \rightarrow Q_n (Q_n \rightarrow Q_{-1}, Q_0 \rightarrow Q_{n+1})$

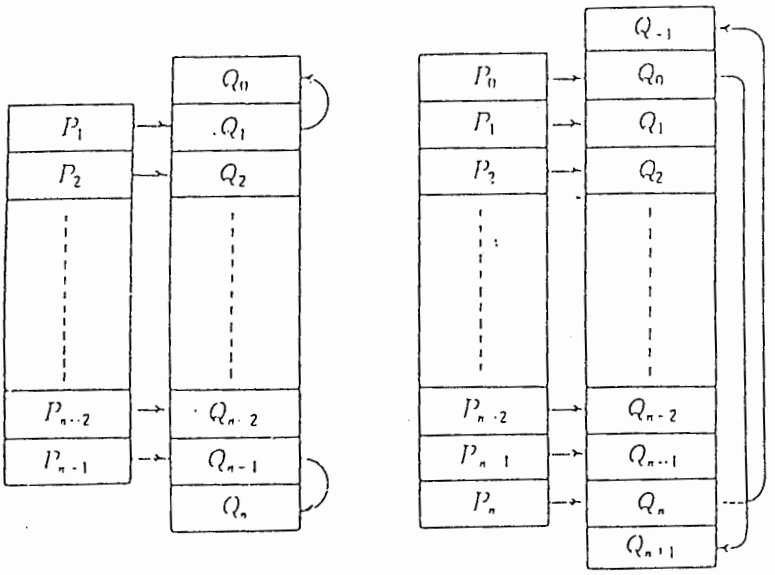
ステップ 2
$\delta_i = P_i - Q_i + \frac{1}{2}\{P_i - \frac{1}{2}(Q_{i-1} + Q_{i+1})\}\delta_i + Q_i \rightarrow Q_i$ (4.13)
そのあと $Q_1 \rightarrow Q_0, Q_{n-1} \rightarrow Q_0, Q_{n-1} \rightarrow Q_n (Q_n \rightarrow Q_{-1}, Q_0 \rightarrow Q_{n+1})$

ステップ 3
$\max \delta_i > \delta_s$ ならステップ 2 に戻る.
$\max \delta_i \leq \delta_s$ なら終了.

4.2.3 B-スプライン曲面

B-スプライン関数を用いて, 3 次式曲面パッチを定義することができる (図 4.5). すなわち, 図 4.5に示すように, 16 個の頂点より成るネットにより, 一つの B-スプライン曲面パッチが定義される.

$$P_{i,j}(u,w) = \left[N_{0,4}(u) \quad N_{1,4}(u) \quad N_{2,4}(u) \quad N_{3,4}(u) \right]$$



（a） 開いた曲線を発生する場合　（b） 閉じた曲線を発生する場合

Figure 4.4: 逆変換アルゴリズムにおける P–テーブルと Q–テーブル

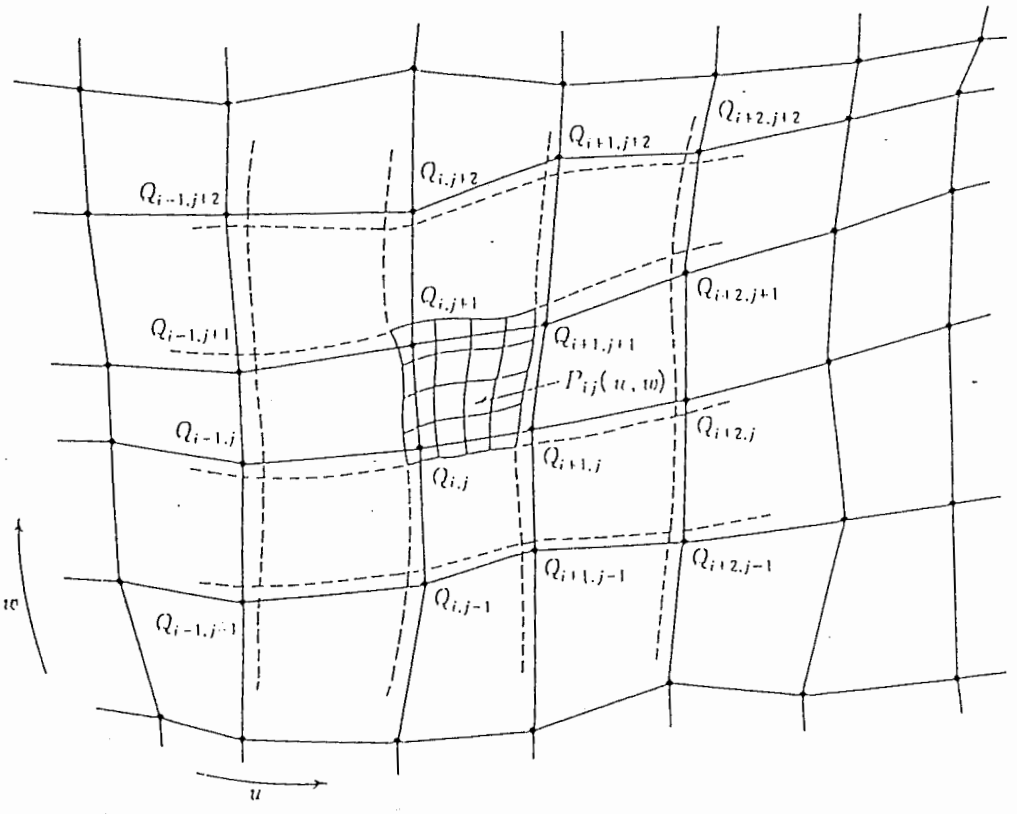


Figure 4.5: B–スプライン曲面パッチと曲面定義ベクトル

$$\times \begin{bmatrix} Q_{i-1,j-1} & Q_{i-1,j} & Q_{i-1,j+1} & Q_{i-1,j+2} \\ Q_{i,j-1} & Q_{i,j} & Q_{i,j+1} & Q_{i+1,j+2} \\ Q_{i+1,j-1} & Q_{i+1,j} & Q_{i+1,j+1} & Q_{i+1,j+2} \\ Q_{i+2,j-1} & Q_{i+2,j} & Q_{i+2,j+1} & Q_{i+2,j+2} \end{bmatrix} \begin{bmatrix} N_{0,4}(u) \\ N_{1,4}(u) \\ N_{1,4}(u) \\ N_{3,4}(u) \end{bmatrix} \tag{4.14}$$

$$= UM_R B_R M_R^T W^T \tag{4.15}$$

ここに

$$B_R(u,w) = \begin{bmatrix} Q_{i-1,j-1} & Q_{i-1,j} & Q_{i-1,j+1} & Q_{i-1,j+2} \\ Q_{i,j-1} & Q_{i,j} & Q_{i,j+1} & Q_{i+1,j+2} \\ Q_{i+1,j-1} & Q_{i+1,j} & Q_{i+1,j+1} & Q_{i+1,j+2} \\ Q_{i+2,j-1} & Q_{i+2,j} & Q_{i+2,j+1} & Q_{i+2,j+2} \end{bmatrix} \tag{4.16}$$

$$U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

$$W = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

この曲面パッチは，一般に，与えられた曲面定義ベクトル（ネットの頂点）のいずれをも通過しない．曲面パッチの4隅の点は，4つの曲線定義ベクトル$Q_{i,j}$，$Q_{i,j+1}$，$Q_{i+1,j}$，$Q_{i+1,j+1}$の近傍にある．

4.2.4 曲面の逆変換

格子状の点列$P_{i,j}(j = 1,2,\ldots,m-1; j = 1,2,\ldots,n-1)$が与えられるとき，それらの点を曲面パッチの4隅の点とする，B-スプライン曲面式（4.15）の定義ベクトルは，4.2.2で述べた，曲線の逆変換アルゴリズムを用いて求めることができる（図4.6）．

B-スプライン関数は

$$N_{0,4}(0) = 1/6, \quad N_{1,4}(1) = 2/3, \quad N_{2,4}(1) = 1/6, \quad N_{3,4}(1) = 0$$
$$N_{0,4}(1) = 0, \quad N_{1,4}(1) = 1/6, \quad N_{2,4}(1) = 2/3, \quad N_{3,4}(1) = 1/6$$

であるので，つぎの関係式が成立する．

$$\begin{aligned} P_{i,j} &= \tfrac{1}{6}V_{i,j-1} + \tfrac{2}{3}V_{i,j} + \tfrac{1}{6}V_{i,j+1}(1 \le i \le m-1; 1 \le j \le n-1) \\ V_{i,j} &= \tfrac{1}{6}Q_{i-1,j} + \tfrac{2}{3}Q_{i,j} + \tfrac{1}{6}Q_{i+1,j}(1 \le i \le m; 1 \le j \le n-1) \end{aligned} \tag{4.17}$$

または，

$$\begin{aligned} P_{i,j} &= \tfrac{1}{6}U_{i,j-1} + \tfrac{2}{3}U_{i,j} + \tfrac{1}{6}U_{i,j+1}(1 \le i \le m-1; 1 \le j \le n-1) \\ U_{i,j} &= \tfrac{1}{6}Q_{i-1,j} + \tfrac{2}{3}Q_{i,j} + \tfrac{1}{6}Q_{i+1,j}(1 \le i \le m; 1 \le j \le n-1) \end{aligned} \tag{4.18}$$

式（4.17）または式（4.18）のいずれの関係を用いても，同一の$Q_{i,j}$を決定することができるが，以下には式（4.17）を利用して，逆変換の手順を説明する．

まず，u方向の$P_{1,1}$，$P_{2,1}$，…，$P_{m-2,1}$，$P_{m-1,1}$なる$m-1$個の点列に対し曲線の逆変換アルゴリズムを適用すると，式（4.17）の関係があるので$V_{0,1}$，$V_{1,1}$，…，$V_{m-1,1}$，$V_{m-1,2}$なる点列が発生する．同様にして，u方向の2番目の点列$P_{1,2}$，$P_{2,2}$，…，$P_{m-2,2}$，$P_{m-1,2}$なる点列が発生する．同様にして，u方向の合計$n-1$組の点列に対し，$n-1$回曲線の逆変換アルゴリズムを適用すると，図4.7に示すような$V_{i,j}$の格子状の点列が作られる．次に，図のw方向の$m+1$組の点列に注目する．まず，最初の点列$V_{0,1}$，$V_{0,2}$，…，$V_{0,n-1}$に対し曲線の逆変換アルゴリズムを適用すると，式（4.17）の関係があるので$Q_{0,0}$，$Q_{0,1}$，$Q_{0,2}$，…，$Q_{0,n-1}$，$Q_{0,n}$なる点列が発生する．同様に他のm組の点列に対しても，それぞれ逆変換アルゴリズムを適用すれば，結局

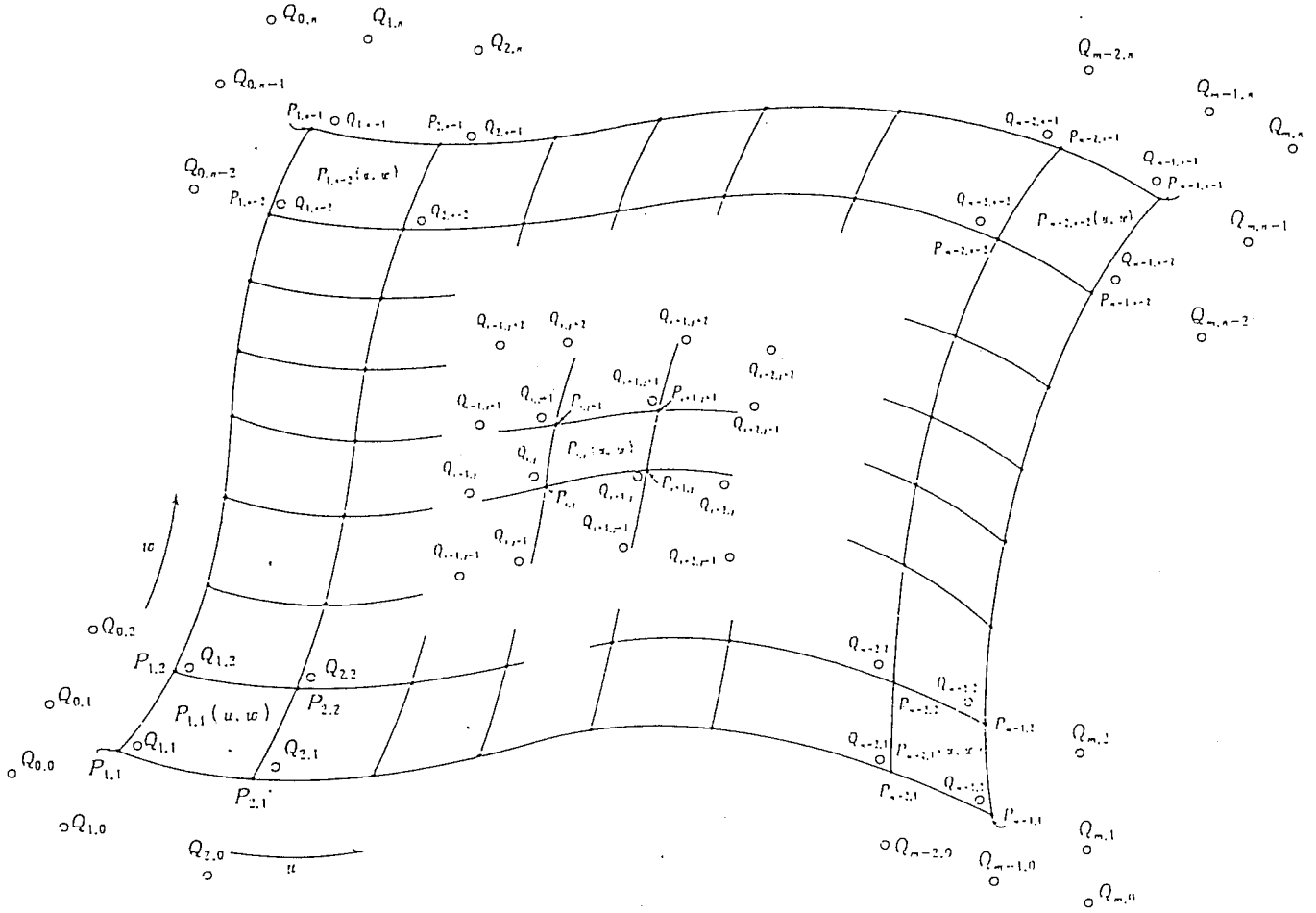


Figure 4.6: 曲面の逆変換

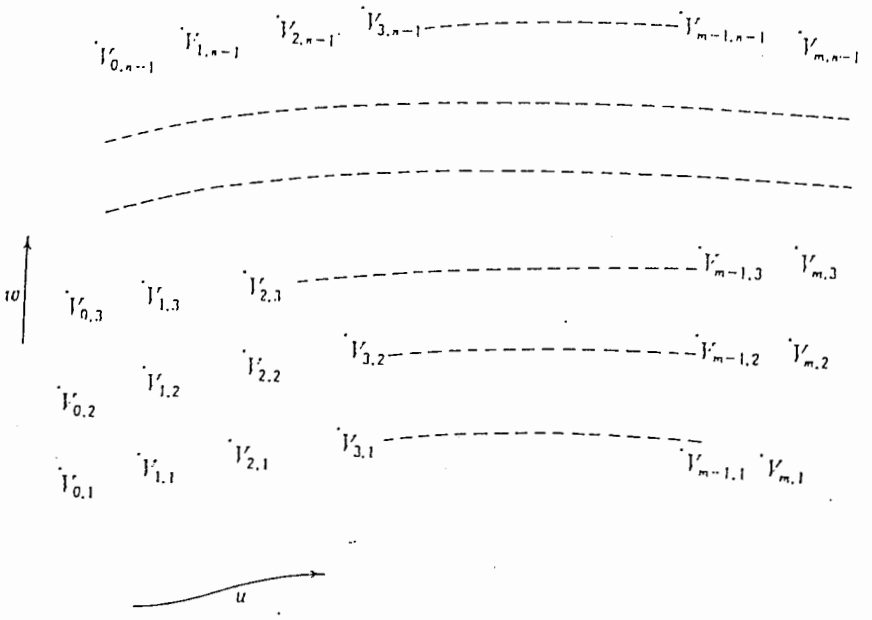


Figure 4.7: 曲面の逆変換アルゴリズムの途中で発生される V ベクトル

$Q_{i,j}(i = 0, 1, 2, \ldots, m; j = 0, j, \ldots, n)$ が発生し，これは，すなわち曲面式（4.15）における曲面定義ベクトルそのものである．

曲面の逆変換アルゴリズムの適用順序を逆とすれは，式（4.18）が成立する．すなわち，まず，$P_{i,j}$の点列の w方向に対し逆変換アルゴリズムを適用し，つぎに発生点列$U_{i,j}$の u 方向に対し，再び逆変換アルゴリズムを適用する．最終結果はいずれの方法によっても同一である．

4.3 ノードの位置推定

4.2で述べたように，B-スプライン曲面近似を行なう場合，複数の制御点から一つの曲面を近似することが可能である．ここでは，曲面を近似することのできる最小単位である 4 つの制御点から構成される曲面パッチを単位として用いることにした．B-スプライン曲線近似は 4 つの制御点から中央 2 つの制御点の近傍を通過する曲線が形成されることから，一つの B-スプライン曲面パッチを形成するためには 16 個の制御点が必要となる．その 16 個の制御点は，その曲面パッチの 4 隅に対応する 4 つの制御点とその制御点を囲む 12 個の制御点から成る．このように，16 個の制御点が与えられれば，中央 4 つの制御点を通過する曲面パッチを形成するような制御点を曲面の逆変換により求めることができる．従って，逆変換により求められた制御点から元の制御点すなわちマーカを通過する一つの曲面パッチが形成される．

顔のワイヤーフレームモデルは 2章で述べたようにして，無表情の状態で作成しておく．一方，3章で述べたステレオカメラにより顔を撮像するが，初期フレームはワイヤーフレームモデルと同様，無表情の状態を撮像する．

初期フレームにおいては，マーカを用いて形成された B-スプライン曲面を利用して，ワイヤーフレームモデルの各ノードの (u, w) の値を求める．第 2 フレーム以降については，ステレオ画像により得られるマーカの位置 (x, y, z) に基づき，曲面パッチを更新していく．ここで，通常の表情変化により生じる皮膚表面形状変化を考慮に入れると，4 つのマーカに囲まれた範囲の空間は

線形的に起こる，すなわちねじれ，逆転などがないと考えられる．従って，u,w の値は一定と考えられるので，初期フレームで求めたu,wを用いて，各時刻（フレーム）におけるノードの位置(x,y,z)を求めることができる．

このような方法により，各フレームでのワイヤーフレームモデルの各ノードの位置をすべて推定することができる．

Chapter 5

実験結果と考察

5.1 実験条件

図3.1に示したように，顔には人物の表情変化に関連する表情筋が存在しており，その動きを追跡するためには，図3.2に示したようにマーカを顔の片面（左半面）に56箇所貼付した．すなわち，ここでは顔の表情は左右対称であると仮定しているため，実質，顔全体で98箇所にマーカを貼付したことになる．

このマーカを貼付された被験者に無表情→怒り→哀しみ→喜び→驚き→嫌悪→恐怖→無表情という表情変化を15秒で表出してもらい，これを1/15秒単位でカラーＴＶカメラよりフレームメモリに記録した．従って，データサンプル数は15（秒）×15（秒）で225フレームである．

また，ワイヤーフレームモデルおよびカラーテクスチャはCyberware Color 3D Digitizerを用いて測定され，ワイヤーフレームモデルのノードは約1,600点から構成される．なお，これは無表情に対応しており，時系列データを持たない．

ところで，4.3で述べたように，4つの制御点から構成される曲面パッチを単位として用いることにしたため，98個のマーカを4点から構成される四角形に分割した（図5.1）．同図に示すように，マーカに対し，規則的に四角形を配置するのは困難な箇所も存在する．従って，どの四角形にも含まれないノードが存在することになるため，4つの制御点から構成される曲面パッチをオーバーラップさせることでこれを防いだ．

また，マーカは唇付近に貼付されているが，上唇と下唇は独立に運動すると仮定できるので，本報告では唇および目付近の近似を検討対象から除外している．

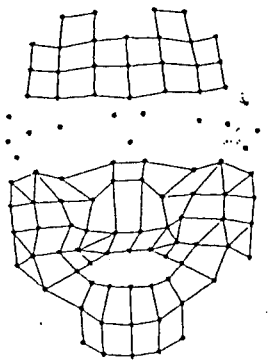


Figure 5.1: マーカの分割

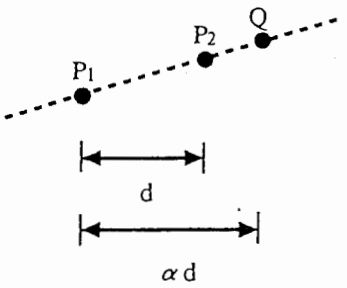


Figure 5.2: 仮想点の配置方法

Figure 5.3: 配置した仮想点

さらに，曲面パッチを形成するためには16個の制御点が必要であるため，その四角形が外縁部にあって8個の四角形で囲まれていない場合については，その四角形の周りに仮想マーカを配置した．その仮想マーカのxおよびy軸座標値は，近隣のマーカとその隣のマーカの延長線上に配置し，z軸座標値は近隣のマーカのz軸座標値と等しく設定した．すなわち，図5.2に示すように，P_1とP_2を結ぶ直線上にP_2からP_1とP_2の距離のα倍の距離離れた位置にP_2とz軸座標値が等しい仮想マーカを配置した．図5.3に配置した仮想マーカを示す．

