ＡＴＲテクニカルレポート表紙

〔非公開〕

ＴＲ－Ｍ－００３３

多視点画像と奥行きマップを用いた３次元映像表現

大　川　　慶
Kei　OHKAWA

ジョンイル　パーク　　井　上　誠　喜
Jong-Il　Park　　Seiki　INOUE

１９９８．２．２７

ＡＴＲ知能映像通信研究所

多視点画像と奥行きマップを用いた
3次元映像表現

3-D Image Expression
from
Multi-View Image and Its Depth Map.

1998年2月27日

概要:

数枚の実画像とステレオマッチングによって得られる奥行き情報（奥行きマップ）を用いて任意視点映像の生成と合成の実験を行った。「任意視点ステレオ映像生成」の実験では、両眼視差による奥行き感を取り入れるためステレオ映像の生成を行い、実写映像だけによるバーチャル空間を構築して、HMDなどでその空間を体験したとき、より3次元的な臨場感溢れる映像表現を試みた。「カメラ動作に合わせた任意視点映像の生成と実時間合成」の実験では、可動のカメラで撮影したスタジオからの映像と合成できるようにカメラの動きに合わせた背景映像を生成して、3次元合成によってよりリアルな仮想空間の構築を試みた。

ＡＴＲ知能映像通信研究所 第３研究室

学外実習生　　大川 慶[1]

実習期間：平成９年１０月１３日 — 平成１０年２月２７日

[1]長岡技術科学大学 電気・電子システム工学課程

1　多視点画像と奥行きマップを用いた任意視点映像生成及び合成に関する基礎

1.1　はじめに

本研究のベースは、「数枚の実画像」と「ステレオマッチングによって得られる奥行き情報（奥行きマップ）」を用いて、奥行き情報を持つ映像を生成することである。

1.2　本研究の概念、および基礎知識

1.2.1　任意視点映像生成の概念

撮影は十字に配置した５台のカメラで行う。
実際のカメラの位置以外に仮想カメラを想定し、その仮想カメラから得られるであろう映像を５台のカメラで撮影した映像と奥行きマップから生成する。（図１）

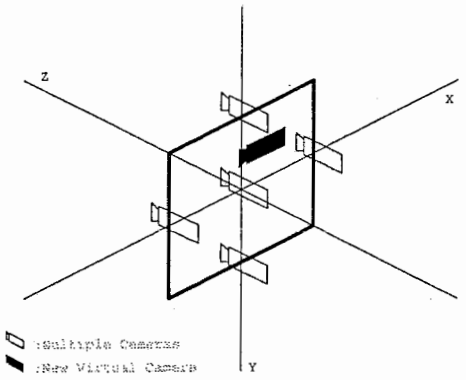


図 1: 任意視点映像生成の概念

1.2.2　任意視点映像生成法

奥行きマップを利用して新しい視点からの映像を生成するためには、２段階の処理が必要となる。第１の処理は視点移動への対応である。つまり、３次元空間中で視点を平行移動させることである。この処理は奥行きが重要なパラメータであるので３次元処理になる。 まず、投影面上の座標変換によって奥行きマップを新しい視点のものに変換する。この際、近くの物体に隠れて見えなくなる領域と新たに見えてくる領域がある。前者はカメラから最も近くにある物体のテクスチャを用いればよいため特に問題はないが、後者は未知のデータを推測する必要がある。 新たに見えてくる領域に対しては、５台のカメラの中から適当な映像を選択してテクスチャマッピングを行う。第２の処理は目的に合わせて視線方向や画角を変更することである。これは奥行きによらないので２次元処理で行うことができる。

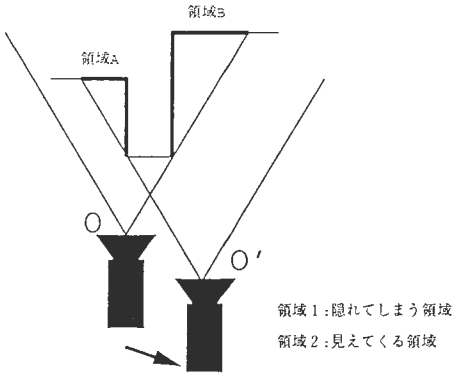


図 2: 視点の移動による隠れてしまう領域と見えてくる領域

1.2.3 奥行きマップの処理

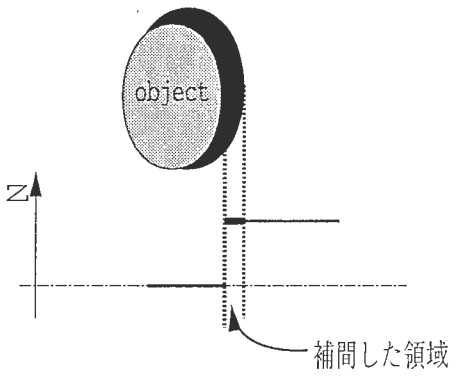


図 3: 視点の移動による見えてくる領域の補間

視点の移動により、新たに見えてくる領域と隠れてしまう領域がある（図 2）。隠れてしまう領域は、奥行きマップを任意視点のものに変換する際、手前のものが重なることになる。つまり、同じ画素に複数の点が割り当てられるのだが、このとき奥行きの手前のものを採用することで解決できる。

新たに見えてくる領域に対しては、補間処理が必要となる。

通常、視点を変えた場合、近くのオブジェクトは大きく動き、遠くのオブジェクトはほとんど動かない。そのため、近くのオブジェクトの後に存在する領域が新たに見えてくる領域であると推察される。このことから、新たにあらわれてくる領域は、隣接する遠くのオブジェクトの奥行き値で補間することにする（図 3）。

1.2.4 テクスチャマッピング

中心から見える領域に対しては、奥行きマップ変換の際、同時にテクスチャマッピングをすることができる。

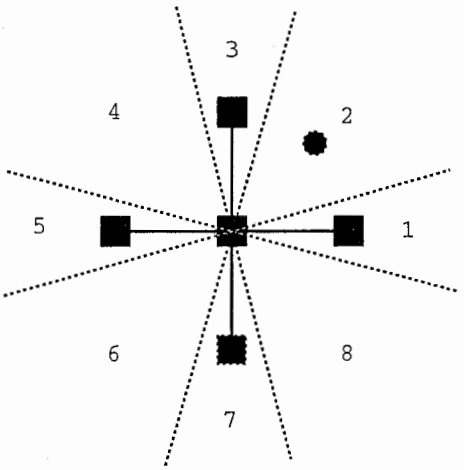


図 4: テクスチャマッピングのためのカメラ平面分割

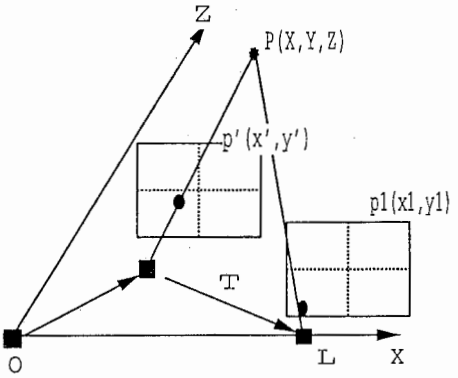


図 5: 周辺カメラ映像からのテクスチャの取得

しかし、新たに見えてくる領域に対しては、補間処理を行う必要がある。ここでは、視点の移動方向にあわせてテクスチャマッピングの対象映像を選ぶことにする。例えば、視点を右に移動すると物体の右側が見えてくるため、右のカメラから撮った映像にはその領域の情報が含まれている可能性が高い。従って、視点の移動方向を領域化しておき、移動位置に合わせて上下左右の映像からテクスチャを持ってくるようにする。