さて，初期フレームにおいて，ワイヤーフレームモデルの各ノードのuおよびwを求めるには，あらかじめワイヤーフレームモデルの各ノードがどの4つのマーカから構成される四角形に含まれているかを調べておく．

また，第2フレーム以降，表情変化に伴い変化するワイヤーフレームモデルのノードは顔前面に対してのみであると考えられるため，変化させるワイヤーフレームモデルのノードはz軸座標値が正のもののみを対象とした（zが負のものは，頭の後部に対応）．

以上の事前準備を行なった後，4章に述べた方法で，初期フレームのワイヤーフレームモデルの各ノードの位置から，各時刻におけるマーカの位置情報を用いて，第2フレーム以降のワイヤーフレームモデルの各ノードの位置を推定した．

5.2 結果と考察

B-スプライン曲面近似を用いて，すべてのフレームでのワイヤーフレームモデルの各ノードの位置を推定した．また，各ノードの位置が推定された後，ワイヤーフレームモデルにカラーテクス

チャをマッピングした．図5.4に原画像，ワイヤーフレームモデルおよびカラーテクスチャをマッピングしたものを示す．これより，良好なノードの位置推定が行なえていることが確認された．

次に，ノードの位置推定の精度を評価した．すなわち，あるノードのx,y座標値から曲面上のu,wを求めるとき，収束計算を行なうため，誤差が生じる．そこで，4つのマーカから構成される曲面パッチの四隅の位置とマーカの位置の誤差評価を行なった．ここでは，フレーム数（225フレーム）×配置された四角形数（59個）×一つの四角形を構成する制御点数（4点）＝53100点の平均誤差，最小誤差，最大誤差を求めた．53100点の平均誤差は約0.34mmであり，最小誤差は0.00mm，最大誤差は約0.66mmであった．また，ある一つの制御点の誤差は約0.2mm～0.5mmであり，フレーム毎による差はほとんど見られなかった．このことから，推定されたワイヤーフレームモデルのノードの誤差はこの程度に抑えられていると考えることができる．

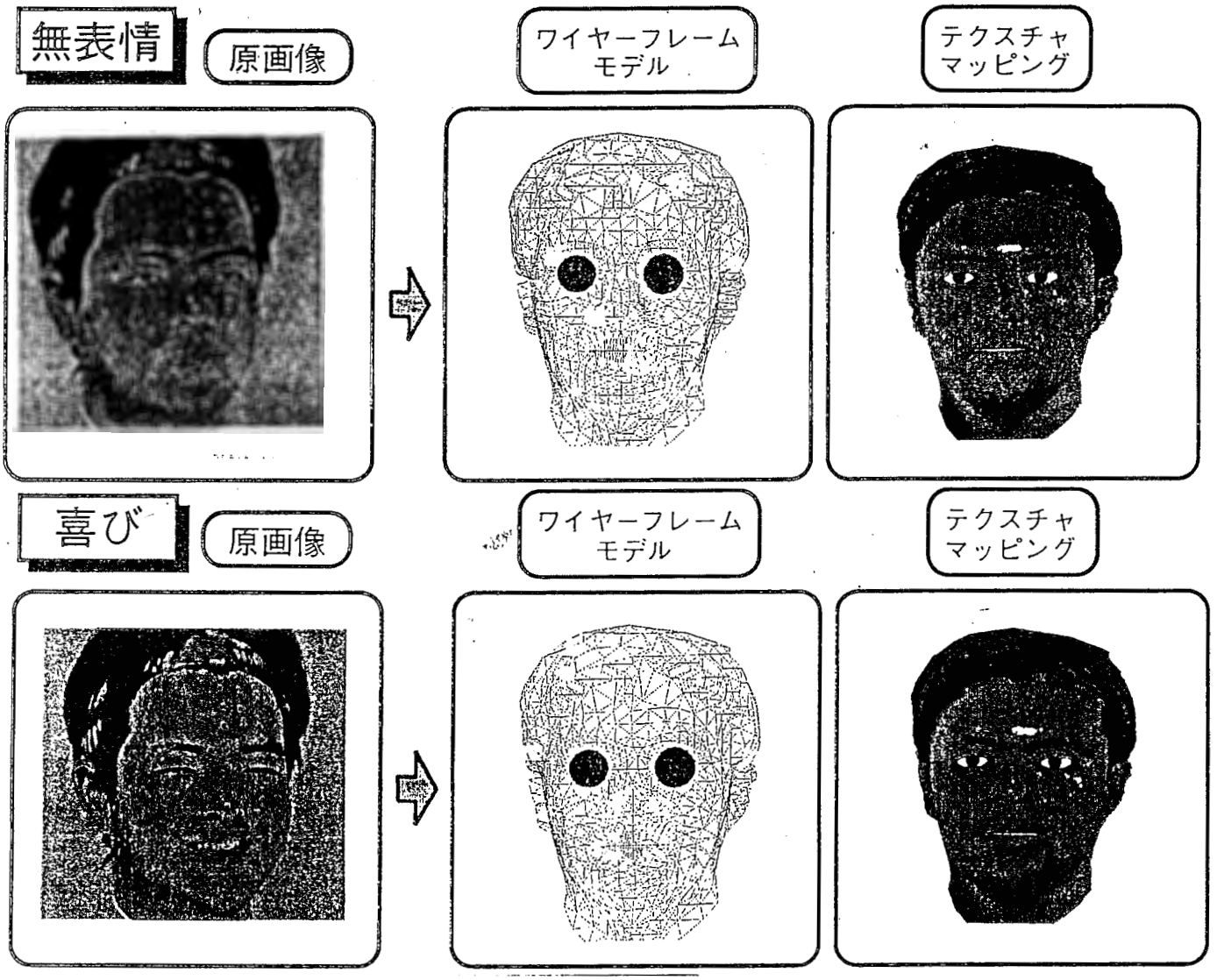


Figure 5.4: 実験結果

(a) 無表情の場合 (b) 喜びを表した場合

Chapter 6

まとめ

本研究では臨場感通信会議における 3 次元顔モデルの表情再現のため，各時刻における顔に貼付されたマーカの 3 次元位置から，各時刻におけるワイヤーフレームのノードの位置推定する手法の検討を行なった．

本研究では，互いに隣接する 4 つのマーカを単位に，これらのマーカを通る B- スプライン曲面パッチにより，顔の皮膚表面を近似し，初期フレームでは，ワイヤーフレームモデルのノードの座標値 (x,y,z) から，曲面上の (u,w) を求め，第 2 フレーム以降は初期フレームで求めた曲面上の (u,w) よりノードの絶対座標値 (x,y,z) を推定する．本手法を 225 フレームから成る表情変化画像に適用した結果，各ノードの各時刻における位置がマーカの位置情報から推定された．位置推定の精度は，曲面パッチの近似精度に依存すると考えられる．そこで，曲面の近似精度を評価するため，曲面の隅とマーカの位置との誤差評価を行なったところ，推定されたワイヤーフレームモデルのノードの誤差は 0.3mm 程度に抑えられていることが分かった．

残された課題として，単純な曲面近似が困難であると考えられる目および唇の対処法を考案することがまず挙げられる．また，今回は 4 つのマーカから構成される曲面パッチを単位としたが，さらに多くの制御点から曲面パッチを形成することで曲面の近似精度は向上すると考えられる．

さらに，今回は 6 表情のみの実験であったため，他の表情についての有効性も検討すべきである．

謝辞

本研究を進めるにあたり，御指導いただきました（株）ＡＴＲ通信システム研究所 岸野文郎室長に厚く御礼申し上げます．

また，環境設定等について御指導いただきました同所北村泰一研究員，表情筋について御助言いただきました同所梶原創一研究員に深く感謝致します．さらに，コンピュータ・グラフィックスおよびB-スプライン近似等について御指導いただきました(株) ＣＳＫ浦真吾氏ならびに始終熱心に御指導いただきました同社井村茂雄氏に感謝致します．

最後に，お世話になりました通信システム研究所知能処理研究室の研究員の皆様ならびに３Ｄ実験室でお世話になりました(株) ＣＳＫの皆様，そして，日頃より励まして下さいました企画課の皆様に厚く御礼申し上げます．

Bibliography

[1] 大谷淳, 北村泰一, 竹村治雄, 岸野文郎, "臨場感通信会議における 3 次元顔画像の実時間表示", 信学技報, HC92-61 (1993.1).

[2] 大谷淳, "コミュニケーションメディアとしての人工現実技術", CAI 学会第 18 回全国大会.

[3] 森於菟, 小川鼎三, 大内弘, 森富, "分担解剖学 1", 金原出版株式会社 (1950,1982).

[4] 梶原創一, 田中弘美, 大谷淳, 岸野文郎, "ホモトピーに基づいた三次元顔画像の表情生成の一検討", テレビジョン学会技術報告, Vol.17, No.58, pp.37–42 (1993.10).

[5] P.Ekman and W.V.Friesen "Facial Action Coding System", Consulting Psychologists Press(1977).

[6] 坂口竜巳, "表情表現を考慮した顔特徴点抽出に関する検討", ATR 通信システム研究所テクニカルレポート (1993.9).

[7] 山口富士夫, "コンピュータディスプレイによる形状処理工学 [II]", 日刊工業新聞社 (1882).

[8] 井村茂雄, "顔画像マーカ座標設定ツール使用マニュアル", (株) CSK(1993.12).

3次元顔画像生成に関する研究

(付録)

付録.1　顔に添付したマーカ位置および4つのマーカから成る四角形
付録.2　顔画像を再合成するための関数
付録.3　プログラムリスト
付録.4　作成したデータファイル

小森正美　　大谷淳
MASAMI KOMORI　JUN OHYA

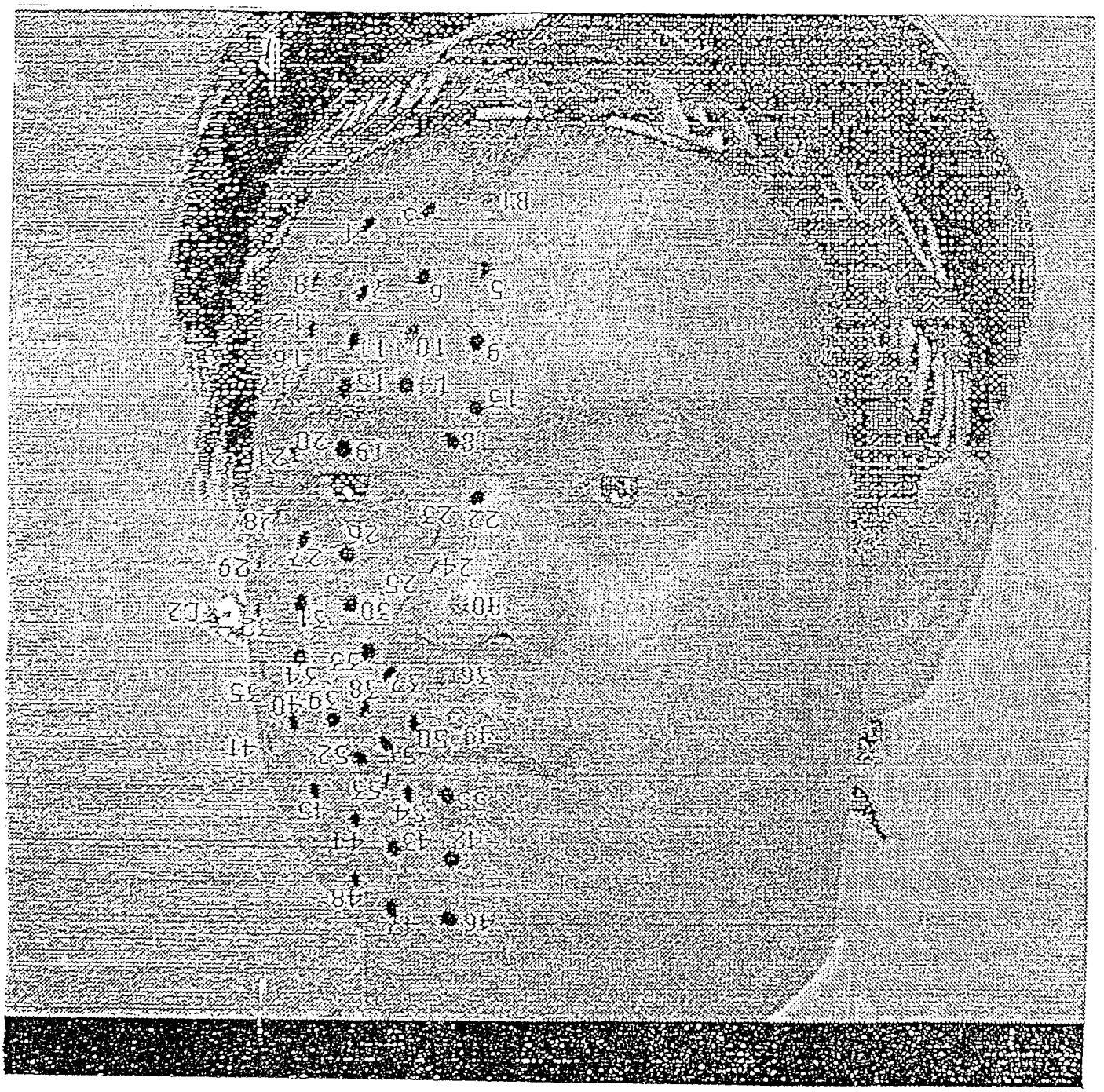
1994.2.25

ATR通信システム研究所

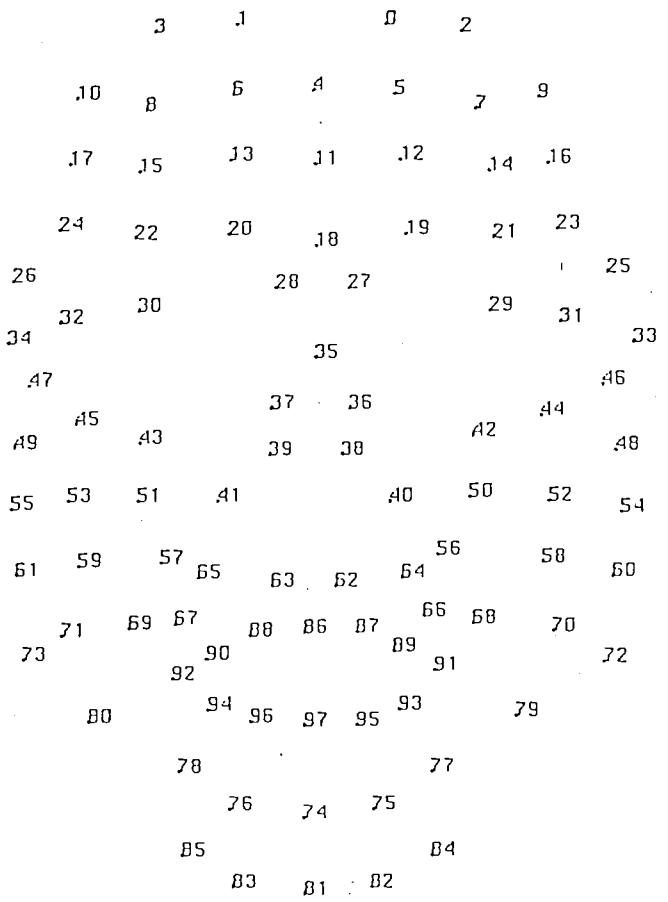
Appendix A

顔に貼付したマーカ位置および4つのマーカから成る四角形

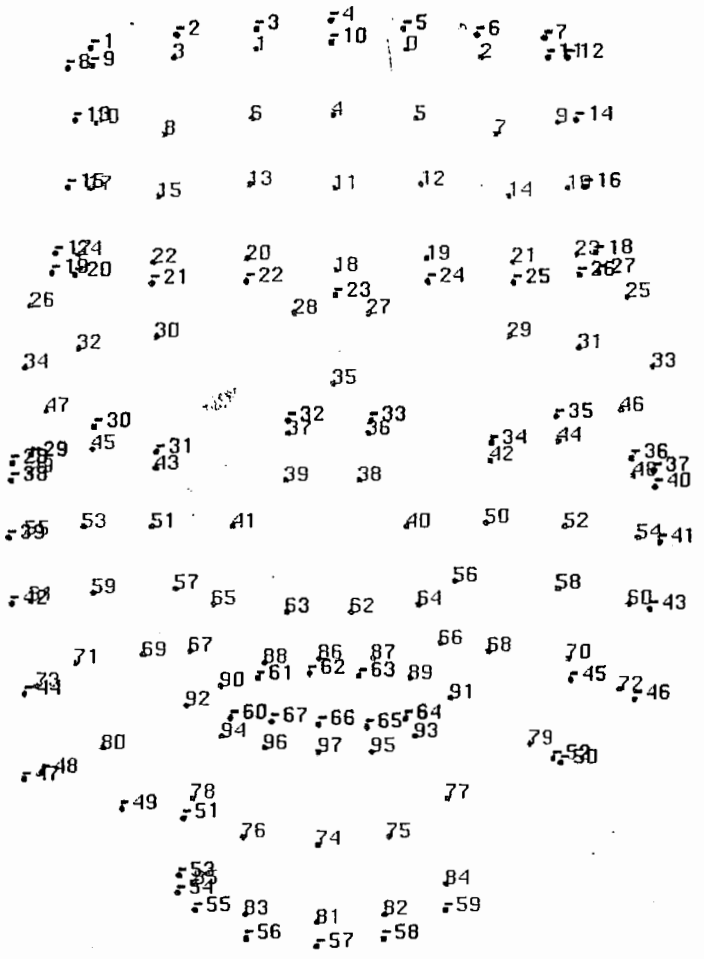
1枚目　本実験 (6 表情変化) で用いたもの (56 測定点)
2枚目　測定された顔片面のマーカ位置をもとに顔全面に配置されたマーカ位置
3枚目　B–スプライン曲面近似を用いるために配置した仮想点
4枚目　4つのマーカから成る曲面パッチの番号



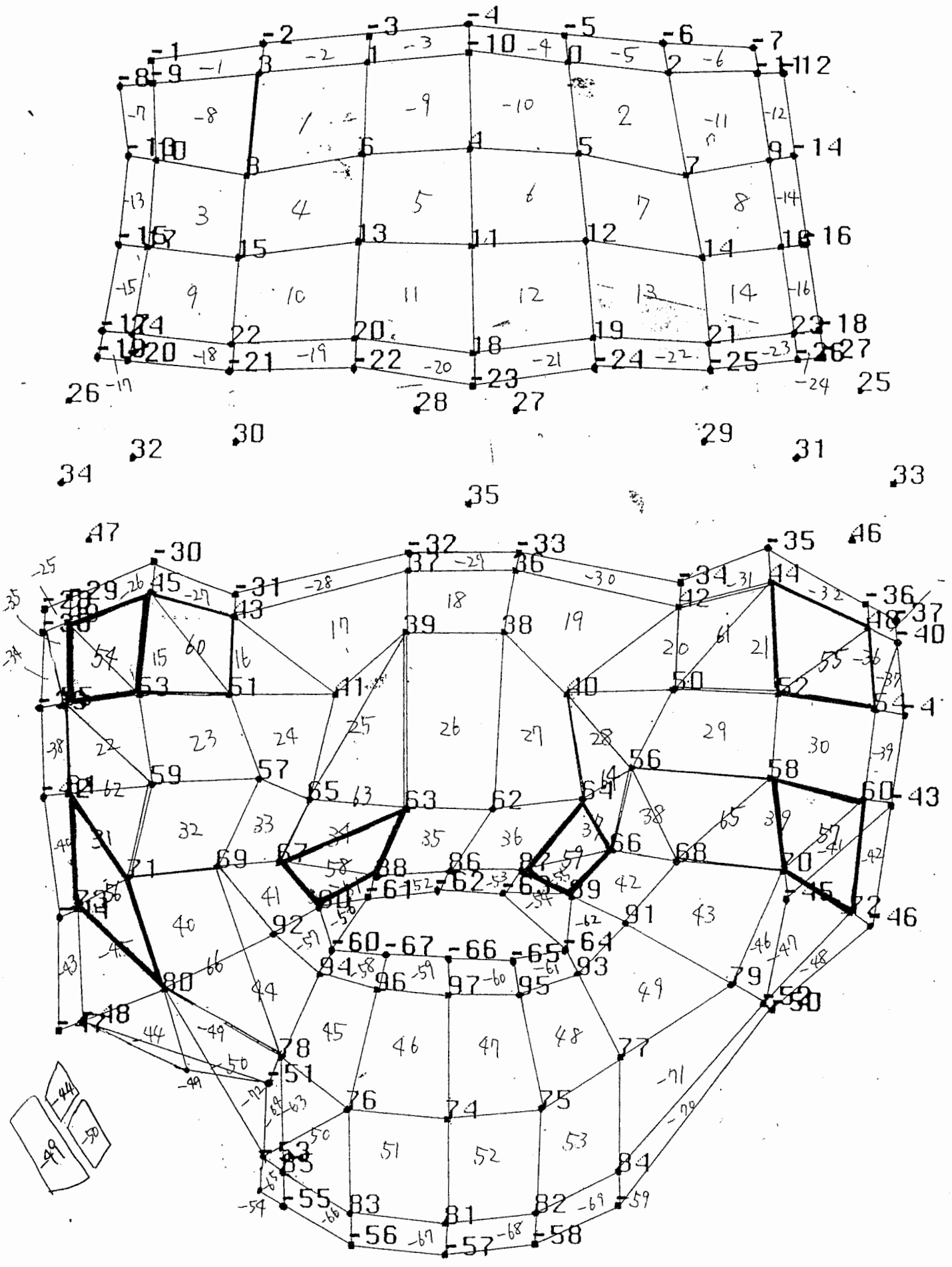
A 2


3 1 0 2
10 8 6 4 5 7 9
17 15 13 11 12 14 16
24 22 20 18 19 21 23
26 28 27 25
32 30 29 31
34 33
35
47 46
45 37 36 44
49 43 39 38 42 48
55 53 51 41 40 50 52 54
61 59 57 65 63 62 64 56 58 60
71 69 67 88 86 87 66 68 70
73 90 89 91 72
92
80 94 96 97 95 93 79
78 77
76 74 75
85 84
83 81 82

A 3



A 4



Appendix B

顔画像を再合成するための関数

B.1 main プログラム

ほとんどの処理は関数ごとにまとめられている. そこで, 行ないたい処理に対応する関数を main 関数に追加し, 実行することで, 目的とする処理が実行できる.

最低限必要な main プログラムを以下に示す.

```
#include        <gl/gl.h>
#include        <gl/device.h>
#include        "std.h"
#include        "cimage.h"
#include        "wavefront.h"

short whitevec[3]= {255, 255, 255};
short cyanvec[3] = {0, 255, 255};
short bluevec[3] = {0, 0, 255};
short greenvec[3] = {0, 255, 0};
short redvec[3] = {255, 0, 0};
short blackvec[3] = {0, 0, 0};

static WaveFront     wf;
static InflPoint     ip;
static MoVector      movect;

main()
{
  char          fname[100];

  int           i;
  char          num[10];
  FILE          *fp;

  foreground();
  prefsize(800, 800);

  winopen( "test");
  RGBmode();
  gconfig();

  ortho(-150.0, 150.0, -150.0, 150.0, -150.0, 150.0);
```

```
  c3s(whitevec);
  clear();

/* ここに行ないたい処理を挿入 */

  printf("end\n");
  gflush();
  sleep(1000);
}
```

B.2 main プログラムに挿入する関数の説明

B.2.1 ReadWaveFront

これは, ワイヤーフレームモデルの各ノードの位置データ (ochi_head_5.obj) を読み込む関数である.

以下を main 関数に挿入する.

```
sprintf( fname , ''/home/komori/atr/data/cyber/ochi\_head\_5.obj'');
ReadWaveFront( fname , \&wf , 10 );
```

B.2.2 ReadControl

これは, マーカの位置データ (control98) を読み込む関数である.

以下を main 関数に挿入する.

```
sprintf( fname , ''/home/komori/atr/data/control/control98'');
ReadControl( fname , &ip);
```

B.2.3 ReadrectangleInfomation

これは, ある四角形がどの 4 点のマーカから成る四角形かを示す情報 (plusrecinf.dat, mrecinf.dat) を読み込む関数である. 以後, mrecinf.dat は使用しない. また, plusrecinf.dat は四角形の番号, 四隅のうちの 1 つめのマーカ番号 (任意), 2 つめのマーカ番号, 3 つめのマーカ番号, 4 つめのマーカ番号の順で入力される. ただし, マーカ番号を入力する順は反時計方向となるようにする.

以下を main 関数に挿入する.

```
ReadrectangleInfomation(&ip);
```

B.2.4 Catwfmgrid

ワイヤーフレームモデルの各ノードの位置をファイル (wfmgrid.dat) に出力する関数である. wfmgrid.dat はノード番号, x 座標値, y座標値, z座標値の順に出力されたものである.

以下を main 関数に挿入する.

```
Catwfmgrid( &wf );
```

B.2.5 WriteWF

ワイヤーフレームモデルの各ノードの位置を window に出力する関数である.

以下を main 関数に挿入する.

```
WriteWF( &wf );
```

B.2.6 Inputzflag

表情変化に伴い変化するワイヤーフレームモデルのノードは顔前面に対してのみであると考えられるため, 変化させるワイヤーフレームモデルのノードは z軸の座標値が正のもののみとした. この関数はワイヤーフレームモデルの各ノードの z軸座標値が正か負かを判断し, その情報を変数に格納するものである.

以下を main 関数に挿入する.

```
InputZflag( &wf );
```

B.2.7 InoutCheck

Inputzflag でノードの z軸座標値が正と判定されたノードに対し, そのノードがどの 4 つのマーカから成る四角形に含まれているかを調べる関数である. 4 つのマーカから成る四角形がオーバーラップしており, ノードが 2 つの四角形に含まれている場合については, 番号の早い四角形に含まれるとした.

以下を main 関数に挿入する.

```
InoutCheck( &wf, &ip);
```

B.2.8 Writecon

各マーカの位置を window に出力する関数である.

以下を main 関数に挿入する.

```
Writecon( &ip );
```

B.2.9 Minuscondeside

B–スプライン曲面近似を用いて, 1 つの曲面パッチを形成するには最低 16 点のマーカが必要である. 顔の外縁部についてはマーカが不足しているため, 仮想マーカを配置した. 仮想マーカの情報 (minuscontrol98.dat) を読み込み, window に仮想マーカを出力する関数である. minuscontrol98.dat は仮想マーカ番号, 図 5.2での P_1, 図 5.2での P_2, 図 5.2でのαの順で入力される.

以下を main 関数に挿入する.

```
Minuscondeside(&ip);
```

B.2.10 ReadPoint16

本研究では B–スプライン曲面近似を用いて, 16 点のマーカから 1 つの曲面パッチを形成した. この関数はその 16 点のマーカの位置情報 (newaropoint16.dat) を読み込むものである. newaropoint16.dat は曲面パッチ番号, ある隅のマーカ番号, その隣のマーカ番号, その隣のマーカ番号という様に直線的に入力され, その直線と平行するように, 残りの 12 点のマーカ番号が 4 つ毎に直線的に入力されている.

以下を main 関数に挿入する.

```
ReadPoint16( &ip );
```

B.2.11 StoreUW

第 1 フレームにおいて, 各ノードの x, y, 座標から曲面上の u, wを求める.

以下を main 関数に挿入する.

```
StoreUW( &wf, &ip);
```

B.2.12 ReadMovement

各フレームでのマーカの移動情報 (movement.new) を読み込む.

以下を main 関数に挿入する.

```
ReadMovement( &movect );
```

B.2.13 AllXYZ

各フレームでのマーカの移動情報から, マーカの位置 (x, y, z) を取得する関数である.

以下を main 関数に挿入する.

```
AllXYZ( &ip, &movect);
```

B.2.14 MinuscondesideNext

各フレームにおいて, 新たな仮想マーカ位置を求める関数である.
以下を main 関数に挿入する.

```
MinuscondesideNext( &movect );
```

B.2.15 MakeTempFile

各フレームにおけるワイヤーフレームモデルの各ノードの位置 (x, y, z) を関数 StoreUW で求めた u, w を用いて推定し, ノードの動き情報として tempfile1.dat に格納する関数である.
以下を main 関数に挿入する.

```
MakeTempFile(&wf, &ip, &movect);
```

B.2.16 Aberration

ノードの位置推定の精度を評価するために, 4 つのマーカから構成される曲面パッチの四隅の位置とマーカの位置の誤差評価を行う関数である. aberration.dat にフレーム番号, 各フレーム毎の平均誤差, 最小誤差, 最大誤差が出力される.
以下を main 関数に挿入する.

```
Aberration(&wf, &ip, &movect);
```

Appendix C

プログラムリスト

```
1
2     #include        <gl/gl.h>
3     #include        <gl/device.h>
4
5     #include        "std.h"
6     #include        "wavefront.h"
7     #include        "cimage.h"
8
9     double CPvalueget();
10
11    /*****************************************************
12      WaveFront Format のデータファイルを読み込む
13    *****************************************************/
14
15    void ReadWaveFront( filename , wf , zoom )
16    char            *filename;
17    WaveFront       *wf;
18    int             zoom;
19    {
20      FILE          *fp;
21      int           i = 0 , j;
22      int           ml1 , ml2 , ml3 , tl1 , tl2 , tl3;
23      float         dx , dy , dz;
24      char          buff[256];
25      char          dum[100] , dum1[100];
26
27      printf( "zoom = %d\n" , zoom );
28
29    /* Wave Front Format のファイルを開く */
30
31      if( NULL == ( fp = fopen( filename , "r" ) ) ){
32        printf( "Cannot open file .. %s ( ReadWaveFront@readwavefront )\n" ,
33                filename );
34        exit( 1 );
35      }
36
37    /* Wave Front Format のヘッダ部分を飛ばす */
38
39      fgets( buff , 256 , fp );
40      while( buff[0] == '#' ) fgets( buff , 256 , fp );
41
42    /* 3次元構造データ (v) を読む */
43
44      while( buff[1] != 't' ){
45        sscanf( buff , "%s %f %f %f" , dum , &dx , &dy , &dz );
46        wf->mgrid[i].x = dx * (float)zoom;
47        wf->mgrid[i].y = dy * (float)zoom;
48        wf->mgrid[i++].z = dz * (float)zoom;
49        fgets( buff , 256 , fp );
50
51    }
52      wf->gridnum = i;
53      printf( "gridnum = %d\n" , wf->gridnum );
54      i = 0;
55
56    /* テクスチャ上の座標を読む */
57
58      while( buff[1] == 't' ){
59        sscanf( buff , "%s %f %f %f" , dum , &dx , &dy , &dz );
60        .
61        wf->tgrid[i].x = dx;
62        wf->tgrid[i].y = dy;
63        wf->tgrid[i++].z = dz;
64        fgets( buff , 256 , fp );
65      }
66      wf->tgridnum = i;
```



```
67      printf( "tgridnum = %d\n" , i );
68      printf("tg2 = %d\n", wf->tgridnum);
69      i = 0;
70
71    /* 情報を読む */
72      while( buff[0] != 'f' ){
73        sscanf( buff , "%s %s" , dum , dum1 );
74        if( strcmp( dum , "usemtl" ) == 0 );
75        if( strcmp( dum , "usemap" ) == 0 ) strcpy( wf->colfname , dum1 );
76        fgets( buff , 256 , fp );
77      }
78
79    /* リンク情報を読む */
80
81      i = 0;
82      GetWFLink( buff , wf , i );
83      i++;
84      while( 0 == feof( fp )){
85        fgets( buff , 256 , fp ) ;
86        GetWFLink( buff , wf  , i );
87        i++;
88      }
89
90      wf->linknum = i;
91
92      fclose( fp );
93    }
94
95    /*****************************************
96      読み込まれた文字列から数値列への変換
97    *****************************************/
98
99    GetWFLink( buff , wf , num )
100   char            *buff;
101   WaveFront       *wf;
102   int             num;
103   {
104     char          tmp1[50] , tmp2[50] , tmp3[50] , dum[50];
105
106     sscanf( buff , "%s %s %s %s" , dum , tmp1 , tmp2 , tmp3 );
107
108     Separate( tmp1 , &(wf->link[num].model.g1) , &(wf->link[num].tex.g1) );
109     Separate( tmp2 , &(wf->link[num].model.g2) , &(wf->link[num].tex.g2) );
110     Separate( tmp3 , &(wf->link[num].model.g3) , &(wf->link[num].tex.g3) );
111   }
112
113   /*********************************************
114     読み込まれた文字列から数値列への変換 (サブ)
115   *********************************************/
116   Separate( tmp , mg , tg )
117   char            *tmp;
118   int             *mg , *tg;
119   {
120     char          mch[100];
121     int           i = 0;
122
123     while( *tmp != '/' ) mch[i++] = *(tmp++);
124
125     mch[i] = '\0';
126     *mg = atoi( mch );
127     *tg = atoi( ++tmp );
128
129   }
130
131   /*****************************************
132     テクスチャ上の座標に対して一番近い点の
```

```
133     番号を返す
134   ****************************************/
135
136   int NearestPoint( wf , x , y )
137   WaveFront       *wf;
138   double          x , y;
139   {
140     double        dis , mindis ;
141     int           min , i;
142
143     min = 0;
144     mindis = sqrt(sqrdb( wf->tgrid[0].x - x ) + sqrdb( wf->tgrid[0].y - y ));
145
146     for( i = 1 ; i < wf->tgridnum ; i++ ){
147       dis = sqrt(sqrdb( wf->tgrid[i].x - x ) + sqrdb( wf->tgrid[i].y - y ));
148       if( mindis > dis ){
149         mindis = dis; min = i;
150       }
151     }
152     return( min );
153   }
154
155   int TextToReal( wf , num )
156   WaveFront               *wf;
157   int                     num;
158   {
159     int           i;
160
161     for( i = 0 ; i < wf->linknum ; i++ ){
162       if( wf->link[i].tex.g1 == (num +1)) return( wf->link[i].model.g1 -1);
163       if( wf->link[i].tex.g2 == (num +1)) return( wf->link[i].model.g2 -1);
164       if( wf->link[i].tex.g3 == (num +1)) return( wf->link[i].model.g3 -1);
165     }
166   }
167
168   /*#ifdef IRIS */
169   void WFwriteOP( wf , siten , sizex , sizey , x , y , dot , col , clsw )
170   WaveFront       *wf;
171   float           sizex , sizey;
172   int             dot , clsw;
173   vector2         siten;
174   long            x , y ;
175   short           col;
176   {
177     int           i , j;
178     float         vert[2];
179
180     char          num[10];
181     short         blackvec[3] = {0, 0, 0};
182
183     if( clsw == CLEAR_ON ){
184       Color( BLACK );
185       clear();
186     }
187
188     pntsize( (short)dot );
189     Color( (Colorindex)col );
190
191     c3s(blackvec);
192     for( i = 0 ; i < wf->tgridnum ; i++ ){
193       vert[0] = (float)wf->tgrid[i].x * sizex - (float)siten.x;
194       vert[1] = (float)wf->tgrid[i].y * sizey - (float)siten.y;
195   /*printf("i = %d x = %f y = %f\n", i, vert[0], vert[1]);
196   */    if( vert[0] > 0.0 && vert[0] < (float)x && vert[1] > 0.0 && vert[1] < (flo
      at)y ){
197         sprintf(num, "%d", i);
```



```
198         cmov2(vert[0], vert[1]);
199         charstr(num);
200         bgnpoint();
201         v2f( vert );
202         endpoint();
203       }
204     }
205   }
206
207   /*********************************************************
208     WaveFront のテクスチャ上の座標を基に画面上に線を引く
209   *********************************************************/
210
211   void WFwrite( wf , siten , sizex , sizey , x , y , dot )
212   WaveFront       *wf;
213   float           sizex , sizey;
214   int             dot;
215   vector2         siten;
216   long            x , y ;
217   {
218     int           i , j;
219     float         vert[3][3];
220
221     vert[0][2] = 0.0 ; vert[1][2] = 0.0 ; vert[2][2] = 0.0 ;
222
223     Color( BLACK );
224     clear();
225
226     for( i = 0 ; i < wf->linknum ; i++ ){
227       Color( WHITE );
228       vert[0][0] = (float)wf->tgrid[wf->link[i].tex.g1-1].x * sizex - (float)siten
      .x;
229       vert[0][1] = (float)wf->tgrid[wf->link[i].tex.g1-1].y * sizey - (float)siten
      .y;
230       vert[1][0] = (float)wf->tgrid[wf->link[i].tex.g2-1].x * sizex - (float)siten
      .x;
231       vert[1][1] = (float)wf->tgrid[wf->link[i].tex.g2-1].y * sizey - (float)siten
      .y;
232       vert[2][0] = (float)wf->tgrid[wf->link[i].tex.g3-1].x * sizex - (float)siten
      .x;
233       vert[2][1] = (float)wf->tgrid[wf->link[i].tex.g3-1].y * sizey - (float)siten
      .y;
234
235       if( vert[0][0] * vert[0][1] > 0 && vert[1][0] * vert[1][1] > 0 &&
236          vert[2][0] * vert[2][1] > 0 ){
237         if( vert[0][0] < x && vert[0][1] < y && vert[1][0] < x &&
238            vert[1][1] < y && vert[2][0] < x && vert[2][1] < y ){
239
240           bgnclosedline();
241           v3f( vert[0] );
242           v3f( vert[1] );
243           v3f( vert[2] );
244           endclosedline();
245
246           if( dot != DOT_OFF ){
247             Color( YELLOW );
248             pntsize( (short)dot );
249             bgnpoint();
250             v2f( vert[0] );
251             v2f( vert[1] );
252             v2f( vert[2] );
253             endpoint();
254           }
255         }
256       }
257     }
```

```
258   }
259
260
261
262   int CPointget( ip , siten , wsizex , wsizey , mval )
263   InflPoint                *ip;
264   vector2                  siten;
265   float                    wsizex , wsizey;
266   short                    mval[];
267   {
268     int           i , j , minnum ;
269     double        dis , mindis;
270     float         x , y;
271
272     minnum = 0;
273     mindis = 10000.0;
274
275     for( i = 0 ; i < ip->pnum ; i++ ){
276       x = (float)ip->tgrid[i].x * wsizex - (float)siten.x;
277       y = (float)ip->tgrid[i].y * wsizey - (float)siten.y;
278
279       dis = sqrdb( (double)( x - (float)mval[0] ) ) + sqrdb( (double)( y -
280               (float)mval[1] ) );
281       dis = sqrt( dis );
282       if( dis < mindis ){
283         mindis = dis;
284         minnum = i;
285       }
286     }
287     return( minnum );
288   }
289
290   Chweight( ig , tpo , num , val , moto )
291   InflGroup                *ig;
292   int                      tpo , num;
293   double          val , moto;
294   {
295     int           i , j , max;
296     double        maxval;
297     if( moto > 0.0 ){
298       for( i = 0 ; i < (ig+tpo)->totcp ; i++ ){
299         if( (ig+tpo)->cpnum[i] == num ){
300           (ig+tpo)->cpweight[i] = val;
301         }
302       }
303     }else{
304       (ig+tpo)->cpnum[(ig+tpo)->totcp] = num;
305       (ig+tpo)->cpweight[(ig+tpo)->totcp] = val;
306       ((ig+tpo)->totcp)++;
307     }
308
309     maxval = 0.0 ; max = (ig+tpo)->near;
310     for( i = 0 ; i < (ig+tpo)->totcp ; i++ ){
311       if( maxval < (ig+tpo)->cpweight[i] ){
312         maxval = (ig+tpo)->cpweight[i];
313         max = (ig+tpo)->cpnum[i];
314       }
315     }
316     (ig+tpo)->near = max;
317
318     printf( "Done.. \n" );
319   }
320
321   double CPvalueget( ig , tpo , num )
322   InflGroup                *ig;
323   int                      tpo , num ;
```



```
324   {
325     int i;
326     for( i = 0 ; i < (ig+tpo)->totcp ; i++ )
327       if( (ig+tpo)->cpnum[i] == num ) return( (ig+tpo)->cpweight[i] );
328     return( -1.0 );
329   }
330
331   int Pointget( wf , siten , sizex , sizey , x , y , mval )
332   WaveFront                *wf;
333   vector2                  siten;
334   float                    sizex , sizey ;
335   long                     x , y;
336   short                    mval[];
337   {
338     int           i , j;
339     float         ipx , ipy;
340
341     ipx = ( (float)mval[0] + (float)siten.x ) / sizex;
342     ipy = ( (float)mval[1] + (float)siten.y ) / sizey;
343   /*
344     printf( "ipx = %f ipy = %f\n" , ipx , ipy );
345   */
346     i = NearestPoint( wf , (double)ipx , (double)ipy );
347
348     printf( "nearest = %d\n" , i );
349     return( i );
350   }
351
352   Pwrite( wf , num ,  siten , sizex , sizey , x , y , dot )
353   WaveFront                *wf;
354   vector2                  siten;
355   float                    sizex , sizey ;
356   long                     x , y;
357   int                      dot , num ;
358   {
359     float                  vert[2];
360
361     vert[0] = (float)( wf->tgrid[num].x ) * sizex - (float)siten.x;
362     vert[1] = (float)( wf->tgrid[num].y ) * sizey - (float)siten.y;
363
364     Color( RED );
365     pntsize( (short)dot );
366     bgnpoint();
367     v2f( vert );
368     endpoint();
369   }
370
371   CPwrite( wf , ip , ig , tpo , siten , wsizex , wsizey , x , y )
372   WaveFront                *wf;
373   InflPoint                *ip;
374   InflGroup                *ig;
375   int                      tpo;
376   vector2                  siten;
377   float                    wsizex , wsizey;
378   long                     x , y;
379   {
380     int                    i , j;
381     float                  vert[2];
382     float                  ipx , ipy ;
383
384     AllCPwrite( ip , siten , wsizex , wsizey , x , y , DOT_ON_HUGE );
385
386     for( i = 0 ; i < (ig+tpo)->totcp ; i++ ){
387       vert[0] = (float)(ip->tgrid[(ig+tpo)->cpnum[i]].x) * wsizex - (float)siten.x
    ;
388       vert[1] = (float)(ip->tgrid[(ig+tpo)->cpnum[i]].y) * wsizey - (float)siten.y
```

```
    ;
389       if( (ig+tpo)->cpnum[i] == (ig+tpo)->near ) Color( BLUE );
390       else                                       Color( CYAN );
391       if( (ig+tpo)->cpweight[i] != 0.0 ){
392         pntsize( DOT_ON_HUGE );
393         bgnpoint();
394         v2f( vert );
395         endpoint();
396       )
397     )
398   }
399
400   CPInfAreaWrite( wf , ip , ig , tpo , siten , wsizex , wsizey , x , y )
401   InflPoint               *ip;
402   InflGroup               *ig;
403   int                     tpo;
404   vector2                 siten;
405   float                   wsizex , wsizey;
406   long                    x , y;
407   WaveFront               *wf;
408   {
409     int                   i , j;
410     float                 vert[2];
411
412     WFwriteOP( wf , siten , wsizex , wsizey , x , y , DOT_ON_MID , YELLOW ,
413                CLEAR_OFF );
414     AllCPwrite( ip , siten , wsizex , wsizey , x , y , DOT_ON_HUGE );
415
416     vert[0] = (float)ip->tgrid[tpo].x * wsizex - (float)siten.x;
417     vert[1] = (float)ip->tgrid[tpo].y * wsizey - (float)siten.y;
418
419     Color( BLUE );
420     pntsize( (short)DOT_ON_HUGE );
421     bgnpoint();
422     v2f( vert );
423     endpoint();
424
425     for( i = 0 ; i < wf->tgridnum ; i++ ){
426       for( j = 0 ; j < (ig+i)->totcp ; j++ ){
427         if( (ig+i)->cpnum[j] == tpo && (ig+i)->cpweight[j] > 0.0 ){
428           Pwrite( wf , i , siten , wsizex , wsizey , x , y , DOT_ON_MID );
429         )
430       )
431     )
432   )
433
434
435
436   /*#endif*/
437
438   /*
439   main()
440   {
441     char          filename[100];
442     WaveFront     wf;
443     int           i;
444
445     strcpy( filename , "/home/tatsu/atr/data/cyber/ochi_head_1.obj" );
446
447     ReadWaveFront( filename , &wf );
448
449     i = NearestPoint( &wf , 0.5 , 0.5 );
450
451     printf( "i = %d x = %f , y = %f\n" , i , wf.tgrid[i].x , wf.tgrid[i].y );
452
453     printf( "gnum = %d , lnum = %d\n" , wf.gridnum , wf.linknum );
```



```
454
455   )
456   */
457
458   void NormalizeIG( ig , gnum )
459   InflGroup       *ig;
460   int             gnum;
461   {
462     int           i , j;
463     double        total;
464
465     for( i = 0 ; i < gnum ; i++ ){
466       total = 0.0;
467       for( j = 0 ; j < (ig+i)->totcp ; j++ )
468         total += (ig+i)->cpweight[j];
469       if( total > 1.0 ){
470         for( j = 0 ; j < (ig+i)->totcp ; j++ )
471           (ig+i)->cpweight[j] /= total;
472       )
473     )
474   )
475   AllCPwrite( ip , siten , wsizex , wsizey , x , y , dot )
476   InflPoint       *ip;
477   vector2         siten;
478   float           wsizex , wsizey;
479   long            x , y ;
480   int             dot;
481   {
482     int           i , j;
483     float         vert[3];
484     float         ipx , ipy;
485
486     short         greenvec[3] = {0, 255, 0};
487     short         redvec[3] = {255, 0, 0};
488     short         bluevec[3] = {0, 0, 255};
489     short         whitevec[3] = {255, 255, 255};
490     short         blackvec[3] = {0, 0, 0};
491     char          num[10];
492
493
494   /*printf("wsizex= %lf wizey = %lf siten.x = %f siten.y %f\n", wsizex, wsizey, (f
      loat)siten.x, (float)siten.y);
495   */
496     for( i = 0 ; i < ip->pnum ; i++ ){
497       ipx = (float)ip->tgrid[i].x * wsizex - (float)siten.x;
498       ipy = (float)ip->tgrid[i].y * wsizey - (float)siten.y;
499         if( ipx > 0 && ipy > 0 && ipx < x && ipy < y ){
500         vert[0] = ipx;
501         vert[1] = ipy;
502         c3s(greenvec);
503         sprintf(num, "%d", i);
504   /*      c3s(cyanvec);
505         circf(vert[0], vert[1], 0.5);
506   */      pntsize( (short)dot );
507         cmov2(vert[0], vert[1]);
508         charstr(num);
509         bgnpoint();
510         v2f( vert );
511         endpoint();
512       )
513     )
514   )
```

```
1     #include    "stdio.h"
2     #include    "cmath.h"
3
4     int SignDouble( val )
5          double val;
6     {
7       if( val < 0 ) return( -1 );
8       if( val > 0 ) return( 1 );
9       return( 0 );
10    }
11
12    double sqrdb( d )
13         double     d;
14    {
15      return( d * d );
16    }
17
```