具体的には、カメラ平面（X－Y平面）を８つの領域に分割する。（図4）領域１，３，５，７に対しては３０度の幅、領域２，４，６，８に対しては６０度の幅を持たせるように分割する。さらにそれぞれの領域に対してテクスチャーを持ってくる映像を割り当てる。

任意視点を（ΔX,ΔY,ΔZ）に移動したとする。（図5） その点を含む領域によってテクスチャの取得に使用する画像を選ぶ。移動した点における奥行きマップを用いて画像面への変換を行う。つまり、（ΔX,ΔY,ΔZ）

における画像面の未知ピクセル $p'(x', y')$ からＴ（L-ΔX,L-ΔY,L-ΔZ）への変換を行う。複数の画像からテクスチャを得る場合には、それぞれの画像面へ変換を行う。距離をキーにして画素値を線形的に加え、（ΔX,ΔY,
における画像面の未知ピクセル $p'(x', y')$ の画素値を得る。

1.2.5　視点の移動

まず３次元カメラ座標系を定義する。十字に配置された多眼カメラのセンターカメラの位置を３次元カメラ座標系の原点Ｏ、水平方向をｘ軸、垂直方向をｙ軸、センターカメラの光軸方向をｚ軸とする。また、カメラの焦点距離をＦとする（図６）。

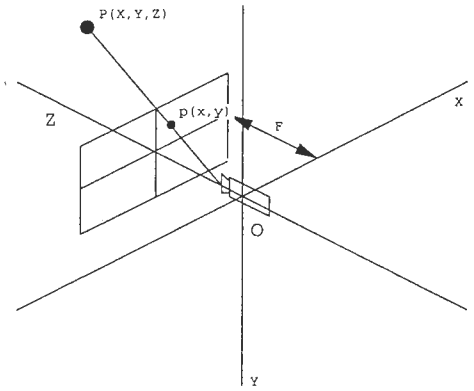


図 6:　３次元空間座標から画像平面への投影

３次元空間中の点 $P = (X, Y, Z)$ から、画像平面 $p = (x, y)$ への投影は次式で表される。

$$x = F\frac{X}{Z} \tag{1}$$

$$y = F\frac{Y}{Z} \tag{2}$$

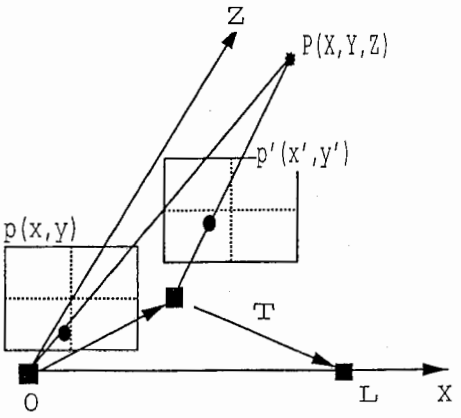


図 7: 画像面の座標変換

視点の任意の位置 $(\Delta x,\Delta y,\Delta z)$ への移動を考える。（図 7）

3 次元空間中の点 $P=(X,Y,Z)$ からセンターカメラの画像平面に投影した点を $p=(x,y)$、任意視点の仮想カメラの画像平面に投影した点を p'= (x',y') とすると、次式で与えられる。

$$x' = F\frac{X-\Delta x}{Z-\Delta z} \tag{3}$$

$$y' = F\frac{Y-\Delta y}{Z-\Delta z} \tag{4}$$

センターカメラと周りのカメラとの間の距離を L_c、3 次元空間中のある点 P (X, Y, Z) を、センターカメラの画像平面と任意視点の画像平面に投影したときの視差を d とすると、$Z=FL_c/d$ の関係が成り立つので、p と p' の関係は次のように表される。

$$x' = \frac{FL_c}{FL_c-d\Delta z}\frac{x-d\Delta x}{L_c} \tag{5}$$

$$y' = \frac{FL_c}{FL_c-d\Delta z}\frac{y-d\Delta y}{L_c} \tag{6}$$

よって、センターカメラで得られる画像面の座標 p を任意視点の仮想カメラの画像面の座標 p' に変換することができる。

1.2.6 視点の回転

視点の回転は3次元空間座標のx軸，y軸，z軸を中心とする回転であると考えられる（図8）。

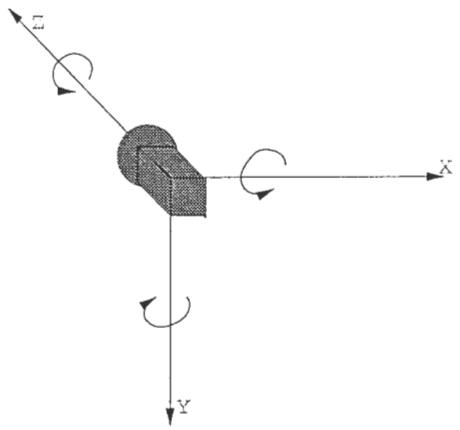


図 8: カメラの回転

回転前の座標を（X,Y,Z）、各軸回転後の座標（X',Y',Z'）は回転行列Rを用いると次のようになる。

$$\begin{pmatrix} \mathrm{X'} \\ \mathrm{Y'} \\ \mathrm{Z'} \end{pmatrix} = R \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{7}$$

X軸回りの回転角をα、Y軸回りの回転角をβ、Z軸回りの回転角をγとすると回転マトリックスRは次のようになる。

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{8}$$

$$= \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{9}$$

$$= \begin{pmatrix} \cos\gamma\cos\beta & \sin\gamma\cos\beta & -\sin\beta \\ -\sin\gamma\cos\alpha + \cos\gamma\sin\beta\sin\alpha & \cos\gamma\cos\alpha + \sin\gamma\sin\beta\sin\alpha & \cos\beta\sin\alpha \\ \sin\gamma\sin\alpha + \cos\gamma\sin\beta\cos\alpha & -\cos\gamma\sin\alpha + \sin\gamma\sin\beta\cos\alpha & \cos\beta\cos\alpha \end{pmatrix} \tag{10}$$

座標変換前の画像面を（x,y）、変換後の画像面を（x', y'）とする。変換後の座標（X', Y', Z'）で投影を考えると、次式が成り立つ。

$$x' = F\frac{X'}{Z'} \tag{11}$$

$$y' = F\frac{Y'}{Z'} \tag{12}$$

以上より、次式が導かれる。

$$x' = F\frac{r_{11}X + r_{12}Y + r_{13}Z}{r_{31}X + r_{32}Y + r_{33}Z} \tag{13}$$

$$y' = F\frac{r_{21}X + r_{22}Y + r_{23}Z}{r_{31}X + r_{32}Y + r_{33}Z} \tag{14}$$

式 (1)(2) を代入すると

$$x' = F\frac{r_{11}x + r_{12}y + r_{13}F}{r_{31}x + r_{32}y + r_{33}F} \tag{15}$$

$$y' = F\frac{r_{21}x + r_{22}y + r_{23}F}{r_{31}x + r_{32}y + r_{33}F} \tag{16}$$

1.2.7 画像の拡大縮小

画像の拡大縮小は、焦点距離を変更することと同じである。つまり、F を F' に変更することで実現が可能である。焦点距離を直接指定するのはわかりにくいのでズーム量 $f = F'/F$ を定義すると、回転後の画像は以下のように表される。

$$x' = fF\frac{r_{11}x + r_{12}y + r_{13}F}{r_{31}x + r_{32}y + r_{33}F} \tag{17}$$

$$y' = fF\frac{r_{21}x + r_{22}y + r_{23}F}{r_{31}x + r_{32}y + r_{33}F} \tag{18}$$

1.2.8 マッピング

forward-mapping フォワードマッピングは、位置が確定された出力イメージの上へ各々の入力画素をマッピング関数によって複写することから成り立つ。

図 9は 1-D の場合のフォワードマッピングを表している。離散の入力と出力はそれぞれ、整数格子（点）の画素の列として表される。各々の入力画素は、空間変換によって出力上の座標値が割り当てられる。ここで、整数集合である入力画素がから実数の集合へマップされることに注意する。

マッピング関数によって与えられた出力位置の値は実数のため、離散の出力上で問題を与える。

画素が点として見られる連続的な領域でマッピングは簡単である。

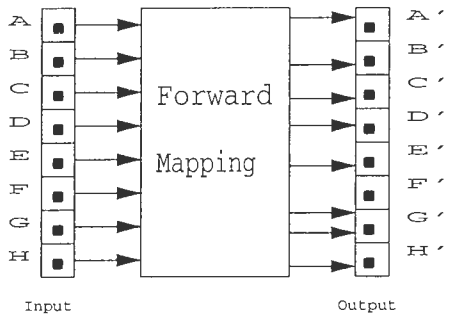


図 9: Forward Mapping

しかし、離散領域中の画素は、整数格子上に位置するように定義された。従って、点から点へのマッピングに空間変換を使用することは妥当ではない。ホールとパッチの２つの問題が持ち上がる。入力サンプルから出力格子上近くのまばらな位置へのマッピングのとき、ホールか画素のパッチが起こる。図でF’は入出力マッピングで、無視されたためのホールである。図でG’は連続的な入力サンプルが１つの出力画素に重なってしまったパッチである。

点から点へのマッピングの問題が、four-corner-mapping パラダイムを使うことによって避けられる。これは出力イメージの任意の四辺形に変換される正方パッチを入力画素とみなす。これはマッピング 後に入力の接触を許す効果がある。写映された入力は出力画像でどこでも自由に位置することができる。入力画素は数個の出力画素にまたがるか、埋まってしまう状態になる。これらの２例は、図10で説明される。受け入れる配列は、各々の出力画素で入力の分割を適切に統合する必要がある。入力破片を各々の出力画素にどのように分割するか決め、そして、分割の破片全てを統合する。破片への分割は、それが覆う画素の分数に比例したスケーリングで扱われる。

four-corner-mappingによる変換は、出力イメージの中でホールを避けるのを許す。それでもこのパラダイムは、フォワードマッピングプロセスで2つの問題を与える。1．交差テストは、ウェイトを得るのが必要である。2．拡大図は、いくらかの出力画素上に使用される、追加フィルターを使用することを除いては、同じ入力値の原因になる。

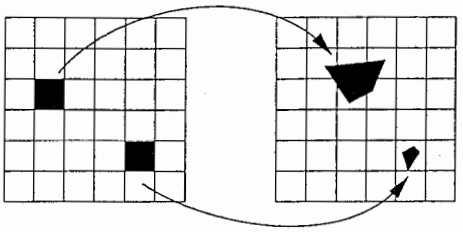


図 10: Accumulator array

両方の問題は、その四辺形のサイズに基づいた入力の適応性サンプリングによって解決できる。もし、入力画素が出力イメージの大きな領域上へマップされるなら、投影した領域が受け入れられる低境界になるまで、入力画素を繰り返して細分されることがベストだi.e. 、1画素サイズ。サンプリング割合が上がると、ウェイトがたった一つの値に収束し、入力はより密としてとられ、計算結果はより高い精度で行なわれる。マッピング関数がアフィン（線型）マッピングである場合を除いては、一様なサンプリング入力イメージが、出力イメージで一様なサンプリングを保証しないことに注目する。従ってアフィンではないマッピングでは、入力イメージは適応的に空間的に変化する割合でサンプリングされなくてはならない。一般にフォワードマッピングは入力イメージが読まれるとき、またはそれがメモリーに完全に存在しないとき使われる。

inverse-mapping インバースマッピングは、マッピング関数によってそれぞれの出力座標を入力イメージに投影する処理を行う。アキュムレーター配列が必要でないとき、そして切り抜き窓の外側の出力画素が評価を必要としないとき、最もありふれた方法である。この方法は、入力イメージがメモリーに完全に蓄えられるとき使われる。

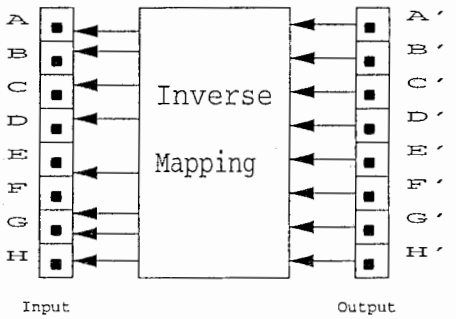


図 11: Inverse Mapping

図11は、インバースマッピングを表現する。それぞれの出力画素は空間変換（インバースマッピング関数）によって入力上にマップバックされる。出力画素が整数座標値に集中することに注意する。それらは入