A15

```
1     #include        "std.h"
2     #include        "wavefront.h"
3
4     #define         THRESH          25.0
5
6
7     /*****************************************************
8       WaveFrontのデータを制御点をもとにグループ分けする
9     *****************************************************/
10
11    DivGroup( wf , ip , ig )
12    WaveFront               *wf;
13    InflPoint               *ip;
14    InflGroup               *ig;
15    {
16      int           i , j , jj , maxnum = 0;
17      double        dis , max ;
18      int           count;
19      double        totalweight;
20
21      for( j = 0 ; j < wf->tgridnum ; j++ ){
22        if( (ig+j)->totcp == 0 ){
23          jj = TextToReal( wf , j );
24          count = 0;
25          max = -2.0;
26          totalweight = 0.0;
27          for( i = 0 ; i < ip->pnum ; i ++ ){
28            dis = sqrdb( ip->mgrid[i].x - wf->mgrid[jj].x ) +
29              sqrdb( ip->mgrid[i].y - wf->mgrid[jj].y ) +
30                sqrdb( ip->mgrid[i].z - wf->mgrid[jj].z );
31            dis = sqrt( dis );
32            if( j == 943 ){
33              printf( "Ip(%f %f %f),Wf(%f %f %f)\n" , ip->mgrid[i].x , ip->mgrid[i].
      y,
34                      ip->mgrid[i].z , wf->mgrid[jj].x , wf->mgrid[jj].y , wf->mgrid[
      jj].z);
35              printf( "dis = %f\n" , dis );
36              printf( "jj =%d\n" , jj );
37            }
38            if ( dis < THRESH ){
39              (ig+j)->cpweight[count] = ( THRESH - dis ) / THRESH;
40              totalweight += (ig+j)->cpweight[count];
41              (ig+j)->cpnum[count] = i;
42              if( max < (ig+j)->cpweight[count] ) {
43                max = (ig+j)->cpweight[count];
44                maxnum = (ig+j)->cpnum[count] ;
45              }
46              count++;
47            }
48          }
49          (ig+j)->totcp = count;
50          (ig+j)->near = maxnum;
51    /*
52          if( totalweight > 1.0 ){
53            totalweight = 1.0 / totalweight;
54            for( i = 0 ; i < count ; i++ )
55              (ig+j)->cpweight[i] *= totalweight;
56          }
57    */
58        }
59      }
60    }
61
62    WriteGroup( filename , gnum , ig )
63    char              *filename;
64    int               gnum;
```



```
65    InflGroup       *ig;
66    {
67      int           i , j , num , maxnum ;
68      FILE          *fp;
69      InflGroup     tmp;
70      double        max;
71
72      if( NULL == ( fp = fopen( filename , "w" ) ) ){
73        printf( "Cannot open file ..%s(WriteGroup@divgroup)\n" , filename );
74        exit( 1 );
75      }
76
77      fprintf( fp , "%d\n" , gnum );
78
79      for( j = 0 ; j < gnum ; j++ ){
80        tmp.totcp = (ig+j)->totcp;
81        num = 0;
82        for( i = 0 ; i < tmp.totcp ; i++ ){
83          if( (ig+j)->cpweight[i] != 0.0 ){
84            tmp.cpnum[num] = (ig+j)->cpnum[i];
85            tmp.cpweight[num] = (ig+j)->cpweight[i];
86            num++;
87          }
88        }
89        tmp.totcp = num;
90        max = 0.0;
91
92        for( i = 0 ; i < num ; i++ ){
93          if( max < tmp.cpweight[i] ){
94            max = tmp.cpweight[i];
95            maxnum = i;
96          }
97        }
98        tmp.near = tmp.cpnum[maxnum];
99
100       fprintf( fp , "%d %d\n" , tmp.totcp , tmp.near );
101       for( i = 0 ; i < tmp.totcp ; i++ )
102         fprintf( fp , "%d %f\n" , tmp.cpnum[i] , tmp.cpweight[i] );
103     }
104
105     fclose( fp );
106   }
107
108   ReadGroup( filename , ig )
109   char              *filename;
110   InflGroup         *ig;
111   {
112     int             i , j , num ;
113     FILE            *fp;
114     int             tot , near , t;
115     float           d;
116
117     if( NULL == ( fp = fopen( filename , "r" ) ) ){
118       printf( "Cannot open file ..%s(ReadGroup@divgroup)\n" , filename );
119       exit( 1 );
120     }
121
122     fscanf( fp , "%d" , &num );
123
124     for( j = 0 ; j < num ; j++ ){
125       fscanf( fp , "%d %d" , &tot , &near );
126       (ig+j)->totcp = tot ;
127       (ig+j)->near = near;
128       for( i = 0 ; i < tot ; i++ ){
129         fscanf( fp , "%d %f" , &t , &d );
130         (ig+j)->cpnum[i] = t;
```

A16

```
131         (ig+j)->cpweight[i] = (double)d;
132       }
133     }
134
135     fclose( fp );
136   }
137
138   ReadControl( filename , ip )
139   char              *filename;
140   InflPoint         *ip;
141   {
142     FILE            *fp;
143     int             i , j , num;
144     float           x , y , z ;
145     char            chartmp[50];
146     int             side , mov;
147
148     if( NULL == ( fp = fopen( filename , "r" ) ) ){
149       printf( "Cannot open file ..%s(ReadControl@divgroup)\n" , filename );
150       exit( 1 );
151     }
152
153     fscanf( fp , "%d" , &num );
154     ip->pnum = num;
155
156     printf("ip->pnum= %d\n", ip->pnum);
157     for( i = 0 ; i < num ; i++ ){
158
159       fscanf( fp , "%s" , chartmp );
160       strcpy( ip->cpname[i] , chartmp );
161
162       fscanf( fp , "%f %f" , &x , &y );
163       ip->tgrid[i].x = (double)x;
164       ip->tgrid[i].y = (double)y;
165
166       fscanf( fp , "%f %f %f" ,&x , &y , &z );
167       ip->mgrid[i].x = (double)x;
168       ip->mgrid[i].y = (double)y;
169       ip->mgrid[i].z = (double)z;
170
171       fscanf( fp , "%d %d" , &side , &mov );
172       ip->side[i] = side;
173       ip->movnum[i] = mov;
174
175     }
176
177     fclose( fp );
178   }
179
180
```

A17

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8     #define         PLUSRECNUM      66
9     #define         MINUSRECNUM     72
10
11    void Readrectangle Infomation(ip)
12    InflPoint       *ip;
13    {
14      FILE          *fp;
15      int           i , j, loop;
16      int           recnumber, side1 , side2 , side3 , side4;
17      char          *fname;
18      int           start, end;
19      int           o;
20
21      for(loop = 0; loop < 2; loop++)
22      {
23        if(loop == 0)
24        {
25          fname = "plusrecinf.dat";
26          start = 1;
27          end = PLUSRECNUM;
28        }
29        else
30        {
31          fname = "mrecinf.dat";
32          start = PLUSRECNUM + 1;
33          end = PLUSRECNUM + MINUSRECNUM;
34        }
35
36        if( NULL == ( fp = fopen( fname , "r" ) ) )
37        {
38          printf( "Cannot open file\n");
39          exit( 1 );
40        }
41        for(i = 0 + start; i <= end; i++)
42        {
43          fscanf( fp , "%d %d %d %d %d" , &recnumber , &side1 , &side2 , &side3, &si
      de4 );
44          ip->rec[i][0] = side1;
45          ip->rec[i][1] = side2;
46          ip->rec[i][2] = side3;
47          ip->rec[i][3] = side4;
48        }
49
50    /*    for(o = 0; o<= end; o++)
51        {
52          printf( " o = %d  rec = %d  %d  %d  %d \n", o, ip->rec[o][0], ip->rec[o][1]
      , ip->rec[o][2], ip->rec[o][3]);
53        }
54    */
55        fclose(fp);
56      }
57    }
```

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8
9     void Catwfmgrid( wf )
10    WaveFront       *wf;
11    {
12      FILE          *fp;
13
14      int           i;
15
16      printf("wf->gridnum = %d\n", wf->gridnum);
17
18      if( NULL == ( fp = fopen( "wfmgrid.dat", "w")))
19      {
20        printf("Cannot open file''wfmgrid.dat''.\n");
21        exit(1);
22      }
23
24      for(i = 0; i < wf->gridnum; i++)
25      {
26        fprintf(fp, "%d  %f  %f  %f\n", i, wf->mgrid[i].x, wf->mgrid[i].y, wf->mgrid
      [i].z);
27      }
28    fclose(fp);
29    }
30
```

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8
9     void WriteWF( wf )
10    WaveFront       *wf;
11    {
12      FILE          *fp;
13
14      int           i;
15      char          num[10];
16
17      for(i = 0; i < wf->gridnum; i++)
18      {
19        if(wf->mgrid[i].z > 0 )
20        {
21          sprintf(num, "%d", i);
22          cmov2(wf->mgrid[i].x, wf->mgrid[i].y);
23          charstr(num);
24          circf(wf->mgrid[i].x, wf->mgrid[i].y, 0.5);
25        }
26      }
27    }
28
```

A20

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8
9     void InputZflag( wf )
10    WaveFront       *wf;
11    {
12      FILE          *fp;
13
14      int           i;
15
16      for(i = 0; i < wf->gridnum; i++)
17      {
18        wf->zflag[i] = -1;
19        if(wf->mgrid[i].z > 0 )
20        {
21          wf->zflag[i] = 1;
22        }
23      }
24    }
```

A21

```
1     #include <gl/gl.h>
2     #include <gl/device.h>
3
4     #include "std.h"
5     #include "wavefront.h"
6     #include "cimage.h"
7     #include "bspline.h"
8
9     void InoutCheck( wf, ip)
10    WaveFront       *wf;
11    InflPoint       *ip;
12    {
13      int     i, j, k;
14      COORD   v[4][XYZ];
15      COORD   p[XYZ];
16
17      FILE    *fp;
18
19      if( NULL == ( fp = fopen( "cprecinform.dat", "w")))
20      {
21        printf("Cannot open file\n");
22        exit(1);
23      }
24
25      printf("wf->gridnum= %d\n", wf->gridnum);
26      for(i = 0; i < wf->gridnum; i++)
27      {
28        wf->insideg[i] = -1;
29        if(wf->zflag[i] > 0 )
30        {
31          p[0] = wf->mgrid[i].x;
32          p[1] = wf->mgrid[i].y;
33          p[2] = 0.0;
34
35          for(j = 1; j <= RECNUM ; j++)
36          {
37            for(k = 0; k<4; k++)
38            {
39    /*        printf("RECNUM=%d\n", RECNUM);
40              printf("rec[%d][%d].x = %d\n", j, k, ip->rec[j][k]);
41    */
42              v[k][0] = ip->mgrid[ip->rec[j][k]].x;
43    /*          printf("v[%d][0] = %f\n", k, v[k][0]);
44    */
45              v[k][1] = ip->mgrid[ip->rec[j][k]].y;
46              v[k][2] = 0.0;
47            }
48
49            if( GtInsideTest( p, v, 4, 2))
50            {
51              wf->insideg[i] = j;
52              break;
53            }
54          }
55        }
56        fprintf(fp, "%8d  %8d\n", i , wf->insideg[i]);
57      }
58      fclose(fp);
59    }
60
```

```
/*
================================================================================
  ファイル名 : inside_test.c

  機能       : 多角形の内外判定 （凸多面体）

  履歴       : 1993年7月2日 ; 作成 ; 浦 真吾
================================================================================
*/

#include <stdio.h>
#include <math.h>
#include <gl.h>
#include "bspline.h"

/*
================================================================================
  関数名 : BOOLEAN GtInsideTest( COORD p[XYZ], COORD V[][XYZ],
                                                    int n, int plane )

  引数   : p      (I)   点座標
           V      (I)   多角形を構成する頂点配列（反時計まわりであること）
           n      (I)   多角形を構成する頂点数
           plane  (I)   内外判定をする平面
                        0 : YZ 平面
                        1 : XZ 平面
                        2 : XY 平面

  戻り値 : 内外判定
             0 : 外部
             1 : 内部

  機能   : 2次元平面にある点が多角形の内部にあるか外部にあるかを判定する。

  履歴   : 1993年7月16日 ; 作成 ; 浦 真吾
================================================================================
*/

BOOLEAN GtInsideTest( COORD p[XYZ], COORD V[][XYZ], int n, int plane )
{
  int       i,j;
  COORD     vec1[XYZ], vec2[XYZ], cross_v1v2[XYZ];

  for( i=0; i<n; i++ ) {

    GtSubVectors( V[(i+1)%n], V[i%n], vec1 );
    GtNormalize( vec1, vec1 );

    GtSubVectors( p, V[i%n], vec2 );
    GtNormalize( vec2, vec2 );

    GtCrossProduct( vec1, vec2, cross_v1v2 );

    if(cross_v1v2[plane] <= 0.0) return( OUTSIDE );

  }

  return( INSIDE );

}
```