力表面上の実数位置に写映される。補間段階は、定義されていない非整数の入力位置から入力値を取り戻すため導入される。

点から点へのフォーワードマッピング計画とは違って、インバースマッピングは全出力画素が計算される。しかしながら類似問題は、入力をサンプリングしているとき大きいホールが残っているかどうかである。この場合、入力データの大きな量は、出力を評価している間捨てられる。従ってフィルタリングは、入力表面に写映された領域を統合するために必要である。このアレンジは、出力スペースの代わりの入力スペースで起こすための補間を許すことに利点がある。これはフォワードマッピングより多くの便利なアプローチであると分かる。図でこれは図の入力と出力の図の表題が逆になった2つに等しい

supersampling (oversampling) それぞれの出力ピクセルは数値を、それぞれのプレイメージの効果を減じたサンプルの平均の重みの計算によって求められる。

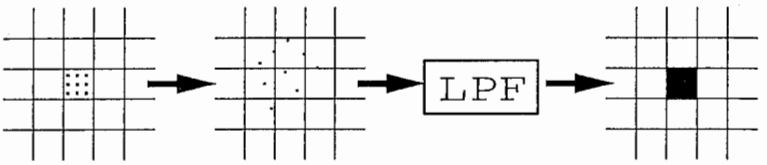


図 12: super sampling

例えば、スーパーサンプリンググリッドが、出力グリッドの3倍の密度ならば、（1ピクセルに9個のグリッド点）それぞれの出力ピクセルは、入力イメージでその投影の効果を減じた9個のサンプルの平均となる（図12）。

例えば、3つのサンプルが緑のオブジェクトにヒットし、6つのサンプルが青のオブジェクトにヒットするなら、ボックスフィルターが使われると仮定すると、出力ピクセルの合成色は1／3が緑で2／3が青になる。

スーパーサンプリングは入力信号をバンドリミッティングすることによってエイリアスを減少する。

高分割スーパーサンプリンググリッドの目的は出力イメージでみられる前イメージの評価を細かに区別することだ。

1.2.9 多眼カメラにおけるマッピングについて

M_{oi} センターカメラから周辺カメラへのマッピングを考える。

Center Camera の移動 を $C_o(C_{xo}, C_{yo}, C_{zo})$

Center Camera の回転 を R_o

Inspection Camera の移動 を $C_i(C_{xi}, C_{yi}, C_{zi})$

Inspection Camera の回転 を R_i

World coordinate system を X_w

Center Camera coordinate system を X_o

とすると、

$$X_o = R_o X_w + C_o \tag{19}$$

となり、

$$X_w = R_o^{-1}(X_o - C_o) \tag{20}$$

が導かれる。また、

Inspection Camera coordinate system を X_i とすると、

$$\begin{aligned} X_i &= R_i X_w + C_i && (21)\\ &= R_i R_o^{-1} X_o - R_i R_o^{-1} C_o + C_i && (22)\\ &= R_{oi} X_o + T_{oi} && (23) \end{aligned}$$

となる。

Center Camera Image coord は、

$$x_o = F_o \frac{X_o}{Z_o} \tag{24}$$

$$y_o = F_o \frac{Y_o}{Z_o} \tag{25}$$

と表されることから、

Inspection Camera Image coord は、

$$\begin{aligned} x_i &= F_i \frac{X_i}{Z_i} && (26)\\ &= F_i \frac{r_{11}X_o + r_{12}Y_o + r_{13}Z_o + T_{xoi}}{r_{31}X_o + r_{32}Y_o + r_{33}Z_o + T_{zoi}} && (27)\\ &= F_i \frac{r_{11}x_o + r_{12}y_o + r_{13}F_o + T_{xoi}F_o/Z_o}{r_{31}x_o + r_{32}y_o + r_{33}F_o + T_{zoi}F_o/Z_o} && (28) \end{aligned}$$

$$\begin{aligned} y_i &= F_i \frac{Y_i}{Z_i} && (29)\\ &= F_i \frac{r_{21}X_o + r_{22}Y_o + r_{23}Z_o + T_{yoi}}{r_{31}X_o + r_{32}Y_o + r_{33}Z_o + T_{zoi}} && (30)\\ &= F_i \frac{r_{21}x_o + r_{22}y_o + r_{23}F_o + T_{yoi}F_o/Z_o}{r_{31}x_o + r_{32}y_o + r_{33}F_o + T_{zoi}F_o/Z_o} && (31) \end{aligned}$$

M_{vi}　仮想カメラから周辺カメラへのマッピングを考える。

Virtual Camera coordinate system を X_v とすると

$$X_v = X_o - V \tag{32}$$

となるので

$$X_o = X_v + V \tag{33}$$

が導かれる。

Inspection Camera coordinate system を X_i とすると

$$\begin{aligned} X_i &= R_{oi}X_o + T_{oi} && (34)\\ &= R_{oi}(X_v + V) + T_{oi} && (35)\\ &= R_{oi}X_v + R_{oi}V + T_{oi} && (36)\\ &= R_{vi}X_v + T_{vi} && (37) \end{aligned}$$

となる。

よって、Inspection Camera Image coord は、

$$\begin{aligned} x_i &= F_i\frac{X_i}{Z_i} && (38)\\ &= F_i\frac{r_{11}X_v + r_{12}Y_v + r_{13}Z_v + T_{xoi}}{r_{31}X_v + r_{32}Y_v + r_{33}Z_v + T_{zvi}} && (39)\\ &= F_i\frac{r_{11}x_v + r_{12}y_v + r_{13}F_v + T_{xoi}F_v/Z_v}{r_{31}x_v + r_{32}y_v + r_{33}F_v + T_{zvi}F_v/Z_v} && (40) \end{aligned}$$

$$\begin{aligned} y_i &= F_i\frac{Y_i}{Z_i} && (41)\\ &= F_i\frac{r_{21}X_v + r_{22}Y_v + r_{23}Z_v + T_{yoi}}{r_{31}X_v + r_{32}Y_v + r_{33}Z_v + T_{zvi}} && (42)\\ &= F_i\frac{r_{21}x_v + r_{22}y_v + r_{23}F_v + T_{yoi}F_v/Z_v}{r_{31}x_v + r_{32}y_v + r_{33}F_v + T_{zvi}F_v/Z_v} && (43) \end{aligned}$$

2　多視点画像と奥行きマップを用いた任意視点ステレオ映像の生成

2.1　はじめに

本研究のベースは、「数枚の実画像」と「ステレオマッチングによって得られる奥行き情報（奥行きマップ）」を用いて、任意視点映像を生成することである。これまでは主に単眼映像に対しての任意視点映像生成が行われてきた。本研究ではステレオ映像の生成を行い、実写映像だけによるバーチャル空間を構築し、ＨＭＤなどでその空間を体験したとき、より３次元的な臨場感の溢れる映像表現を試みる。

2.2　本研究の概念、および手法

2.2.1　ステレオ映像生成の概念

単眼の場合の奥行き感は、運動視差のみに頼っていた。しかし、人間が最も奥行き感を感じるのは両眼視差である。３次元の物体を眺めた場合、左右両眼の位置が眼間距離だけ離れているため透視変換に違いが生じ、それぞれの網膜に異なった像が映る。この違いが大脳視覚野に送られ、比較されて奥行き感が生成される。従って、奥行き感がある映像を生成するためには左右両眼に、視差に応じた別々の映像を供給してやればよい。
単眼の場合と同様、撮影は５台のカメラを用いて行う。その後、実際のカメラ位置以外に両眼の距離だけ離した２台の仮想カメラを想定し、それらの仮想カメラから得られるであろう映像を５台のカメラで撮影した映像と奥行きマップから生成する。（図 13）

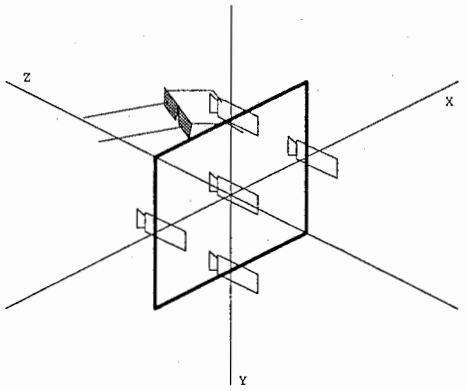


図 13: 任意視点ステレオ映像生成の概念

2.2.2　ＨＭＤについて

ＨＭＤ (Head Mounted Display) とは、ユーザーの眼球直前にレンズを介して置かれた小型の映像表示装置に、頭の向いている方向の映像を常に供給するようにされたディスプレイである。人間は常に自分の周囲３６０度を見ているわけではなく、頭の向いている前方しか見ていない。つまり、見えている部分を眼球に供給することにより、全天周スクリーンを等価的に作り出したことになる。

2.2.3 任意視点の配置

任意視点の初期配置は、次の通りである（図 14）。

- x 軸上に間隔 L で、2 つの任意視点間の中点が原点となるようにおく。
- y 軸の値が 0 となるようにおく。
- z 軸の値が 0 となるようにおき、z 軸方向に視線方向をとる。

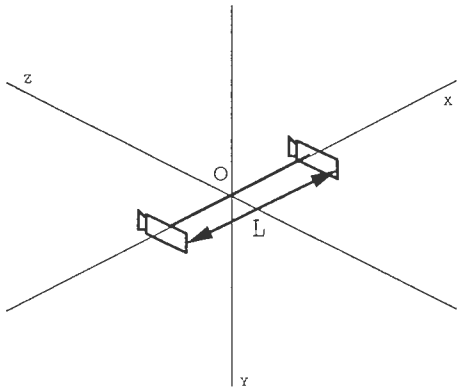


図 14: 任意視点の初期配置

2 つの任意視点をそれぞれ C_r, C_l とすると、任意視点の初期配置は次式のようにあらわされる。

$$C_r : (x, y, z) = (\frac{L}{2}, 0, 0) \tag{44}$$

$$C_l : (x, y, z) = (-\frac{L}{2}, 0, 0) \tag{45}$$

2.2.4 視点の動作処理

それぞれ2つの視点に対して視点の移動及び回転の計算を行ったのでは計算量が増えてしまう。そこで、任意視点間の中点 C_c が移動するとして考える。そうすることにより、単眼の場合の計算と同じにすることができる。計算により C_c の座標が決定したならば、2つの任意視点の座標を容易に求めることができる。

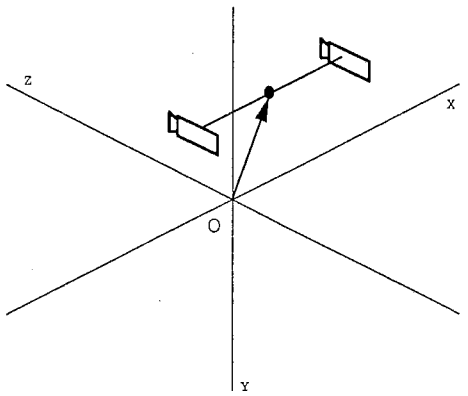


図 15: 任意視点の移動

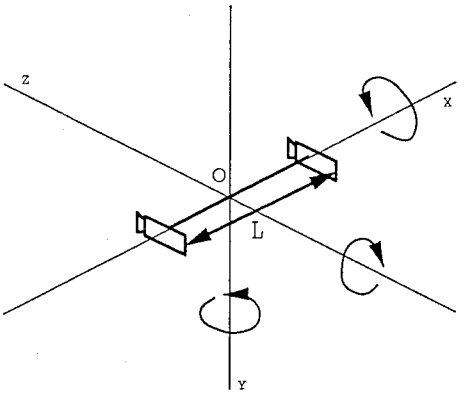


図 16: 任意視点の回転

視点の移動で視線方向に対する変化は全くなく、2つの任意視点は平行移動するものとする。(図 15) 移動後の C_c の座標を $C_c = (x, y, z) = (p, q, r)$ とすると、移動後の2つの任意視点 $C_{r'}$, $C_{l'}$ は次式で表される。

$$C_{r'} : (x, y, z) = (p, q, r) + (\frac{L}{2}, 0, 0) = (p + \frac{L}{2}, q, r) \tag{46}$$

$$C_{l'} : (x, y, z) = (p, q, r) + (-\frac{L}{2}, 0, 0) = (p - \frac{L}{2}, q, r) \tag{47}$$

また、視点の回転は、任意視点間の中点を中心とする回転を考える。左右それぞれの任意視点の回転は、次式で示される。(図 16)

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} \frac{L}{2} \\ 0 \\ 0 \end{pmatrix} \tag{48}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} -\frac{L}{2} \\ 0 \\ 0 \end{pmatrix} \tag{49}$$

direction_cosine を用いた回転マトリックスの異なる表記　任意視点がある 1 点を見続けながら回転動作を行う場合、その時の回転マトリックスは次で表される（図 17）。