A23

```
1     /*
2     ------------------------------------------------------------------------
3       ファイル名 : vector_utility.c
4
5       機能       : ベクトル基本演算
6
7       履歴       : 1993年7月1日 ; 作成 ; 浦 真吾
8     ------------------------------------------------------------------------
9     */
10
11    #include <stdio.h>
12    #include <math.h>
13    #include <gl.h>
14    #include "bspline.h"
15
16    /*
17    ------------------------------------------------------------------------
18      関数名 : COORD GtLengthVector( COORD *v )
19
20      引数   : v    (I)   v(x,y,z)ベクトル
21
22      戻り値 : 長さ
23
24      機能   : ベクトルの長さ（スカラー）
25
26      履歴   : 1993年7月1日 ; 作成 ; 浦 真吾
27    ------------------------------------------------------------------------
28    */
29    COORD GtLengthVector( COORD *v )
30    {
31        return( (COORD)sqrt( v[X]*v[X] + v[Y]*v[Y] + v[Z]*v[Z] ) );
32    }
33
34
35    /*
36    ------------------------------------------------------------------------
37      関数名 : COORD GtDistanceVectors( COORD *v1, COORD *v2 )
38
39      引数   : v1    (I)   v1(x,y,z)座標
40               v2    (I)   v2(x,y,z)座標
41
42      戻り値 : 2点間の距離
43
44      機能   : 2点間の距離
45
46      履歴   : 1993年7月1日 ; 作成 ; 浦 真吾
47    ------------------------------------------------------------------------
48    */
49    COORD GtDistanceVectors( COORD *v1, COORD *v2 )
50    {
51        return( sqrt( (v1[X]-v2[X])*(v1[X]-v2[X]) + (v1[Y]-v2[Y])*(v1[Y]-v2[Y]) +
52                      (v1[Z]-v2[Z])*(v1[Z]-v2[Z]) ) );
53    }
54
55
56    /*
57    ------------------------------------------------------------------------
58      関数名 : void GtAddVectors( COORD *u , COORD *v, COORD *s )
59
60      引数   : u    (I)   u(x,y,z) ベクトル
61               v    (I)   v(x,y,z) ベクトル
62               s    (O)   u + v  ベクトル
63
64      戻り値 : なし
65
66      機能   : ベクトルのたし算
```



```
67
68      履歴   : 1993年7月1日 ; 作成 ; 浦 真吾
69    ------------------------------------------------------------------------
70    */
71    void GtAddVectors( COORD *u , COORD *v, COORD *s )
72    {
73        s[X] = u[X] + v[X];
74        s[Y] = u[Y] + v[Y];
75        s[Z] = u[Z] + v[Z];
76    }
77
78
79    /*
80    ------------------------------------------------------------------------
81      関数名 : void GtSubVectors( COORD *u, COORD *v, COORD *s )
82
83      引数   : u    (I)   u(x,y,z) ベクトル
84               v    (I)   v(x,y,z) ベクトル
85               s    (O)   u - v  ベクトル
86
87      戻り値 : なし
88
89      機能   : ベクトルのひき算
90
91      履歴   : 1993年7月1日 ; 作成 ; 浦 真吾
92    ------------------------------------------------------------------------
93    */
94    void GtSubVectors( COORD *u, COORD *v, COORD *s )
95    {
96        s[X] = u[X] - v[X];
97        s[Y] = u[Y] - v[Y];
98        s[Z] = u[Z] - v[Z];
99    }
100
101
102   /*
103   ------------------------------------------------------------------------
104     関数名 : void GtScalarMult( COORD a, COORD *v, COORD *s )
105
106     引数   : a    (I)   定数
107              v    (I)   v(x,y,z) ベクトル
108              s    (O)   a * v  ベクトル
109
110     戻り値 : なし
111
112     機能   : ベクトルに定数をかける
113
114     履歴   : 1993年7月1日 ; 作成 ; 浦 真吾
115   ------------------------------------------------------------------------
116   */
117   void GtScalarMult( COORD a, COORD *v, COORD *s )
118   {
119       s[X] = v[X] * a;
120       s[Y] = v[Y] * a;
121       s[Z] = v[Z] * a;
122   }
123
124
125   /*
126   ------------------------------------------------------------------------
127     関数名 : void GtCrossProduct( COORD *a, COORD *b, COORD *c )
128
129     引数   : a    (I)   a(x,y,z) ベクトル
130              b    (I)   b(x,y,z) ベクトル
131              c    (O)   a × b ベクトル （外積）
132
```

```
133     戻り値 : なし
134
135     機能   : ベクトルの外積
136
137     履歴   : 1993年7月1日 ; 作成 ; 浦 真吾
138   ------------------------------------------------------------------------
139   */
140   void GtCrossProduct( COORD *a, COORD *b, COORD *c )
141   {
142       c[X] = a[Y]*b[Z] - a[Z]*b[Y];
143       c[Y] = a[Z]*b[X] - a[X]*b[Z];
144       c[Z] = a[X]*b[Y] - a[Y]*b[X];
145   }
146
147
148   /*
149   ------------------------------------------------------------------------
150     関数名 : COORD GtInnerProduct( COORD *a, COORD *b )
151
152     引数   : a    (I)   a(x,y,z) ベクトル
153              b    (I)   b(x,y,z) ベクトル
154
155     戻り値 : a ・ b （内積値）
156
157     機能   : ベクトルの内積
158
159     履歴   : 1993年7月1日 ; 作成 ; 浦 真吾
160   ------------------------------------------------------------------------
161   */
162   COORD GtInnerProduct( COORD *a, COORD *b )
163   {
164       COORD c;
165
166       c = a[X]*b[X] + a[Y]*b[Y] + a[Z]*b[Z];
167
168       return(c);
169
170   }
171
172
173   /*
174   ------------------------------------------------------------------------
175     関数名 : void GtNormalize( COORD *a, COORD *b )
176
177     引数   : a    (I)   a(x,y,z) ベクトル
178              b    (I)   a の正規化ベクトル
179
180     戻り値 : なし
181
182     機能   : ベクトルの正規化
183
184     履歴   : 1993年7月1日 ; 作成 ; 浦 真吾
185   ------------------------------------------------------------------------
186   */
187   void GtNormalize( COORD *a, COORD *b )
188   {
189       COORD  dis;
190
191       dis = sqrt( a[X]*a[X] + a[Y]*a[Y] + a[Z]*a[Z] );
192       .
193       b[X] = a[X]/dis;
194       b[Y] = a[Y]/dis;
195       b[Z] = a[Z]/dis;
196
197   #if 0
198       dis = sqrt( GtInnerProduct(a,a) );
```



```
199
200       GtScalarMult(1.0/dis,a,b);
201   #endif
202
203   }
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
```

A25

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8
9     void Writecon( ip )
10    InflPoint       *ip;
11    {
12      FILE          *fp;
13
14      int           i;
15      char          num[10];
16
17      for(i = 0; i < CONTROLPOINTNUM; i++)
18      {
19        sprintf(num, "%d", i);
20        cmov2(ip->mgrid[i].x, ip->mgrid[i].y);
21        charstr(num);
22        circf(ip->mgrid[i].x, ip->mgrid[i].y, 0.5);
23      }
24    }
25
```

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8
9     void Minuscondeside(ip)
10    InflPoint       *ip;
11    {
12      FILE          *fp;
13      char          *fname;
14      char          num[10];
15
16      int           i;
17
18      int           newconnumber, connumber, storcr, p1, p2, pf1, pt1, pf2, pt2;
19      float         amp;
20      float         x, y;
21      int           npt2, pp2;
22
23      fname = "minuscontrol98.dat";
24
25      if( NULL == ( fp = fopen( fname , "r" ) ) )
26      {
27        printf( "Cannot open file\n");
28        exit( 1 );
29      }
30
31      for(i = -1; i >= -67; i--)
32      {
33        fscanf(fp , "%d %d %d %f", &connumber, &p1, &p2, &amp);
34    /*    printf("connumber = %d p1 = %d p2 = %d amp = %f\n", connumber, p1, p2, amp
      );
35    */
36        StrightLine(ip, p1, p2, amp, &x, &y);
37        newconnumber = (-1) * connumber + 100;
38    /*    printf("newconnumber%d\n", newconnumber);
39    */
40        ip->mgrid[newconnumber].x = x;
41        ip->mgrid[newconnumber].y = y;
42        pp2 = p2;
43        if(p2 < 0)
44        {
45           pp2 = (-1) * p2 + 100;
46        }
47        ip->mgrid[newconnumber].z = ip->mgrid[pp2].z;
48
49        sprintf(num, "%d", connumber);
50        cmov2(x, y);
51        charstr(num);
52        circf(x, y, 0.5);
53      }
54      fclose(fp);
55    }
56
```

A27

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8     #include        "math.h"
9
10    void Line( ip, np1, np2, m, b)
11    InflPoint        *ip;
12    int               np1;
13    int               np2;
14    float             *m;
15    float             *b;
16    {
17      int i;
18
19    /*  printf("p1 = %d p2 = %d\n", p1, p2);
20      printf("p1x = %f p1y= %f\n", ip->mgrid[p1].x , ip->mgrid[p1].y);
21    */
22      *m = (ip->mgrid[np2].y - ip->mgrid[np1].y)/(ip->mgrid[np2].x - ip->mgrid[np1].
      x);
23      *b = ip->mgrid[np1].y - *m * ip->mgrid[np1].x;
24    }
25
26    /* 2点を結ぶ直線上に点を配置する(第1フレーム) */
27    void StrightLine( ip, p1, p2, amp, x, y)
28    InflPoint       *ip;
29    int             p1;
30    int             p2;
31    float           amp;
32    float           *x;
33    float           *y;
34    {
35      int           np1, np2;
36
37      np1 = p1;
38      np2 = p2;
39
40      if(p1 < 0)
41      {
42        np1 = (-1) * p1 + 100;
43      }
44      if(p2 < 0)
45      {
46        np2 = (-1) * p2 + 100;
47      }
48
49      *x = ip->mgrid[np1].x + (1 + amp) * (ip->mgrid[np2].x - ip->mgrid[np1].x);
50      *y = ip->mgrid[np1].y + (1 + amp) * (ip->mgrid[np2].y - ip->mgrid[np1].y);
51    }
52
53    void CrossLine(ip, pf1, pt1, pf2, pt2, x, y)
54    InflPoint       *ip;
55    int              pf1;
56    int              pt1;
57    int              pf2;
58    int              pt2;
59    float           *x;
60    float           *y;
61    {
62      float          ma, mb, ba, bb;
63      int            npf1, npt1, npf2, npt2;
64
65      npf1 = pf1;
```



```
66      npt1 = pt1;
67      npf2 = pf2;
68      npt2 = pt2;
69
70      if(pf1 < 0)
71      {
72        npf1 = (-1)  * pf1 + 100;
73      }
74      if(pt1 < 0)
75      {
76        npt1 = (-1)  * pt1 + 100;
77      }
78      if(pf2 < 0)
79      {
80        npf2 = (-1)  * pf2 + 100;
81      }
82      if(pt2 < 0)
83      {
84        npt2 = (-1)  * pt2 + 100;
85      }
86
87      Line(ip, npf1, npt1, &ma, &ba);
88      Line(ip, npf2, npt2, &mb, &bb);
89      *x = (bb-ba)/(ma-mb);
90      *y = mb * *x + bb;
91    }
92
```

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8     #include        "bspline.h"
9
10    void ReadPoint16(ip)
11    InflPoint       *ip;
12    {
13      int           i, j, onepoint, recnum;
14
15      FILE          *fp;
16      char          *fname;
17
18      fname = "newaropoint16.dat";
19
20      if( NULL == ( fp = fopen( fname , "r" ) ) )
21      {
22        printf( "Cannot open file\n");
23        exit( 1 );
24      }
25
26      for(i = 1; i < RECNUM; i++)
27      {
28        fscanf(fp, "%d", &recnum);
29        for(j = 0; j < 16; j++)
30        {
31          fscanf(fp, "%d", &onepoint);
32          ip->numpoint[i][j] = onepoint;
33        }
34      }
35      fclose(fp);
36    }
```

A29

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8     #include        "bspline.h"
9
10
11    void StoreUW( wf, ip )
12    WaveFront       *wf;
13    InflPoint       *ip;
14    {
15      static  Matrix  p[3], q[3];
16      COORD           ans[3];
17      COORD           u, w;
18      int             i,l, k, j;
19      float           point[3];
20      static  float   test[4][3];
21      int             st;
22
23    short whitevec[3]= {255, 255, 255};
24    short cyanvec[3] = {0, 255, 255};
25    short bluevec[3] = {0, 0, 255};
26    short greenvec[3] = {0, 255, 0};
27    short redvec[3] = {255, 0, 0};
28    short blackvec[3] = {0, 0, 0};
29
30      for(i = 0; i < wf->gridnum; i++)
31      {
32        wf->uwbox[i].ubox = -999.0;
33        wf->uwbox[i].wbox = -999.0;
34        if(wf->insideg[i] != -1)
35        {
36          u = -1.0;
37          w = -1.0;
38          Point16xyz( wf, ip, wf->insideg[i], p);
39          printf("insedeg[%d] = %d\n", i, wf->insideg[i]);
40    /*      for(l = 0; l < 3; l++)
41          {
42            for( j = 0; j < 4; j++)
43            {
44              for( k = 0; k < 4; k++)
45              {
46                printf("%10.5f", p[l][j][k]);
47              }
48            }
49            printf("\n");
50          }
51
52          c3s(cyanvec);
53          for(j = 0; j < 4; j++)
54          {
55            for( k = 0; k < 4; k++)
56            {
57              pushmatrix();
58              translate(p[0][j][k], p[1][j][k], p[2][j][k]);
59              circf(0.0, 0.0, 0.5);
60              popmatrix();
61            }
62          }
63    */
64          point[0] = wf->mgrid[i].x;
65          point[1] = wf->mgrid[i].y;
66          point[2] = wf->mgrid[i].z;
```



```
67    /*      printf("wf->mgrid[%d] = %f  %f  %f\n", i, wf->mgrid[i].x, wf->mgrid[i].y
      , wf->mgrid[i].z);
68          printf("point@storeuw= %f  %f  %f\n", point[0], point[1], point[2]);
69    */
70          c3s(bluevec);
71
72          for(j = 0; j < 3; j++)
73          {
74            test[0][j] = p[j][1][1];
75            test[1][j] = p[j][1][2];
76            test[2][j] = p[j][2][2];
77            test[3][j] = p[j][2][1];
78          }
79          for(j = 0; j < 4; j++)
80          {
81    /*        printf("%10.3f %10.3f %10.3f\n", test[j][0], test[j][1], test[j][2]);
82    */
83            pushmatrix();
84            translate(test[j][0], test[j][1], test[j][2]);
85            circf(0.0, 0.0, 0.5);
86            popmatrix();
87          }
88
89    /*      c3s(blackvec);
90          pushmatrix();
91          translate(point[0], point[1], point[2]);
92          circf(0.0, 0.0, 0.5);
93          popmatrix();
94
95          st = newGtInsideTest( point, test, 4, 2);
96          if( st == OUTSIDE ) {
97            printf("out@storeuw(controlpoint4)\n");
98          } else
99          {
100           printf("in@storeuw(controlpoint4)\n");
101         }
102   */
103         XyzToUw(p, q, point, ans, &u, &w);
104         wf->uwbox[i].ubox = u;
105         wf->uwbox[i].wbox = w;
106         printf("i = %d u = %f w = %f\n", i, u, w);
107       }
108     }
109   }
110
```

```
1     /* 指定された16個のマーカ番号のx,y,z座標値を求める(第1フレーム) */
2
3     #include        <gl/gl.h>
4     #include        <gl/device.h>
5
6     #include        "std.h"
7     #include        "wavefront.h"
8     #include        "cimage.h"
9     #include        "string.h"
10    #include        "bspline.h"
11
12    void Point16xyz(wf, ip, onerec, p)
13    WaveFront       *wf;
14    InflPoint       *ip;
15    int             onerec;
16    Matrix          p[];
17    {
18      int           l, pointor, j, k, onepoint;
19
20      pointor = 0;
21      for(j = 0; j < 4; j++)
22      {
23        for(k = 0; k < 4; k++)
24        {
25          onepoint = ip->numpoint[onerec][pointor];
26          if(onepoint < 0)
27          {
28            onepoint = (-1) * onepoint + 100;
29          }
30          p[0][j][k] = ip->mgrid[onepoint].x;
31          p[1][j][k] = ip->mgrid[onepoint].y;
32          p[2][j][k] = ip->mgrid[onepoint].z;
33          pointor = pointor++;
34        }
35      }
36    }
```

A31

```
1     #include <stdio.h>
2     #include <math.h>
3     #include <gl.h>
4     #include "bspline.h"
5
6     /* 16点のx,y,zから発生する曲面上のu,wを求める */
7     void XyzToUw( MATRIX p[XYZ], MATRIX q[XYZ], COORD point[XYZ], COORD result[XYZ],
       COORD *u, COORD *w)
8     {
9             int   i, j, k;
10            int   st;
11            COORD3D p2[4][4], q2[4][4];
12
13            float   test[4][3];
14
15    short whitevec[3]= {255, 255, 255};
16    short cyanvec[3] = {0, 255, 255};
17    short bluevec[3] = {0, 0, 255};
18    short greenvec[3] = {0, 255, 0};
19    short redvec[3] = {255, 0, 0};
20    short blackvec[3] = {0, 0, 0};
21
22            for(i=0;i<4;i++) {
23              for(j=0;j<4;j++) {
24                for(k=0;k<3;k++) {
25                  p2[i][j][k] = p[k][i][j];
26                }
27    /*            c3s(greenvec);
28                pushmatrix();
29                translate(p2[i][j][0], p2[i][j][1], p2[i][j][2]);
30                circf(0.0, 0.0, 0.5);
31                popmatrix();
32    */
33              }
34            }
35
36            GtBsplineSurfaceControlPoint((COORD3D *)p2, (int)4, (int)4, (COORD3D *)q
      2);
37
38            for(i=0;i<4;i++) {
39              for(j=0;j<4;j++) {
40                for(k=0;k<3;k++) {
41                  q[k][i][j] = q2[i][j][k];
42                }
43
44    /*            c3s(bluevec);
45                pushmatrix();
46                translate(q2[i][j][0], q2[i][j][1], q2[i][j][2]);
47                circf(0.0, 0.0, 0.5);
48                popmatrix();
49    */
50              }
51            }
52
53            printf("point= %f  %f  %f\n", point[0], point[1], point[2]);
54            c3s(redvec);
55            pushmatrix();
56            translate(point[0], point[1], point[2]);
57            circf(0.0, 0.0, 0.5);
58            popmatrix();
59
60            st = newGtBsplineSurface2( q, point, result, u, w);
61
62            if(st == FALSE) {
63                printf("outside error@newxyztouw(side4)\n");
64            }
```



```
65
66    }
```

```
/*
-------------------------------------------------------------------------
  ファイル名 : matrix_utility.c

  機能       : マトリックス基本演算

  履歴       : 1993年9月22日 ; 作成 ; 浦 真吾
-------------------------------------------------------------------------
*/


#include <stdio.h>
#include <math.h>
#include <gl.h>
#include "bspline.h"

/*
-------------------------------------------------------------------------
  関数名 : void GtInverMatrix( MATRIX m1, MATRIX m2 )

  引数   : m1    (I)  マトリックス
         : m2    (O)  逆行列 (Inversion)

  戻り値 : なし

  機能   : ４×４の行列（ｍ１）の逆行列（ｍ２）を掃きだし法により計算する。

  履歴   : 1993年9月22日 ; 作成 ; 浦 真吾
-------------------------------------------------------------------------
*/

void GtInverMatrix( MATRIX m1, MATRIX m2 )
{
    int     i, j, k;
    double  t, u;

    GtCopyMatrix( m2, m1 );

    for( i=0; i<4; i++ ) {
        t = m2[i][i];
        for( j=0; j<4; j++ )  m2[i][j] /= t;

        m2[i][i] = 1 / t;
        for( j=0; j<4; j++ ) {
            if( i != j ) {
                u = m2[j][i];
                for( k=0; k<4; k++ ) {
                    if( i != k ) m2[j][k] -= m2[i][k] * u;
                    else         m2[j][k] = -u / t;
                }
            }
        }
    }
}

/*
-------------------------------------------------------------------------
  関数名 : void GtCopyMatrix( MATRIX m1, MATRIX m2 )

  引数   : m1    (I)  マトリックス
         : m2    (O)  m1 のコピーマトリックス

  戻り値 : なし

  機能   : ４×４の行列（ｍ１）を（ｍ２）にコピーする。

```



```
  履歴   : 1993年9月22日 ; 作成 ; 浦 真吾
-------------------------------------------------------------------------
*/
void GtCopyMatrix( MATRIX m1, MATRIX m2 )
{
    m2[0][0] = m1[0][0];
    m2[0][1] = m1[0][1];
    m2[0][2] = m1[0][2];
    m2[0][3] = m1[0][3];
    m2[1][0] = m1[1][0];
    m2[1][1] = m1[1][1];
    m2[1][2] = m1[1][2];
    m2[1][3] = m1[1][3];
    m2[2][0] = m1[2][0];
    m2[2][1] = m1[2][1];
    m2[2][2] = m1[2][2];
    m2[2][3] = m1[2][3];
    m2[3][0] = m1[3][0];
    m2[3][1] = m1[3][1];
    m2[3][2] = m1[3][2];
    m2[3][3] = m1[3][3];
}

/*
-------------------------------------------------------------------------
  関数名 : void GtMultMatrix1( COORD a[XYZW], MATRIX b, COORD c[XYZW] )

  引数   : a    (I)  XYZW 座標
         : b    (I)  マトリックス ４行４列
         : c    (O)  a × b の計算結果の座標

  戻り値 : なし

  機能   : 座標値 (a) と ４×４の行列 (b) のかけ算

  履歴   : 1993年9月22日 ; 作成 ; 浦 真吾
-------------------------------------------------------------------------
*/
void GtMultMatrix1( COORD a[4], MATRIX b, COORD c[4] )
{
        int     m,n;
        COORD   w[4];

        for(n=0;n<4;n++){
                w[n] = 0.0;
                for(m=0;m<4;m++) w[n] += a[m] * b[m][n];
        }

        for(n=0;n<4;n++) c[n] = w[n];
}

/*
-------------------------------------------------------------------------
  関数名 : void GtMultMatrix3( Matrix a, COORD b[XYZW], COORD c[XYZW] )

  引数   : a    (I)  マトリックス ４行４列
         : b    (I)  XYZW 座標
         : c    (O)  a × b の計算結果の座標

  戻り値 : なし

  機能   : ４×４の行列 (a) と座標値 (b) のかけ算

  履歴   : 1993年9月22日 ; 作成 ; 浦 真吾
-------------------------------------------------------------------------
*/
```

```
133  void GtMultMatrix3( Matrix a, COORD b[4], COORD c[4] )
134  {
135          int     m,n;
136          COORD   w[4];
137
138          for(n=0;n<4;n++){
139                  w[n] = 0.0;
140                  for(m=0;m<4;m++) w[n] += a[n][m] * b[m];
141          }
142
143          for(n=0;n<4;n++) c[n] = w[n];
144  }
145
146
147  /*
148  ------------------------------------------------------------------------
149    関数名 : void GtMultMatrix2( MATRIX a, MATRIX b, MATRIX c )
150
151    引数   : a    (I)   マトリックス 4行4列
152           : b    (I)   マトリックス 4行4列
153           : c    (O)   a × b の計算結果の マトリックス 4行4列
154
155    戻り値 : なし
156
157    機能   : 4×4の行列 (a) と 4×4の行列 (b) のかけ算
158
159    履歴   : 1993年9月22日 ; 作成 ; 浦 真吾
160  ------------------------------------------------------------------------
161  */
162
163  void GtMultMatrix2( MATRIX a, MATRIX b, MATRIX c )
164  {
165          int     i,j,k;
166          MATRIX  w;
167
168          for(i=0;i<4;i++){
169                  for(j=0;j<4;j++) {
170                          w[i][j] = 0.0;
171                          for(k=0;k<4;k++)
172                                  w[i][j] += a[i][k] * b[k][j];
173                  }
174          }
175
176          for(i=0;i<4;i++) {
177            for(j=0;j<4;j++) {
178              c[i][j] = w[i][j];
179            }
180          }
181
182  }
183
184  /*
185  ------------------------------------------------------------------------
186    関数名 : void GtTransMatrix( MATRIX m1, MATRIX m2 )
187
188    引数   : m1    (I)   マトリックス 4行4列
189           : m2    (I)   m1 の転置マトリックス 4行4列
190
191    戻り値 : なし
192
193    機能   : 4×4の行列の転置
194
195    履歴   : 1993年9月22日 ; 作成 ; 浦 真吾
196  ------------------------------------------------------------------------
197  */
198
```