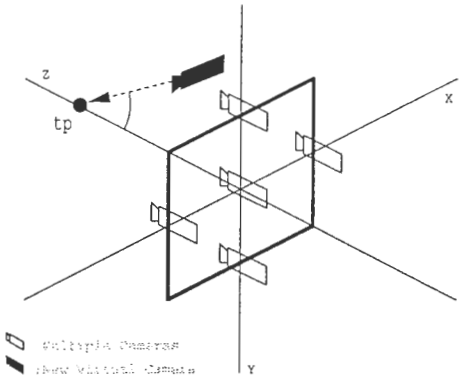


図 17: ある 1 点を見続けながらの回転動作

$$\begin{pmatrix} n_1^2+(1-n_1^2)\cos\theta & n_1n_2(1-\cos\theta)-n_3\sin\theta & n_1n_3(1-\cos\theta)+n_2\sin\theta \\ n_1n_2(1-\cos\theta)+n_3\sin\theta & n_2^2+(1-{n_2}^2)\cos\theta & n_2n_3(1-\cos\theta)-n_1\sin\theta \\ n_1n_3(1-\cos\theta)-n_2\sin\theta & n_2n_3(1-\cos\theta)+n_1\sin\theta & {n_3}^2+(1-{n_3}^2)\cos\theta \end{pmatrix} \tag{50}$$

このマトリックスは *direction cosine* と呼ばれるもので、あるベクトル (n_1,n_2,n_3) を中心に θ だけ回転を行うものである。（図 18）。

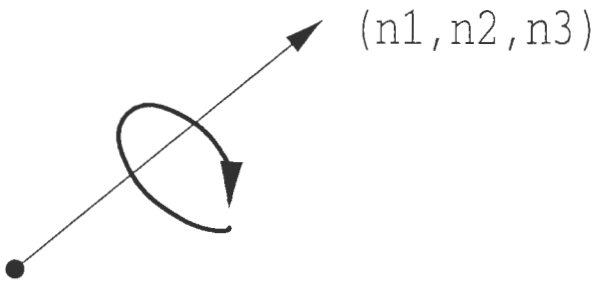
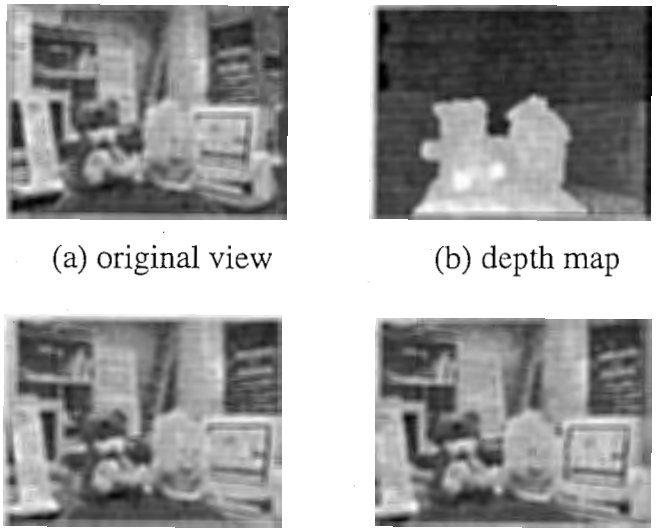


図 18: direction_cosine

2.3 実験

実験で得られた任意視点映像の、あるフレームを図19に示す。2つの画像に両眼視差が生じたことがわかる。実験条件を表1に示す。



(a) original view　(b) depth map

(c) left view　(d) right view

図 19: 実験結果

表 1: 実験条件

基準点の位置	x= 16.63[mm]
	y= 5.23[mm]
	z=-242.35[mm]
両眼距離	60[mm]
基線長 L	50[mm]
F	7.5[mm]
CCD	1/2”
tp	x=0[mm]
	y=0[mm]
	z=900.00[mm]

2.4 考察

実験に用いた視点の軌跡は次式であらわされる。

移動範囲が大きいシーケンス

$$x = \sin(\theta \times \pi/15) * 80.0 \quad (51)$$

$$y = \sin(\theta \times \pi/30) * 50.0 \quad (52)$$

$$z(A) = (1.0 - \cos(\theta \times \pi/22.5)) * 150.0 \quad (53)$$

$$z(B) = (\cos((\theta \times \pi/22.5) - 1.0) * 150.0 \quad (54)$$

移動範囲が小さいシーケンス

$$x = \sin(\theta \times \pi/15) * 80.0/5.0 \quad (55)$$

$$y = \sin(\theta \times \pi/30) * 50.0/5.0 \quad (56)$$

$$z(A) = (1.0 - \cos(\theta \times \pi/22.5)) * 150.0 \quad (57)$$

$$z(B) = (\cos((\theta \times \pi/22.5) - 1.0) * 150.0 \quad (58)$$

- A:frame no.000 から frame no.045 θ を 1 づつ増加
- B:frame no.046 から frame no.090 θ を 1 づつ減少

実際に構築した空間を体験する実験

実験は偏光グラスを用いて行った。この実験により両眼距離だけ離れた映像生成を実現でき、奥行き感が増加したことが確認できた。

移動範囲が小さいシーケンスでは、単眼の場合より鮮明な映像であると感じた。移動範囲が大きいシーケンスでは、多眼カメラからは測定できない領域で誤差が生じ違和感が感じられた。（図 20）。

図 20: マッピングの誤差

3　カメラの動作に合わせた任意視点映像の生成と実時間合成

3.1　はじめに

これまでの本手法に基づく任意視点背景映像生成に関する研究は、静止している物体にを多眼カメラで撮影し、奥行きマップの取得と任意視点映像の生成を行いこれらを背景映像にして、スタジオの固定カメラによって撮影した対象物の映像と3次元合成していた。本テーマでは、静止したカメラによる撮影を可動のカメラに対応できるように拡張し、カメラ操作にあわせて背景映像を生成し3次元合成することで、よりリアルな仮想空間の構築を試みる。

3.2　本研究の概念及び手法

3.2.1　IE-Room（イメージ表現ルーム）について

COMI&CSでは、メディア操作機器をコンピュータで統一的に管理、制御することにより、専門家でないユーザでも簡単に操作し、種々のメディアを自由に組み合わせてイメージを表現できる機能を提供する。

COMI&CSの概念を図21に示す。COMI&CSにおいては、通常の映像や音だけでなく、映像や音を構成する、あるいは、映像や音に付随する情報、例えば、形状や奥行きなどの3次元情報、テキスチャ、動き情報などをマルチメディア部品としてデータベースに蓄積しておき、作品生成時には、それらを自由に組合せて出力する。その際、専門家でないユーザが特別の知識なしにシステムを利用できることを意図し、入出力メディアの相互接続により簡単に作品が生成できるよう工夫している。

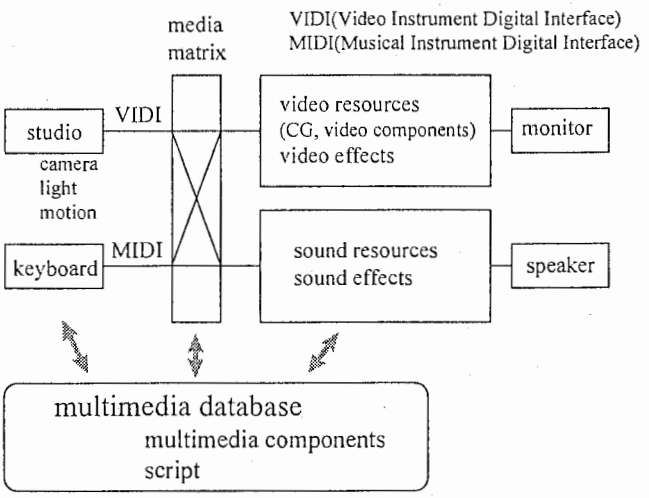


図 21: Concept of the COMI&CS.

システムのハードウェア構成を図22に示す。システムはクロマキースタジオを中心にユーザが合成映像を確認できるよう、ハーフミラー付きの大画面スクリーンを配置している。これによりユーザはあたかも鏡を見るかのように、仮想シーン内での自分の姿を確認できる。

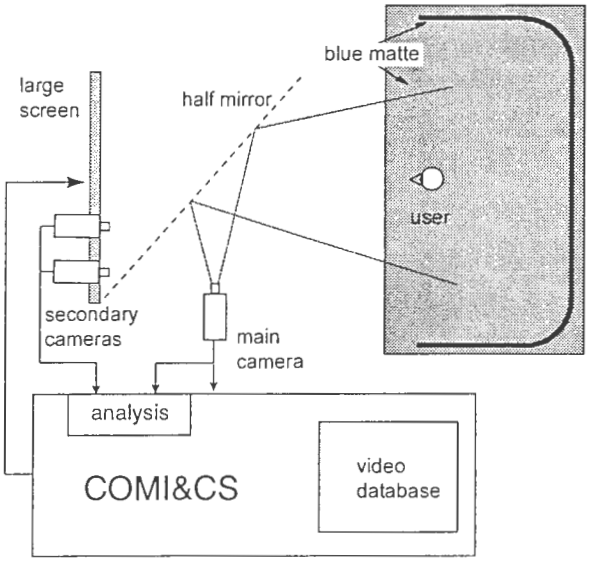


図 22: The Image Expression Room.

ユーザの動作や反応も一つのメディアと考える。認識処理部では、映像中のユーザ領域の位置や大きさ、ユーザの動作や反応などの認識を行い、入力メディアとして利用する。

合成は、既存のスイッチャやDVE(Degital Video Effect)装置をコンピュータコントロールすることにより行う。スイッチャの選択、特殊効果のパラメータ変更などを、ユーザの動作、反応といった入力メディアに従い、インタラクティブかつリアルタイムに行うことが可能である。

合成用素材映像として、映像データベースからの出力、および、CG ワークステーションで生成されるリアルタイム CG を利用する。マルチメディアデータベースは、映像用 MPEG 2 コデック、大容量ディスク (100GB 程度)、MT ラック (1TB 程度) などから構成される。

3.2.2 Z-key 映像合成法

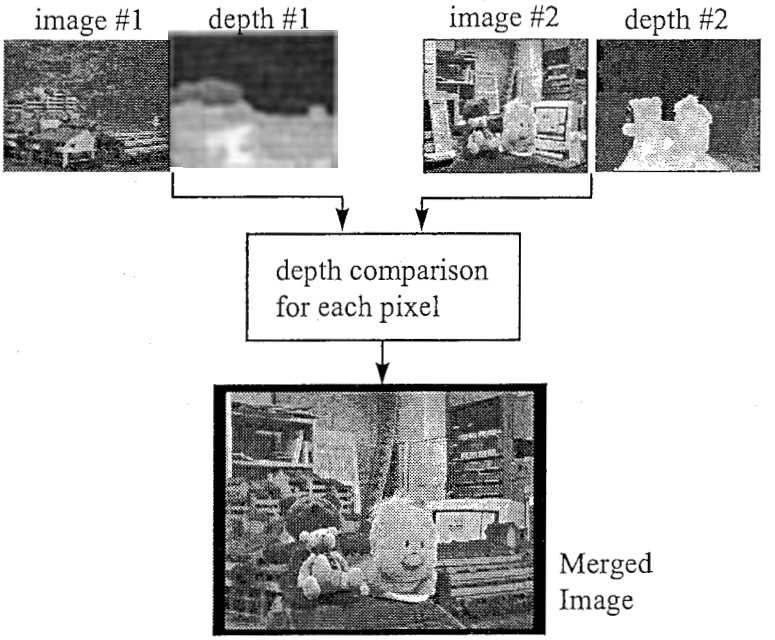


図 23: Illustration of Z-key composition.

合成の対象となる複数の映像とその奥行きマップをまず用意する。画面全体にかけて、画素ごとに、複数の映像の奥行き値を比較し、奥行き値の小さいもの、つまり、カメラから近いものが表示されるように画素値を選択する。Z-key 法による映像合成の例を図 23に示す。合成結果をみると、あたかも二つの空間がマージしているかのようにみえる。このように、複雑そうに思われる空間的マージ処理を非常に簡単な処理だけで済ませるということが Z-key 法の強力なところである。さらに、Z-key 法に色々な映像処理技術を組み合わせると実に多様なイメージ表現が実現できる。例えば、ある被写体を別のものに置き換えたり、ある物体だけ特殊効果をかけたりすることが容易にできる。

3.2.3 カメラ操作時の映像合成

IE Room において、カメラを操作する場合の映像表現の概念図を図 24に示す。カメラの操作に応じて仮想世界の視点を変えることにより、自然な映像合成を図るものである。

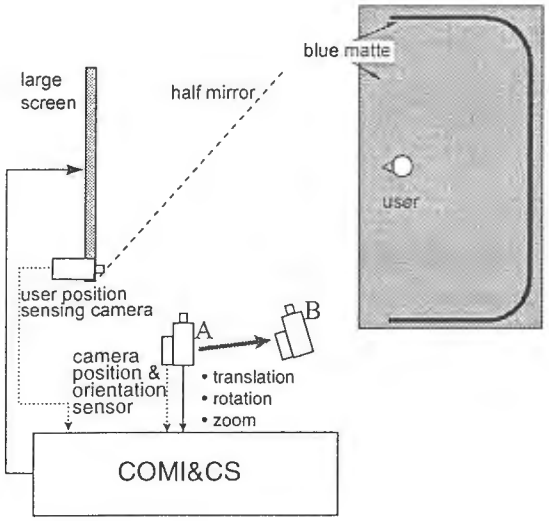


図 24: Configuration of the IE Room with camera motion.