```
199  void GtTransMatrix( MATRIX m1, MATRIX m2 )
200  {
201      m2[0][0] = m1[0][0];
202      m2[0][1] = m1[1][0];
203      m2[0][2] = m1[2][0];
204      m2[0][3] = m1[3][0];
205      m2[1][0] = m1[0][1];
206      m2[1][1] = m1[1][1];
207      m2[1][2] = m1[2][1];
208      m2[1][3] = m1[3][1];
209      m2[2][0] = m1[0][2];
210      m2[2][1] = m1[1][2];
211      m2[2][2] = m1[2][2];
212      m2[2][3] = m1[3][2];
213      m2[3][0] = m1[0][3];
214      m2[3][1] = m1[1][3];
215      m2[3][2] = m1[2][3];
216      m2[3][3] = m1[3][3];
217  }
218
219  /*
220  ------------------------------------------------------------------------
221    関数名 : void GtPrintMatrix( MATRIX m1 )
222
223    引数   : m1    (I)   マトリックス 4行4列
224
225    戻り値 : なし
226
227    機能   : 4×4の行列の表示
228
229    履歴   : 1993年9月22日 ; 作成 ; 浦 真吾
230  ------------------------------------------------------------------------
231  */
232
233  void GtPrintMatrix( MATRIX m1 )
234  {
235    int  i,j;
236
237    for(i=0;i<4;i++) {
238      for(j=0;j<4;j++) {
239        printf("%7.5f ",m1[i][j]);
240      }
241      printf("\n");
242    }
243
244  }
245
246
247
248
249
250
251
252
253
254
255
256
257
258
```

```
/*
 ------------------------------------------------------------------------
   ファイル名 : bspline_control_point.c

   機能       : 補間点を通る Bspline の制御点を逆変換により計算する。
                (Gauss-Seidel法)

   履歴       : 1993年7月29日 ; 作成 ; 浦 真吾 ・ (参考) 出来雅江
 ------------------------------------------------------------------------
 */

#include <stdio.h>
#include <math.h>
#include <gl.h>
#include "bspline.h"

#define PRECISION       0.000001

/*
 ------------------------------------------------------------------------
   関数名 : viod GtBsplineControlPoint( COORD p[][XYZ], COORD q[][XYZ], int n )

   引数   : p    (I)   補間点
            q    (O)   逆変換により計算した制御点
            n    (I)   補間点の個数

   戻り値 : なし

   機能   : 補間点を通る Bspline の制御点を逆変換により計算する。
            (Gauss-Seidel法)

   履歴   : 1993年7月29日 ; 作成 ; 浦 真吾
 ------------------------------------------------------------------------
*/

void GtBsplineControlPoint( COORD3D *p, COORD3D *q, int n )
{
        int   i,nn;
        COORD thresh = PRECISION;
        COORD th,max_th;

        nn = n - 1;

        for(i=1;i<nn;i++) {
          q[i][X] = p[i][X];
          q[i][Y] = p[i][Y];
          q[i][Z] = p[i][Z];
        }
        q[0][X] = q[1][X];
        q[0][Y] = q[1][Y];
        q[0][Z] = q[1][Z];
        q[nn][X] = q[nn-1][X];
        q[nn][Y] = q[nn-1][Y];
        q[nn][Z] = q[nn-1][Z];

                                        /* 制御点 x 座標の計算 */
        do{
                max_th = 0.0;
                for(i=1;i<nn;i++){
                        th = p[i][X] - q[i][X] + (p[i][X] - (q[i-1][X] + q[i+1][
X]) / 2.0) / 2.0;
                        q[i][X] += th;
                        if(th > max_th) max_th = th;
                }
                q[0][X] = q[1][X];
                q[nn][X] = q[nn-1][X];
        }while(max_th > thresh);

                                        /* 制御点 Y 座標の計算 */
        do{
                max_th = 0.0;
                for(i=1;i<nn;i++){
                        th = p[i][Y] - q[i][Y] + (p[i][Y] - (q[i-1][Y] + q[i+1][
Y]) / 2.0) / 2.0;
                        q[i][Y] += th;
                        if(th > max_th) max_th = th;
                }
                q[0][Y] = q[1][Y];
                q[nn][Y] = q[nn-1][Y];
        }while(max_th > thresh);

                                        /* 制御点 z 座標の計算 */
        do{
                max_th = 0.0;
                for(i=1;i<nn;i++){
                        th = p[i][Z] - q[i][Z] + (p[i][Z] - (q[i-1][Z] + q[i+1][
Z]) / 2.0) / 2.0;
                        q[i][Z] += th;
                        if(th > max_th) max_th = th;

                }
                q[0][Z] = q[1][Z];
                q[nn][Z] = q[nn-1][Z];
        }while(max_th > thresh);
}
```

```
/*
================================================================================
  ファイル名 : bspline_surface_control_point.c

  機能       : Bspline 曲面の逆変換

  履歴       : 1994年1月7日 ; 作成 ; 浦 真吾
================================================================================
*/

#include <stdio.h>
#include <math.h>
#include <gl.h>
#include "bspline.h"

/*
================================================================================
  関数名 : void GtBsplineSurfaceControlPoint( COORD3D *p, int u,
                                                  int w, COORD3D *q )
  引数   : p   (I)   補間点座標 [W][U][XYZ]
           u   (I)   u 方向データ数
           w   (I)   w 方向データ数
           q   (O)   逆変換により計算された制御点座標 [W][U][XYZ]

  戻り値 : なし

  機能   : 補間点を通るBspline曲面の制御点を逆変換により計算する。
           (Gauss-Seidel法)

           参考文献
             形状処理工学[II] 山口富士夫著 日刊工業新聞社発行 ( P99 )


  履歴   : 1994年1月7日 ; 作成 ; 浦 真吾
================================================================================
*/

void GtBsplineSurfaceControlPoint( COORD3D *p, int u, int w, COORD3D *q )

{
  COORD3D *wq, *wp;
  int     i,j,k,size;

                                /* ワーク領域の確保 */
  if(u > w) {
    size = u;
  }
  else {
    size = w;
  }

  wq = (COORD3D *)malloc(sizeof(COORD3D)*size);
  wp = (COORD3D *)malloc(sizeof(COORD3D)*size);


                                /* U方向の逆変換 */
  for( i=0; i<w; i++ ) {
    for(j=0; j<u; j++) {
      for(k=0; k<XYZ; k++) {
        wp[j][k] = p[j+i*u][k];
      }
    }

    GtBsplineControlPoint( wp, wq, u );

    for(j=0; j<u; j++) {
      for(k=0;k<XYZ; k++) {
        q[j+i*u][k] = wq[j][k];
      }
    }
  }

                                /* W方向の逆変換 */
  for( i=0; i<u; i++ ){
    for(j=0; j<w; j++) {
      for(k=0; k<XYZ; k++) {
        wp[j][k] = q[j*u+i][k];
      }
    }

    GtBsplineControlPoint( wp, wq, w );

    for(j=0; j<w; j++) {
      for(k=0;k<XYZ; k++) {
          q[j*u+i][k] = wq[j][k];
      }
    }
  }

                                /* ワーク領域の解放 */
  free(wp);
  free(wq);

}
```

```
1     /*
2     ************************************************************************
3       ファイル名 : bspline_surface.c
4
5       機能       : Bspline 曲面の基本式
6
7       履歴       : 1993年10月21日 ; 作成 ; 浦 真吾
8     ************************************************************************
9     */
10
11    #include <stdio.h>
12    #include <math.h>
13    #include <gl.h>
14    #include "bspline.h"
15
16    /*
17    ************************************************************************
18      関数名 : void GtBsplineSurface( MATRIX q[XYZ], COORD u, COORD w,
19                                                          COORD result[XYZ] )
20      引数   : q      (I)   制御点 ( 4 × 4 )
21               u      (I)   u 方向パラメータ
22               w      (I)   w 方向パラメータ
23               result (I/O) パラメータ u, v の曲面上の座標  (x,y,z)
24
25      戻り値 : なし
26
27      機能   : Bspline 曲面の基本式
28               曲面を生成する制御点と曲面の縦横パラメータ u,w から
29               曲面上の 3 次元座標値を計算する。
30
31                 Pi,j(u,w) = U Mr Br Mrt Wt
32                     U  = [ u^3, u^2, u, 1 ]
33                     W  = [ w^3, w^2, w, 1 ]  ※ Wt は W の転置マトリックス
34                     Mr = Bspline の基底ベクトル
35                          (
36                             ( -1.0/6.0,      0.5,    -0.5, 1.0/6.0 ),
37                             (      0.5,     -1.0,     0.5,     0.0 ),
38                             (     -0.5,      0.0,     0.5,     0.0 ),
39                             (  1.0/6.0,  2.0/3.0, 1.0/6.0,     0.0 ),
40                          );
41                          ※ Mrt は Mr の転置マトリックス
42                     Br = Bspline の制御点
43                          (
44                             ( Qi-1,j-1  Qi-1,j  Qi-1,j+1  Qi-1,j+2 )
45                             ( Qi,j-1    Qi,j    Qi,j+1    Qi,j+2 )
46                             ( Qi+1,j-1  Qi+1,j  Qi+1,j+1  Qi+1,j+2 )
47                             ( Qi-2,j-1  Qi-2,j  Qi-2,j+1  Qi-2,j+2 )
48                          )
49
50               参考文献
51                 形状処理工学[II] 山口富士夫著 日刊工業新聞社発行 ( P86 )
52
53
54      履歴   : 1993年10月21日 ; 作成 ; 浦 真吾
55    ************************************************************************
56    */
57                                    /* Bspline の基底ベクトル */
58    static MATRIX Mr = {
59            { -1.0/6.0, 0.5, -0.5, 1.0/6.0 },
60            { 0.5, -1.0, 0.5, 0.0 },
61            { -0.5, 0.0, 0.5, 0.0 },
62            { 1.0/6.0, 2.0/3.0, 1.0/6.0, 0.0 },
63    };
64
65    void GtBsplineSurface( MATRIX q[XYZ], COORD u, COORD w, COORD result[XYZ] )
66    {
```



```
67            COORD   Uu[4],Ww[4];
68            Matrix  Mrt;
69            COORD   work[4], work2[4];
70            int     i;
71            MATRIX  Br[XYZ];
72
73                                    /* U = [ u^3, u^2, u, 1 ] マトリックスの生成 */
74            Uu[0] = u*u*u;
75            Uu[1] = u*u;
76            Uu[2] = u;
77            Uu[3] = 1.0;
78                                    /* W = [ w^3, w^2, w, 1 ] マトリックスの生成 */
79            Ww[0] = w*w*w;
80            Ww[1] = w*w;
81            Ww[2] = w;
82            Ww[3] = 1.0;
83                                    /* Mr の転置マトリックス Mrt の作成 */
84            GtTransMatrix( Mr, Mrt );
85
86            for( i=0; i<XYZ; i++ ){
87
88                                    /* Mrt × Wt */
89                GtMultMatrix3( Mrt, Ww, work);
90
91                                    /* Brの作成 */
92                GtTransMatrix( q[i], Br[i] );
93
94                                    /* Br × MrtWt */
95                GtMultMatrix3( Br[i], work, work2);
96
97                                    /* Mr × BrMrtWt */
98                GtMultMatrix3( Mr, work2, work2);
99
100                                   /* U × MrBrMrtWt */
101               result[i] = Uu[X]*work2[X] + Uu[Y]*work2[Y] + Uu[Z]*work2[Z]
102                           + Uu[W]*work2[W];
103           }
104
105   }
106
107   /*
108   ************************************************************************
109     関数名 : BOOLEAN GtBsplineSurface2( MATRIX q[XYZ], COORD point[XYZ],
110                                                          COORD result[XYZ] )
111     引数   : q      (I)   制御点 ( 4 × 4 )
112              point  (I)    (x,y) 座標
113              result (O)   曲面上の座標  (x,y,z)
114
115     戻り値 : なし
116
117     機能   : Bspline 曲面の逆計算
118              曲面上の point (x,y) 座標から曲面上の (x,y,z) 座標を計算する。
119
120              参考文献
121                形状処理工学[II] 山口富士夫著 日刊工業新聞社発行 ( P86 )
122
123
124     履歴   : 1993年10月21日 ; 作成 ; 浦 真吾
125   ************************************************************************
126   */
127
128   BOOLEAN newGtBsplineSurface2( MATRIX q[XYZ], COORD point[XYZ], COORD result[XYZ]
      , COORD *u, COORD *w)
129   {
130
131           COORD   devid,devidu, devidw;
```

```
132         COORD   left,right,bottom,top;
133         COORD   devpoint[9][XYZ];
134         COORD   devplane[4][XYZ];
135         COORD   eps = 1.0e-6;
136         int     inout;
137         int     i,j;
138         static  int     pn[4][4] = {    { 4,1,5,8 },
139                                         { 8,5,2,6 },
140                                         { 7,8,6,3 },
141                                         { 0,4,8,7 } };
142
143 /*  曲面を以下の図のように分割し、これを繰り返すことで収束させる。
144
145                         bottom
146           devpoint[0]     devpoint[7]       devpoint[3]
147                 *-------------*---------------*
148                 |             |               |
149                 |             |               |
150                 | devplane[3]     devplane[2] |
151                 |             |               |
152                 |             |               |
153                 |             |               |
154           devpoint[4]     devpoint[8]         |devpoint[6]
155                 * - - - - - - *- - - - - - - - *
156                 |             |               |
157                 |             |               |
158                 |             |               |
159                 | devplane[0]     devplane[1] |
160                 |             |               |
161                 |             |               |
162                 |             |               |
163                 |             |               |
164                 *-------------*---------------*
165           devpoint[1]     devpoint[5]       devpoint[2]
166           left              top               right
167 */
168
169                             /* 曲面のパラメータに初期値を設定する */
170         right = top = 1.0;
171         left = bottom = 0.0;
172
173                             /* 初期パラメータによる曲面の３次元座標 */
174         GtBsplineSurface( q, left, bottom , devpoint[0] );
175         GtBsplineSurface( q, right, bottom , devpoint[3] );
176         GtBsplineSurface( q, left, top , devpoint[1] );
177         GtBsplineSurface( q, right, top , devpoint[2] );
178
179
180         for(i=0;i<3;i++) devplane[0][i] = devpoint[0][i];
181         for(i=0;i<3;i++) devplane[1][i] = devpoint[1][i];
182         for(i=0;i<3;i++) devplane[2][i] = devpoint[2][i];
183         for(i=0;i<3;i++) devplane[3][i] = devpoint[3][i];
184
185                             /* point  devpoint の内外判定 */
186         inout = newGtInsideTest( point, devplane, (int)4, (int)2 );
187         if( inout == OUTSIDE ) {
188             return(FALSE);
189         }
190
191                             /* 初期分割値 */
192         devid = devidu = devidw = 0.5;
193
194                             /* 収束計算 */
195         while( devid > eps ){
196
197                             /* 分割点座標の計算 */
```



```
198             GtBsplineSurface( q, left,   devidw, devpoint[4] );
199             GtBsplineSurface( q, devidu, top,    devpoint[5] );
200             GtBsplineSurface( q, right,  devidw, devpoint[6] );
201             GtBsplineSurface( q, devidu, bottom, devpoint[7] );
202             GtBsplineSurface( q, devidu, devidw, devpoint[8] );
203
204                 /* point 座標が devplane[0～3] のどこに位置するかテストする */
205             for(j=0;j<4;j++) {
206                 for(i=0;i<3;i++) devplane[0][i] = devpoint[pn[j][0]][i];
207                 for(i=0;i<3;i++) devplane[1][i] = devpoint[pn[j][1]][i];
208                 for(i=0;i<3;i++) devplane[2][i] = devpoint[pn[j][2]][i];
209                 for(i=0;i<3;i++) devplane[3][i] = devpoint[pn[j][3]][i];
210
211                 inout = newGtInsideTest( point, devplane, (int)4, (int)2 );
212
213                 if(inout == INSIDE ) break;
214             }
215
216                                 /* devplane の位置により領域を狭める */
217             switch(j) {
218             case 0:
219                 for(i=0;i<3;i++) devpoint[2][i] = devpoint[5][i];
220                 for(i=0;i<3;i++) devpoint[3][i] = devpoint[8][i];
221                 for(i=0;i<3;i++) devpoint[0][i] = devpoint[4][i];
222                 right  = devidu;
223                 bottom = devidw;
224                 devidu = devidu - devid*0.5;
225                 devidw = devidw + devid*0.5;
226                 break;
227             case 1:
228                 for(i=0;i<3;i++) devpoint[1][i] = devpoint[5][i];
229                 for(i=0;i<3;i++) devpoint[0][i] = devpoint[8][i];
230                 for(i=0;i<3;i++) devpoint[3][i] = devpoint[6][i];
231                 left   = devidu;
232                 bottom = devidw;
233                 devidu = devidu + devid*0.5;
234                 devidw = devidw + devid*0.5;
235                 break;
236             case 2:
237                 for(i=0;i<3;i++) devpoint[1][i] = devpoint[8][i];
238                 for(i=0;i<3;i++) devpoint[2][i] = devpoint[6][i];
239                 for(i=0;i<3;i++) devpoint[0][i] = devpoint[7][i];
240                 left = devidu;
241                 top  = devidw;
242                 devidu = devidu + devid*0.5;
243                 devidw = devidw - devid*0.5;
244                 break;
245             case 3:
246                 for(i=0;i<3;i++) devpoint[3][i] = devpoint[7][i];
247                 for(i=0;i<3;i++) devpoint[2][i] = devpoint[8][i];
248                 for(i=0;i<3;i++) devpoint[1][i] = devpoint[4][i];
249                 right = devidu;
250                 top   = devidw;
251                 devidu = devidu - devid*0.5;
252                 devidw = devidw - devid*0.5;
253                 break;
254             }
255
256            devid *= 0.5;
257            *u = devidu;
258            *w = devidw;
259         }
260
261         for(i=0;i<3;i++) result[i] = devpoint[8][i];
262
263         return(TRUE);
```

```
264
265    }
```

```
1     /*
2     ================================================================================
3       ファイル名 : inside_test.c
4
5       機能       : 多角形の内外判定 （凸多面体）
6
7       履歴       : 1993年7月2日 ; 作成 ; 涌 真吾
8     ================================================================================
9     */
10
11    #include <stdio.h>
12    #include <math.h>
13    #include <gl.h>
14    #include "bspline.h"
15
16    /*
17    ================================================================================
18      関数名 : BOOLEAN GtInsideTest( COORD p[XYZ], COORD V[][XYZ],
19                                                          int n, int plane )
20
21      引数   : p      (I)   点座標
22               V      (I)   多角形を構成する頂点配列（反時計まわりあるいは時計まわり
      であること）
23               n      (I)   多角形を構成する頂点数
24               plane  (I)   内外判定をする平面
25                            0 : YZ 平面
26                            1 : XZ 平面
27                            2 : XY 平面
28
29      戻り値 : 内外判定
30                 0 : 外部
31                 1 : 内部
32
33      機能   : 2次元平面にある点が多角形の内部にあるか外部にあるかを判定する。
34
35    --------------------------------------------------------------------------------
36
37    */
38
39    BOOLEAN newGtInsideTest( COORD p[XYZ], COORD V[][XYZ], int n, int plane )
40    {
41      int       i,j;
42      COORD     vec1[XYZ], vec2[XYZ], cross_v1v2[XYZ], basevec[4][XYZ];
43      float     vecflag;
44
45      for( i=0; i<n; i++ ) {
46
47        GtSubVectors( V[(i+1)%n], V[i%n], vec1 );
48        GtNormalize( vec1, vec1 );
49
50        GtSubVectors( p, V[i%n], vec2 );
51        GtNormalize( vec2, vec2 );
52
53        GtCrossProduct( vec1, vec2, cross_v1v2 );
54        basevec[i][plane] = cross_v1v2[plane];
55
56        vecflag = basevec[0][plane] * basevec[i][plane];
57        if(vecflag < 0.0) return( OUTSIDE );
58      }
59
60      return( INSIDE );
61
62    }
63
64
65
```



```
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
```

```
1     #include          <gl/gl.h>
2     #include          <gl/device.h>
3
4     #include          "std.h"
5     #include          "wavefront.h"
6     #include          "cimage.h"
7     #include          "string.h"
8     #include          "bspline.h"
9
10    void ReadMovement( movect )
11    MoVector          *movect;
12    {
13      FILE            *fp;
14      float           x , y , z;
15      char            *fname;
16      int            name;
17      int             i , j , jj ;
18
19      fname = "/home/komori/atr/data/mov/movement.new";
20
21      if( NULL == ( fp = fopen( fname , "r" ) ) )
22      {
23        printf( "Cannot open file\n");
24        exit( 1 );
25      }
26
27      fscanf( fp , "%d %d" , &i , &j );
28      movect->fnum = i;
29      movect->pnum = j;
30
31      for( i = 3 ; i <= 56 ; i++ ){
32        printf("controlpointnumber = %d\n", i);
33
34        fscanf( fp , "%d" , &name );
35        for( j = 1 ; j <= movect->fnum ; j++ ){
36          fscanf( fp , "%d %f %f %f" , &jj , &x , &y , &z );
37          movect->movbase[i][j].x = (double)x;
38          movect->movbase[i][j].y = (double)y;
39          movect->movbase[i][j].z = (double)z;
40    /*
41          printf( "i = %d j = %d movector = (%f)\n" ,i, j, movect->movbase[i][j].x);
42    */
43        }
44      }
45    /*  printf("%f", movect->movbase[56][225]);
46    */
47      fclose(fp);
48    }
```

```
1     /* マーカ番号(98点)を計測されたマーカ番号(56点)で表す */
2
3     #include         <gl/gl.h>
4     #include         <gl/device.h>
5
6     #include         "std.h"
7     #include         "wavefront.h"
8     #include         "cimage.h"
9     #include         "string.h"
10    #include         "bspline.h"
11
12    void ReadMatchCP( movect )
13    MoVector         *movect;
14    {
15      FILE           *fp;
16      char           *fname;
17      int            i, inumber, usecpnum, lr;
18
19      fname = "controlmatch.dat";
20
21      if( NULL == ( fp = fopen( fname , "r" ) ) )
22      {
23        printf( "Cannot open file\n");
24        exit( 1 );
25      }
26
27      for( i = 0 ; i < CONTROLNUM ; i++ )
28      {
29        fscanf( fp , "%d %d %d" , &inumber , &usecpnum , &lr);
30        movect->usecpnum[i] = usecpnum;
31          printf( "i = %d movect->usecpnum= %d\n" ,i, movect->usecpnum[i]);
32      }
33      fclose(fp);
34    }
```

A42

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8     #include        "bspline.h"
9
10    void AllXYZ( ip, movect )
11    InflPoint       *ip;
12    MoVector        *movect;
13    {
14      int           i , j;
15
16      ReadMatchCP( movect );
17    /*  printf("movect->usecpnum[0] = %d\n", movect->usecpnum[0]);
18      printf("ip->mgrid[0] = %f\n", ip->mgrid[0].x);
19      printf("movect->pnum = %d\n", movect->pnum);
20      movect->mov[0][1].x = ip->mgrid[0].x;
21    */
22      for(i = 0; i < CONTROLNUM; i++)
23      {
24        movect->mov[i][1].x = ip->mgrid[i].x;
25        movect->mov[i][1].y = ip->mgrid[i].y;
26        movect->mov[i][1].z = ip->mgrid[i].z;
27    /*    printf("i= %d  movect->mov = %f\n", i, movect->mov[i][1].x);
28    */
29        for(j = 2; j <= movect->fnum; j++)
30        {
31    /*      printf("i = %d j = %d movect->movbase[%d][%d] = %f\n", i, j, movect->use
      cpnum[i], j, movect->movbase[movect->usecpnum[i]][j].x);
32    */
33          movect->mov[i][j].x = ip->mgrid[i].x + ip->side[i] * movect->movbase[movec
      t->usecpnum[i]][j].x;
34          movect->mov[i][j].y = ip->mgrid[i].y + movect->movbase[movect->usecpnum[i]
      ][j].y;
35          movect->mov[i][j].z = ip->mgrid[i].z + movect->movbase[movect->usecpnum[i]
      ][j].z;
36    /*      printf("movect->mov[%d][%d].x = %f\n", i, j, movect->mov[i][j].x);
37    */
38        }
39        printf("controlnum = %d\n", i);
40      }
41    }
```

```
1     /* 2点を結ぶ直線上に点を配置する(第2フレーム以降) */
2
3     #include         <gl/gl.h>
4     #include         <gl/device.h>
5
6     #include         "std.h"
7     #include         "wavefront.h"
8     #include         "cimage.h"
9     #include         "string.h"
10    #include         "math.h"
11
12    void StrightLineNext( movect, p1, p2, amp, x, y, framenum)
13    MoVector          *movect;
14    int               p1;
15    int               p2;
16    float             amp;
17    float             *x;
18    float             *y;
19    int               framenum;
20    {
21      int             np1, np2;
22
23      np1 = p1;
24      np2 = p2;
25
26      if(p1 < 0)
27      {
28        np1 = (-1) * p1 + 100;
29      }
30      if(p2 < 0)
31      {
32        np2 = (-1) * p2 + 100;
33      }
34
35      *x = movect->mov[np1][framenum].x + (1 + amp) * (movect->mov[np2][framenum].x
      - movect->mov[np1][framenum].x);
36      *y = movect->mov[np1][framenum].y + (1 + amp) * (movect->mov[np2][framenum].y
      - movect->mov[np1][framenum].y);
37    }
38
```

A44

```
1     /* 曲面上のu,wからx,y,zを求める */
2     #include <stdio.h>
3     #include <math.h>
4     #include <gl.h>
5     #include "bspline.h"
6
7     void UWtoxyz( MATRIX p[XYZ], MATRIX q[XYZ], COORD *uu, COORD *ww, COORD result[X
      YZ])
8     {
9
10            int    i,j,k;
11            COORD3D p2[4][4] , q2[4][4];
12
13            float  u, w;
14
15            u = *uu;
16            w = *ww;
17
18            for(i=0;i<4;i++) {
19              for(j=0;j<4;j++) {
20                for(k=0;k<3;k++) {
21                  p2[i][j][k] = p[k][i][j];
22                }
23              }
24            }
25
26            GtBsplineSurfaceControlPoint((COORD3D *)p2,
27                                         (int)4,(int)4, (COORD3D *)q2);
28
29            for(i=0;i<4;i++) {
30              for(j=0;j<4;j++) {
31                for(k=0;k<3;k++) {
32                  q[k][i][j] = q2[i][j][k];
33                }
34              }
35            }
36
37            GtBsplineSurface( q, u, w, result);
38    }
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
```



```
66
67
68
69
```

A45

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8     #include        "bspline.h"
9
10
11    void MakeTempFile( wf, ip, movect)
12    WaveFront       *wf;
13    InflPoint       *ip;
14    MoVector        *movect;
15    {
16      static  Matrix  p[3], q[3];
17      COORD           ans[3];
18      COORD           postans[3];
19
20      FILE            *fp;
21      char            *fname;
22
23
24      int             wfnum, framenum;
25      float           u, w;
26
27    short whitevec[3]= {255, 255, 255};
28    short cyanvec[3] = {0, 255, 255};
29    short bluevec[3] = {0, 0, 255};
30    short greenvec[3] = {0, 255, 0};
31    short redvec[3] = {255, 0, 0};
32    short blackvec[3] = {0, 0, 0};
33
34       fname = "/home/komori/atr/data/dirtempFILE/tempfile1.dat";
35
36       if( NULL == ( fp = fopen( fname , "w" ) ) )
37       {
38         printf("Cannot open file\n");
39         exit(1);
40       }
41
42      framenum = 1;
43      for(wfnum = 0; wfnum < wf->gridnum; wfnum++)
44      {
45         fprintf(fp, "%d  %f    %f    %f\n", wfnum, 0.0, 0.0, 0.0);
46      }
47      fprintf(fp, "%d  %f   %f   %f\n", (-1) * framenum, (float)framenum, (float)fra
      menum, (float)framenum);
48
49      for(framenum = 2; framenum <= movect->fnum; framenum++)
50      {
51        for(wfnum = 0; wfnum < wf->gridnum; wfnum++)
52        {
53          if(wf->uwbox[wfnum].ubox < 0)
54          {
55            fprintf(fp, "%d  %f    %f    %f\n", wfnum, 0.0, 0.0, 0.0);
56          }
57          else
58          {
59    /*          printf("wf->insideg[%d] = %d\n", wfnum, wf->insideg[wfnum]);
60    */
61            Point16xyzNext(ip, movect, wf->insideg[wfnum], p, framenum);
62    /*        printf("insedeg[%d] = %d\n", wfnum, wf->insideg[wfnum]);
63            printf("u = %f w = %f\n", wf->uwbox[wfnum].ubox, wf->uwbox[wfnum].wbox);
64    */
65            u = wf->uwbox[wfnum].ubox;
```



```
66            w = wf->uwbox[wfnum].wbox;
67            UWtoxyz( p, q, &u, &w , ans);
68    /*        printf("ans = %f  %f  %f\n", ans[0], ans[1], ans[2]);
69    */
70            postans[0] = ans[0] - wf->mgrid[wfnum].x;
71            postans[1] = ans[1] - wf->mgrid[wfnum].y;
72            postans[2] = ans[2] - wf->mgrid[wfnum].z;
73
74            fprintf(fp, "%d  %f    %f    %f\n", wfnum, postans[0]*0.1, postans[1]*0.
      1, postans[2]*0.1);
75          }
76        }
77        fprintf(fp, "%d  %f   %f   %f\n", (-1) * framenum, (float)framenum, (float)f
      ramenum, (float)framenum);
78        printf("framenum = %d\n", framenum);
79      }
80      fclose(fp);
81    }
82
```

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8
9     void MinuscondesideNext( movect )
10    MoVector        *movect;
11    {
12      FILE          *fp;
13      char          *fname;
14      char          num[10];
15
16      int           i, framenum;
17
18      int           newconnumber, connumber, storcr, p1, p2, pf1, pt1, pf2, pt2;
19      float         amp;
20      float         x, y;
21      int           npt2, pp2;
22
23    short whitevec[3]= {255, 255, 255};
24    short cyanvec[3] = {0, 255, 255};
25    short bluevec[3] = {0, 0, 255};
26    short greenvec[3] = {0, 255, 0};
27    short redvec[3] = {255, 0, 0};
28    short blackvec[3] = {0, 0, 0};
29
30      fname = "minuscontrol98.dat";
31
32      for(framenum = 2; framenum <= movect->fnum; framenum++)
33      {
34        c3s(bluevec);
35        if(framenum == 3)
36        {
37          c3s(greenvec);
38        }
39        if(framenum == 4)
40        {
41          c3s(redvec);
42        }
43
44        if( NULL == ( fp = fopen( fname , "r" ) ) )
45        {
46          printf( "Cannot open file\n");
47          exit( 1 );
48        }
49
50        for(i = -1; i >= -67; i--)
51        {
52          fscanf(fp , "%d %d %d %f", &connumber, &p1, &p2, &amp);
53          StrightLineNext(movect, p1, p2, amp, &x, &y, framenum);
54          newconnumber = (-1) * connumber + 100;
55          movect->mov[newconnumber][framenum].x = x;
56          movect->mov[newconnumber][framenum].y = y;
57
58          pp2 = p2;
59          if(p2 < 0)
60        . {
61              pp2 = (-1) * p2 + 100;
62          }
63          movect->mov[newconnumber][framenum].z = movect->mov[pp2][framenum].z;
64
65    /*      sprintf(num, "%d", connumber);
66          printf("framenum = %d i = %d connumber= %d\n", framenum, i, connumber);
```



```
67
68          cmov2(x, y);
69          charstr(num);
70          circf(x, y, 0.5);
71    */
72        }
73        printf("framenum = %d\n", framenum);
74        fclose(fp);
75      }
76    }
77
```

```
1     /* 指定された16個のマーカ番号のx,y,z座標値を求める(第2フレーム以降) */
2     #include        <gl/gl.h>
3     #include        <gl/device.h>
4
5     #include        "std.h"
6     #include        "wavefront.h"
7     #include        "cimage.h"
8     #include        "string.h"
9     #include        "bspline.h"
10
11    void Point16xyzNext(ip, movect, onerec, p, framenum)
12    InflPoint       *ip;
13    MoVector        *movect;
14    int             onerec;
15    Matrix          p[];
16    int             framenum;
17    {
18      int           i, pointor, j, k, onepoint;
19
20      pointor = 0;
21      for(j = 0; j < 4; j++)
22      {
23        for(k = 0; k < 4; k++)
24        {
25          onepoint = ip->numpoint[onerec][pointor];
26          if(onepoint < 0)
27          {
28            onepoint = (-1) * onepoint + 100;
29          }
30          p[0][j][k] = movect->mov[onepoint][framenum].x;
31          p[1][j][k] = movect->mov[onepoint][framenum].y;
32          p[2][j][k] = movect->mov[onepoint][framenum].z;
33          pointor = pointor++;
34        }
35      }
36    }
```

A48

```
1     #include        <gl/gl.h>
2     #include        <gl/device.h>
3
4     #include        "std.h"
5     #include        "wavefront.h"
6     #include        "cimage.h"
7     #include        "string.h"
8     #include        "bspline.h"
9
10
11    void Aberration(wf, ip, movect)
12    WaveFront       *wf;
13    InflPoint       *ip;
14    MoVector        *movect;
15    {
16      static  Matrix  p[3], q[3];
17      COORD           ans[3];
18      COORD           postans[2][2][3];
19
20      FILE            *fp;
21      char            *fname;
22
23
24      int             i ,j , framenum, recnum;
25      int           u, w;
26      float         fu, fw, sum, sum14, sum1, sum2, sum3, sum4, ave;
27
28      float         frameave, framemax, framemin, framesum, avemax, avemin;
29
30    short whitevec[3]= {255, 255, 255};
31    short cyanvec[3] = {0, 255, 255};
32    short bluevec[3] = {0, 0, 255};
33    short greenvec[3] = {0, 255, 0};
34    short redvec[3] = {255, 0, 0};
35    short blackvec[3] = {0, 0, 0};
36
37
38      fname = "aberration.dat";
39
40      if( NULL == ( fp = fopen( fname , "w" ) ) )
41      {
42        printf( "Cannot open file\n");
43        exit( 1 );
44      }
45
46      fprintf(fp, "%d  %d\n", RECNUM, movect->fnum);
47
48      framenum = 1;
49      sum = 0.0;
50      ave = 0.0;
51      j = 0;
52
53      frameave = 0.0;
54      framemax = -999.0;
55      framemin = 999.0;
56      framesum = 0.0;
57
58      ave = 0.0;
59      avemax = -999.0;
60      avemin = 999;
61
62      for(recnum = 1; recnum <= RECNUM; recnum++)
63      {
64        Point16xyz( wf, ip, recnum, p);
65        for(u = 0; u <= 1; u++)
66        {
```



```
67          for(w = 0; w <= 1; w++)
68          {
69            fu = (float)u;
70            fw = (float)w;
71
72            UWtoxyz( p, q, &fu, &fw , ans);
73            for(i = 0; i < 3; i++)
74            {
75              postans[u][w][i] = ans[i];
76            }
77          }
78        }
79        sum1 = sqrt( (postans[0][0][0] - p[0][1][1]) * (postans[0][0][0] - p[0][1][1
      ]) + (postans[0][0][1] - p[1][1][1]) * (postans[0][0][1] - p[1][1][1]) + (postan
      s[0][0][2] - p[2][1][1]) * (postans[0][0][2] - p[2][1][1]));
80        sum2 = sqrt( (postans[0][1][0] - p[0][2][1]) * (postans[0][1][0] - p[0][2][1
      ]) + (postans[0][1][1] - p[1][2][1]) * (postans[0][1][1] - p[1][2][1]) + (postan
      s[0][1][2] - p[2][2][1]) * (postans[0][1][2] - p[2][2][1]));
81        sum3 = sqrt( (postans[1][0][0] - p[0][1][2]) * (postans[1][0][0] - p[0][1][2
      ]) + (postans[1][0][1] - p[1][1][2]) * (postans[1][0][1] - p[1][1][2]) + (postan
      s[1][0][2] - p[2][1][2]) * (postans[1][0][2] - p[2][1][2]));
82        sum4 = sqrt( (postans[1][1][0] - p[0][2][2]) * (postans[1][1][0] - p[0][2][2
      ]) + (postans[1][1][1] - p[1][2][2]) * (postans[1][1][1] - p[1][2][2]) + (postan
      s[1][1][2] - p[2][2][2]) * (postans[1][1][2] - p[2][2][2]));
83
84      /*    printf("p[0][1][2] = %f p[1][1][2] = %f p[2][1][2] = %f postans[1][0][0] =
       %f postans[1][0][1] = %f postans[1][0][2] = %f sum1 = %f\n", p[0][1][2], p[1][1
      ][2], p[2][1][2], postans[1][0][0], postans[1][0][1], postans[1][0][2], sum1);
85      */
86
87        sum = sum + sum1 + sum2 + sum3 + sum4;
88      /*    printf("sum1 = %f sum2 = %f sum3 = %f sum4 = %f\n", sum1, sum2, sum3, sum4
      );
89      */
90        framesum = framesum + sum1 + sum2 + sum3 + sum4;
91
92        if(framemax < sum1)
93        {
94          framemax = sum1;
95        }
96        if(framemax < sum2)
97        {
98          framemax = sum2;
99        }
100       if(framemax < sum3)
101       {
102         framemax = sum3;
103       }
104       if(framemax < sum4)
105       {
106         framemax = sum4;
107       }
108       if(framemin > sum1)
109       {
110         framemin = sum1;
111       }
112       if(framemin > sum2)
113       {
114         framemin = sum2;
115       }
116       if(framemin > sum3)
117       {
118         framemin = sum3;
119       }
120       if(framemin > sum4)
121       {
```

```
122         framemin = sum4;
123       }
124
125       j = j + 1;
126     }
127
128       frameave = framesum / RECNUM /4.0;
129       fprintf(fp, "%d  %f  %f  %f\n", framenum, frameave, framemin, framemax);
130
131     if(avemax < framemax)
132     {
133       avemax = framemax;
134     }
135     if(avemin > framemin)
136     {
137       avemin = framemin;
138     }
139
140     for(framenum = 2; framenum <= movect->fnum; framenum++)
141     {
142       frameave = 0.0;
143       framemax = -999.0;
144       framemin = 999.0;
145       framesum = 0.0;
146       for(recnum = 1; recnum <= RECNUM; recnum++)
147       {
148         Point16xyzNext(ip, movect, recnum, p, framenum);
149         UWtoxyz( p, q, &fu, &fw , ans);
150         for(u = 0; u <= 1; u++)
151         {
152           for(w = 0; w <= 1; w++)
153           {
154             fu = (float)u;
155             fw = (float)w;
156
157             UWtoxyz( p, q, &fu, &fw , ans);
158             for(i = 0; i < 3; i++)
159             {
160               postans[u][w][i] = ans[i];
161             }
162           }
163         }
164         j = j + 1;
165         sum1 = sqrt( (postans[0][0][0] - p[0][1][1]) * (postans[0][0][0] - p[0][1
      [1]) + (postans[0][0][1] - p[1][1][1]) * (postans[0][0][1] - p[1][1][1]) + (post
      ans[0][0][2] - p[2][1][1]) * (postans[0][0][2] - p[2][1][1]));
166         sum2 = sqrt( (postans[0][1][0] - p[0][2][1]) * (postans[0][1][0] - p[0][2
      [1]) + (postans[0][1][1]
167     - p[1][2][1]) * (postans[0][1][1] - p[1][2][1]) + (postans[0][1][2] - p[2][2][1
      ) * (postans[0][1][2] - p[2][2][1]));
168         sum3 = sqrt( (postans[1][0][0] - p[0][1][2]) * (postans[1][0][0] - p[0][1
      [2]) + (postans[1][0][1]
169     - p[1][1][2]) * (postans[1][0][1] - p[1][1][2]) + (postans[1][0][2] - p[2][1][2
      ) * (postans[1][0][2] - p[2][1][2]));
170         sum4 = sqrt( (postans[1][1][0] - p[0][2][2]) * (postans[1][1][0] - p[0][2
      [2]) + (postans[1][1][1]
171     - p[1][2][2]) * (postans[1][1][1] - p[1][2][2]) + (postans[1][1][2] - p[2][2][2
      ) * (postans[1][1][2] - p[2][2][2]));
172         sum = sum + sum1 + sum2 + sum3 + sum4;
173
174         framesum = framesum + sum1 + sum2 + sum3 + sum4;
175
176         if(framemax < sum1)
177         {
178           framemax = sum1;
179         }
```



```
180         if(framemax < sum2)
181         {
182           framemax = sum2;
183         }
184         if(framemax < sum3)
185         {
186           framemax = sum3;
187         }
188         if(framemax < sum4)
189         {
190           framemax = sum4;
191         }
192         if(framemin > sum1)
193         {
194           framemin = sum1;
195         }
196         if(framemin > sum2)
197         {
198           framemin = sum2;
199         }
200         if(framemin > sum3)
201         {
202           framemin = sum3;
203         }
204         if(framemin > sum4)
205         {
206           framemin = sum4;
207         }
208
209     /*      printf("p[0][1][2] = %f p[1][1][2] = %f p[2][1][2] = %f postans[1][0][0]
       = %f postans[1][0][1] = %f postans[1][0][2] = %f sum1 = %f\n", p[0][1][2], p[1]
      [1][2], p[2][1][2], postans[1][0][0], postans[1][0][1], postans[1][0][2], sum3);
210         printf("sum1 = %f sum2 = %f sum3 = %f sum4 = %f\n", sum1, sum2, sum3, sum4
      );
211         printf("ave %d = %f\n", framenum, (sum1 + sum2 + sum3 + sum4)/4.0);
212     */
213       }
214       frameave = framesum / RECNUM /4.0;
215       fprintf(fp, "%d  %f  %f  %f\n", framenum, frameave, framemin, framemax);
216       if(avemax < framemax)
217       {
218         avemax = framemax;
219       }
220       if(avemin > framemin)
221       {
222         avemin = framemin;
223       }
224     }
225
226     printf("j = %d\n", j);
227     ave = sum / (j * 4.0);
228     fprintf(fp, "%f\n", ave);
229     fprintf(fp, "%f\n", avemax);
230     fprintf(fp, "%f\n", avemin);
231     fclose(fp);
232   }
```

```
1     CC        =  cc
2     LIBS      =  -lm
3     SGILIB    =  -lc_s -lgl_s -lfm_s -limage
4     CFLAGS    =  -g
5     HOMEDIR   =  /home/komori
6     HEADER    =  $(HOMEDIR)/atr/header2
7
8     OBJECTS   =\
9             main.o\
10            wavefront.o\
11            cmath.o\
12            inside_test.o\
13            vector_utility.o\
14            matrix_utility.o\
15            inoutcheck.o\
16            readrecinf.o\
17            line.o\
18            readmicon.o\
19            writecon.o\
20            catwfmgrid.o\
21            writewf.o\
22            inputzflag.o\
23            storeuw.o\
24            point16xyz.o\
25            newbspline_surface.o\
26            bspline_surface_control_point.o\
27            newxyztouw.o\
28            bspline_control_point.o\
29            readpoint16.o\
30            readmov.o\
31            newinside_test.o\
32            allxyz.o\
33            readmatchcp.o\
34            linenext.o\
35            readmiconnext.o\
36            maketempfile.o\
37            newuwtoxyz.o\
38            point16xyznext.o\
39            aberration.o\
40            divgroup.o
41
42    a.out: $(OBJECTS)
43            $(CC) $(OBJECTS) $(SGILIB) $(LIBS)
44
45    main.o: main.c
46            $(CC) $(CFLAGS) -I$(HEADER) -c main.c
47
48    wavefront.o: wavefront.c
49            $(CC) $(CFLAGS) -I$(HEADER)  -c wavefront.c
50
51    cmath.o: cmath.c
52            $(CC) $(CFLAGS) -I$(HEADER) -c cmath.c
53
54    divgroup.o: divgroup.c
55            $(CC) $(CFLAGS) -I$(HEADER) -c divgroup.c
56
57    inside_test.o: inside_test.c
58            $(CC) $(CFLAGS) -I$(HEADER) -c inside_test.c
59
60    matrix_utility.o: matrix_utility.c
61            $(CC) $(CFLAGS) -I$(HEADER) -c matrix_utility.c
62
63    vector_utility.o: vector_utility.c
64            $(CC) $(CFLAGS) -I$(HEADER) -c vector_utility.c
65
66    inoutcheck.o: inoutcheck.c
```



```
67            $(CC) $(CFLAGS) -I$(HEADER) -c inoutcheck.c
68
69    readrecinf.o: readrecinf.c
70            $(CC) $(CFLAGS) -I$(HEADER) -c readrecinf.c
71
72    line.o: line.c
73            $(CC) $(CFLAGS) -I$(HEADER) -c line.c
74
75    readmicon.o: readmicon.c
76            $(CC) $(CFLAGS) -I$(HEADER) -c readmicon.c
77
78    writecon.o: writecon.c
79            $(CC) $(CFLAGS) -I$(HEADER) -c writecon.c
80
81    catwfmgrid.o: catwfmgrid.c
82            $(CC) $(CFLAGS) -I$(HEADER) -c catwfmgrid.c
83
84    writewf.o: writewf.c
85            $(CC) $(CFLAGS) -I$(HEADER) -c writewf.c
86
87    inputzflag.o: inputzflag.c
88            $(CC) $(CFLAGS) -I$(HEADER) -c inputzflag.c
89
90    storeuw.o: storeuw.c
91            $(CC) $(CFLAGS) -I$(HEADER) -c storeuw.c
92
93    point16xyz.o: point16xyz.c
94            $(CC) $(CFLAGS) -I$(HEADER) -c point16xyz.c
95
96    newbspline_surface.o: newbspline_surface.c
97            $(CC) $(CFLAGS) -I$(HEADER) -c newbspline_surface.c
98
99    bspline_surface_control_point.o: bspline_surface_control_point.c
100           $(CC) $(CFLAGS) -I$(HEADER) -c bspline_surface_control_point.c
101
102   newxyztouw.o: newxyztouw.c
103           $(CC) $(CFLAGS) -I$(HEADER) -c newxyztouw.c
104
105   bspline_control_point.o: bspline_control_point.c
106           $(CC) $(CFLAGS) -I$(HEADER) -c bspline_control_point.c
107
108   readpoint16.o: readpoint16.c
109           $(CC) $(CFLAGS) -I$(HEADER) -c readpoint16.c
110
111   readmov.o: readmov.c
112           $(CC) $(CFLAGS) -I$(HEADER) -c readmov.c
113
114   newinside_test.o: newinside_test.c
115           $(CC) $(CFLAGS) -I$(HEADER) -c newinside_test.c
116
117   allxyz.o: allxyz.c
118           $(CC) $(CFLAGS) -I$(HEADER) -c allxyz.c
119
120   readmatchcp.o: readmatchcp.c
121           $(CC) $(CFLAGS) -I$(HEADER) -c readmatchcp.c
122
123   readmiconnext.o: readmiconnext.c
124           $(CC) $(CFLAGS) -I$(HEADER) -c readmiconnext.c
125
126   linenext.o: linenext.c
127           $(CC) $(CFLAGS) -I$(HEADER) -c linenext.c
128
129   maketempfile.o: maketempfile.c
130           $(CC) $(CFLAGS) -I$(HEADER) -c maketempfile.c
131
132   newuwtoxyz.o: newuwtoxyz.c
```

```
133             $(CC) $(CFLAGS) -I$(HEADER) -c newuwtoxyz.c
134
135     point16xyznext.o: point16xyznext.c
136             $(CC) $(CFLAGS) -I$(HEADER) -c point16xyznext.c
137
138     aberration.o: aberration.c
139             $(CC) $(CFLAGS) -I$(HEADER) -c aberration.c
140
```

Appendix D

作成したデータファイル

1	1	8	6	1	3
2	2	5	7	2	0
3	3	17	15	8	10
4	4	15	13	6	8
5	5	13	11	4	6
6	6	11	12	5	4
7	7	12	14	7	5
8	8	14	16	9	7
9	9	24	22	15	17
10	10	22	20	13	15
11	11	20	18	11	13
12	12	18	19	12	11
13	13	19	21	14	12
14	14	21	23	16	14
15	15	53	51	45	49
16	16	51	41	43	45
17	17	41	39	37	43
18	18	39	38	36	37
19	19	38	40	42	36
20	20	40	50	44	42
21	21	50	52	48	44
22	22	61	59	53	55
23	23	59	57	51	53
24	24	57	65	41	51
25	25	65	63	39	41
26	26	63	62	38	39
27	27	62	64	40	38
28	28	64	56	50	40
29	29	56	58	52	50
30	30	58	60	54	52
31	31	73	71	59	61
32	32	71	69	57	59
33	33	69	67	65	57
34	34	67	88	63	65
35	35	88	86	62	63
36	36	86	87	64	62
37	37	87	66	56	64
38	38	66	68	58	56
39	39	68	70	60	58
40	40	80	92	69	71
41	41	92	90	67	69
42	42	89	91	68	66
43	43	91	79	70	68
44	44	78	94	92	80
45	45	78	76	96	94
46	46	76	74	97	96
47	47	74	75	95	97
48	48	75	77	93	95
49	49	77	79	91	93
50	50	85	83	76	78
51	51	83	81	74	76
52	52	81	82	75	74
53	53	82	84	77	75
54	54	55	53	45	49
55	55	52	54	48	44
56	56	73	80	71	61
57	57	70	72	60	58
58	58	90	88	63	67
59	59	87	89	66	64
60	60	53	51	43	45
61	61	50	52	44	42
62	62	61	71	59	55
63	63	67	63	39	65
64	64	64	66	56	40
65	65	68	70	58	56
66	66	80	78	69	71

67

1	-1	-9	3	-2	-1
2	-2	3	1	-3	-2
3	-3	1	-10	-4	-3
4	-4	-10	0	-5	-4
5	-5	0	2	-6	-5
6	-6	2	-11	-7	-6
7	-7	-13	10	-9	-8
8	-8	10	8	3	-9
9	-9	6	4	-10	1
10	-10	4	5	0	-10
11	-11	7	9	-11	2
12	-12	9	-14	-12	-11
13	-13	-15	17	10	-13
14	-14	16	-16	-14	9
15	-15	-17	24	17	-15
16	-16	23	-18	-16	16
17	-17	-19	-20	24	-17
18	-18	-20	-21	22	24
19	-19	-21	-22	20	22
20	-20	-22	-23	18	20
21	-21	-23	-24	19	18
22	-22	-24	-25	21	19
23	-23	-25	-26	23	21
24	-24	-26	-27	-18	23
25	-25	-38	49	-29	-28
26	-26	49	45	-30	-29
27	-27	45	43	-31	-30
28	-28	43	37	-32	-31
29	-29	37	36	-33	-32
30	-30	36	42	-34	-33
31	-31	42	44	-35	-34
32	-32	44	48	-36	-35
33	-33	48	-40	-37	-36
34	-34	-39	55	49	-38
35	-35	55	53	49	-38
36	-36	52	54	-40	48
37	-37	54	-41	-40	48
38	-38	-42	61	55	-39
39	-39	60	-43	-41	54
40	-40	-44	73	61	-42
41	-41	-45	-43	60	70
42	-42	72	-46	-43	60
43	-43	-47	-48	73	-44
44	-44	-48	-49	80	73
45	-45	-48	80	71	73
46	-46	79	-52	-45	70
47	-47	79	-52	72	70
48	-48	-52	-50	-46	72
49	-49	-48	-51	78	80
50	-50	-49	-51	78	80
51	-51	90	-61	88	67
52	-52	-61	-62	86	88
53	-53	-62	-63	87	86
54	-54	-63	-64	89	87
55	-55	-63	89	66	87
56	-56	90	-60	-61	88
57	-57	92	94	-60	90
58	-58	94	96	-67	-60
59	-59	96	97	-66	-67
60	-60	97	95	-65	-66
61	-61	95	93	-64	-65
62	-62	-64	93	91	89
63	-63	-53	76	78	-51
64	-64	-53	85	78	-51
65	-65	-54	-55	85	-53
66	-66	-55	-56	83	85

67	-67	-56	-57	81	83
68	-68	-57	-58	82	81
69	-69	-58	-59	84	82
70	-70	-59	-50	-52	84
71	-71	84	-52	79	77
72	-72	-53	85	78	80
73					

1	-9	1	3	1.0
2	-10	11	4	1.0
3	-11	16	9	1.0
4	-1	10	-9	0.3
5	-2	8	3	0.3
6	-3	6	1	0.3
7	-4	4	-10	0.3
8	-5	5	0	0.3
9	-6	7	2	0.3
10	-7	9	-11	0.3
11	-13	8	10	0.3
12	-14	7	9	0.3
13	-15	15	17	0.3
14	-16	14	16	0.3
15	-17	22	24	0.3
16	-18	21	23	0.3
17	-20	17	24	0.3
18	-21	15	22	0.3
19	-22	13	20	0.3
20	-23	11	18	0.3
21	-24	12	19	0.3
22	-25	14	21	0.3
23	-26	16	23	0.3
24	-8	3	-9	0.3
25	-12	2	-11	0.3
26	-19	-15	-17	0.3
27	-27	-25	-26	0.3
28	-29	55	49	0.3
29	-30	53	45	0.3
30	-31	51	43	0.3
31	-32	39	37	0.3
32	-33	38	36	0.3
33	-34	50	42	0.3
34	-35	52	44	0.3
35	-36	54	48	0.3
36	-38	45	49	0.3
37	-39	53	55	0.3
38	-42	59	61	0.3
39	-44	71	73	0.3
40	-40	44	48	0.3
41	-41	52	54	0.3
42	-43	58	60	0.3
43	-46	70	72	0.3
44	-28	-30	-29	0.3
45	-37	-35	-36	0.3
46	-48	61	73	1.0
47	-47	80	-48	0.3
48	-45	58	70	0.3
49	-52	91	79	0.3
50	-50	79	-52	0.3
51	-49	71	80	0.7
52	-51	94	78	0.3
53	-53	83	85	0.3
54	-55	78	85	0.3
55	-56	76	83	0.3
56	-57	74	81	0.3
57	-58	75	82	0.3
58	-59	77	84	0.3
59	-54	-51	-53	0.3
60	-60	78	94	0.3
61	-67	76	96	0.3
62	-66	74	97	0.3
63	-65	75	95	0.3
64	-64	77	93	0.3
65	-63	64	87	0.3
66	-62	62	86	0.3

67	-61	63	88	0.3

1	0	3	1
2	1	3	-1
3	2	4	1
4	3	4	-1
5	4	5	1
6	5	6	1
7	6	6	-1
8	7	7	1
9	8	7	-1
10	9	8	1
11	10	8	-1
12	11	9	1
13	12	10	1
14	13	10	-1
15	14	11	1
16	15	11	-1
17	16	12	1
18	17	12	-1
19	18	13	1
20	19	14	1
21	20	14	-1
22	21	15	1
23	22	15	-1
24	23	16	1
25	24	16	-1
26	25	17	1
27	26	17	-1
28	27	18	1
29	28	18	-1
30	29	19	1
31	30	19	-1
32	31	20	1
33	32	20	-1
34	33	21	1
35	34	21	-1
36	35	22	1
37	36	23	1
38	37	23	-1
39	38	24	1
40	39	24	-1
41	40	25	1
42	41	25	-1
43	42	26	1
44	43	26	-1
45	44	27	1
46	45	27	-1
47	46	28	1
48	47	28	-1
49	48	29	1
50	49	29	-1
51	50	30	1
52	51	30	-1
53	52	31	1
54	53	31	-1
55	54	32	1
56	55	32	-1
57	56	33	1
58	57	33	-1
59	58	34	1
60	59	34	-1
61	60	35	1
62	61	35	-1
63	62	36	1
64	63	36	-1
65	64	37	1
66	65	37	-1

67	66	38	1
68	67	38	-1
69	68	39	1
70	69	39	-1
71	70	40	1
72	71	40	-1
73	72	41	1
74	73	41	-1
75	74	42	1
76	75	43	1
77	76	43	-1
78	77	44	1
79	78	44	-1
80	79	45	1
81	80	45	-1
82	81	46	1
83	82	47	1
84	83	47	-1
85	84	48	1
86	85	48	-1
87	86	49	1
88	87	50	1
89	88	50	-1
90	89	51	1
91	90	51	-1
92	91	52	1
93	92	52	-1
94	93	53	1
95	94	53	-1
96	95	54	1
97	96	54	-1
98	97	55	1
99			

1	1	17	15	13	11	10	8	6	4	-9	3	1	-10	-1	-2	-3	-
2	2	11	12	14	16	4	5	7	9	-10	0	2	-11	-4	-5	-6	-
3	3	-17	24	22	20	-15	17	15	13	-13	10	8	6	-8	-9	3	
4	4	24	22	20	18	17	15	13	11	10	8	6	4	-9	3	1	-1
5	5	22	20	18	19	15	13	11	12	8	6	4	5	3	1	-10	
6	6	20	18	19	21	13	11	12	14	6	4	5	7	1	-10	0	
7	7	18	19	21	23	11	12	14	16	4	5	7	9	-10	0	2	-1
8	8	19	21	23	-18	12	14	16	-16	5	7	9	-14	0	2	-11	-1
9	9	-19	-20	-21	-22	-17	24	22	20	-15	17	15	13	-13	10	8	
10	10	-20	-21	-22	-23	24	22	20	18	17	15	13	11	10	8	6	
11	11	-21	-22	-23	-24	22	20	18	19	15	13	11	12	8	6	4	
12	12	-22	-23	-24	-25	20	18	19	21	13	11	12	14	6	4	5	
13	13	-23	-24	-25	-26	18	19	21	23	11	12	14	16	4	5	7	
14	14	-24	-25	-26	-27	19	21	23	-18	12	14	16	-16	5	7	9	-1
15	15	61	59	57	65	55	53	51	41	-38	49	45	43	-28	-29	-30	-3
16	16	59	57	65	63	53	51	41	39	49	45	43	37	-29	-30	-31	-3
17	17	57	65	63	62	51	41	39	38	45	43	37	36	-30	-31	-32	-3
18	18	65	63	62	64	41	39	38	40	43	37	36	42	-31	-32	-33	-3
19	19	63	62	64	56	39	38	40	50	37	36	42	44	-32	-33	-34	-3
20	20	62	64	56	58	38	40	50	52	36	42	44	48	-33	-34	-35	-3
21	21	64	56	58	60	40	50	52	54	42	44	48	-40	-34	-35	-36	-3
22	22	-44	73	71	69	-42	61	59	57	-39	55	53	51	-38	49	45	4
23	23	73	71	69	67	61	59	57	65	55	53	51	41	-38	49	45	4
24	24	71	69	67	88	59	57	65	63	53	51	41	39	49	45	43	3
25	25	69	67	88	86	57	65	63	62	51	41	39	38	45	43	37	3
26	26	67	88	86	87	65	63	62	64	41	39	38	40	43	37	36	4
27	27	88	86	87	66	63	62	64	56	39	38	40	50	37	36	42	4
28	28	86	87	66	68	62	64	56	58	38	40	50	52	36	42	44	4
29	29	87	66	68	70	64	56	58	60	40	50	52	54	42	44	48	-4
30	30	68	70	72	-46	56	58	60	-43	50	52	54	-41	42	44	48	-4
31	31	-47	-48	80	92	-44	73	71	69	-42	61	59	57	-39	55	53	5
32	32	-48	80	92	90	73	71	69	67	61	59	57	65	55	53	51	4
33	33	80	92	90	-61	71	69	67	88	59	57	65	63	53	51	41	3

34	34	92	90	-61	-62	69	67	88	86	57	65	63	62	51	41	39	3
35	35	90	-61	-62	-63	67	88	86	87	65	63	62	64	41	39	38	4
36	36	-61	-62	-63	89	88	86	87	66	63	62	64	56	39	38	40	5
37	37	-62	-63	89	91	86	87	66	68	62	64	56	58	38	40	50	5
38	38	-63	89	91	79	87	66	68	70	64	56	58	60	40	50	52	5
39	39	89	91	79	-52	66	68	70	-45	56	58	60	-43	50	52	54	-4
40	40	-51	78	94	-60	-48	80	92	90	73	71	69	67	61	59	57	6
41	41	78	94	-60	-61	80	92	90	88	71	69	67	63	59	57	65	3
42	42	-63	-64	93	77	87	89	91	79	64	66	68	70	40	56	58	6
43	43	-64	93	77	84	89	91	79	-52	66	68	70	-45	56	58	60	-4
44	44	-53	76	96	-67	-51	78	94	-60	-48	80	92	90	73	71	69	6
45	45	81	74	97	-66	83	76	96	-67	85	78	94	-60	-53	80	92	9
46	46	82	75	95	-65	81	74	97	-66	83	76	96	-67	85	78	94	-6
47	47	84	77	93	-64	82	75	95	-65	81	74	97	-66	83	76	96	-6
48	48	-52	79	91	89	84	77	93	-64	82	75	95	-65	81	74	97	-6
49	49	-45	70	68	66	-52	79	91	89	84	77	93	-64	82	75	95	-6
50	50	-57	81	74	97	-56	83	76	96	-55	85	78	94	-54	-53	80	9
51	51	-58	82	75	95	-57	81	74	97	-56	83	76	96	-55	85	78	9
52	52	-59	84	77	93	-58	82	75	95	-57	81	74	97	-56	83	76	9
53	53	-50	-52	79	91	-59	84	77	93	-58	82	75	95	-57	81	74	9
54	54	-42	61	59	57	-39	55	53	51	-38	49	45	43	-28	-29	-30	-3
55	55	56	58	60	-43	50	52	54	-41	42	44	48	-40	-34	-35	-36	-3
56	56	-47	-48	-49	-51	-44	73	80	78	-42	61	71	69	-39	55	59	5
57	57	91	79	-52	-50	68	70	72	-46	56	58	60	-43	50	52	54	-4
58	58	94	-60	-61	-62	92	90	88	86	69	67	63	62	57	65	39	3
59	59	-62	-63	-64	93	86	87	89	91	62	64	66	68	38	40	56	5