3 次元映像合成の処理の流れを図 25に示す。

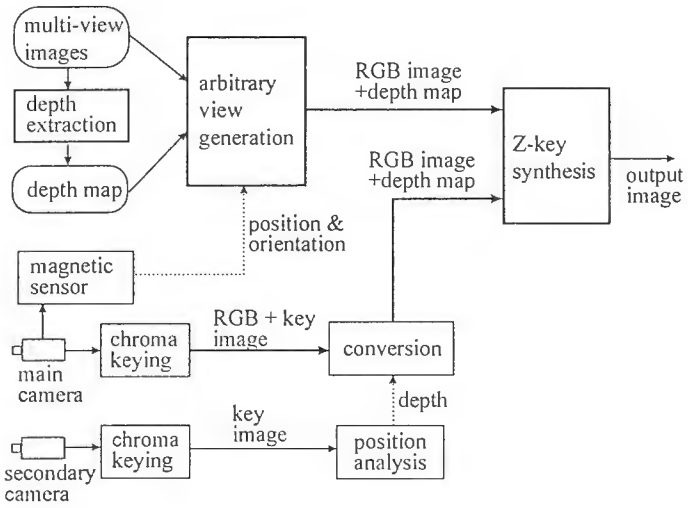


図 25: Conceptual flow diagram of 3D composition in the IE Room.

3.2.4 システム構成

カメラに磁気センサーをつけてカメラ位置と方向を取得し、その値に合わせた仮想空間を生成する。しかし、磁気センサーからのデータを用いてリアルタイムにカメラの位置と方向から見た仮想空間を生成することは、現状のハードウェアではできない。そこで、任意視点が動く範囲を規則的にサンプリングし、その位置での映像をあらかじめ計算して任意視点映像を生成し、フレームメモリに蓄積しておくことにする。それから、スタジオからの映像との合成を行うときに、カメラ位置の最近傍の映像をフレームメモリから取り出して合成を行うことにする。

3.2.5 磁気センサー

センサーシステムのなかで、現実空間内の1点の位置情報を計測するものがある。よく使用されるのが、磁界を利用した空間位置センサーである（図26）。

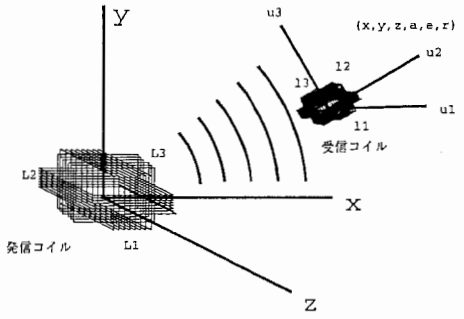


図 26: 磁気センサーの仕組み

これは、、磁界の変化によってコイルに起電力を生ずるという原理を使用している。図において、大コイル L_1 に流れる電流を変化させると、小コイル l_1, l_2, l_3 を切る磁束が変化する。各磁束変化は、小コイルと L_1 との位置関係によって決まるので、小コイルに生じる起電力 $v_i(i = 1, 2, 3)$ は、

$$v_i = v_i(x, y, z, \theta, \phi, \omega) \tag{59}$$

で表すことができる。x,y,z,θ,ϕ,ω は、大コイルから見た小コイル系の位置と姿勢である。

L_1 と同様に L_2、L_3 を次々励磁すると、3種類の磁界が発生する。よって、9組のデータを得ることが出来る。

$$v_{ij} = v_{ij}(x, y, z, \theta, \phi, \omega) \tag{60}$$

9元連立方程式を解くことによって、x,y,z,θ,ϕ,ω を逆推定してやることが出来る。

3.2.6 視点の回転角の導出

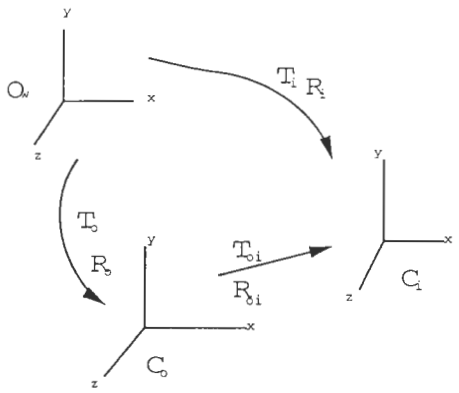


図 27: 座標系の変換

任意の座標系 O_w を考える（図 27）。座標系 O_w から座標系 C_o への移動が R_o, T_o で表される。また、座標系 O_w から座標系 C_i への移動が R_i, T_i で表される。このとき、C_o から C_i への座標系の移動 R_{oi}, T_{oi} を求めると。

$$T_{oi} = T_i - T_o \tag{61}$$

$$R_i = R_{oi} R_o \tag{62}$$

$$R_i R_o^{-1} = R_{oi} R_o R_o^{-1} \tag{63}$$

$$R_{oi} = R_i R_o^{-1} \tag{64}$$

R_{oi} を用いて、カメラ（磁気センサー）の方向の回転角度を算出している。

3.3 実験

カメラ移動方向をｘ軸上及びｙ軸上に限定したものをシステムＡとし、システム図を以下（図 28）に示す。

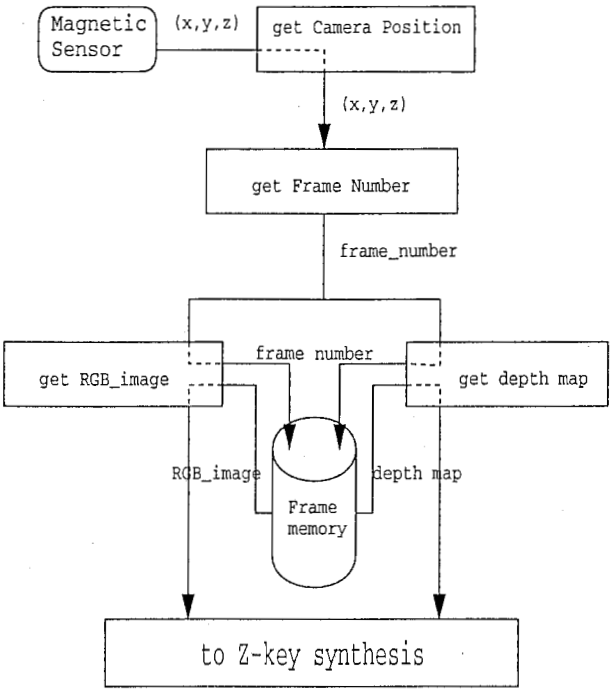


図 28: システムＡ

合成の結果を以下（図 29, 図 30）に示す。位置を検出して、フレームを変化することができている。この場合はカメラが下方向に移動。

図 29: 実験結果 1 (a)

図 30: 実験結果 1 (b)

カメラ移動及び回転の制限がないものをシステムBとし、システム図を以下（図31）に示す。

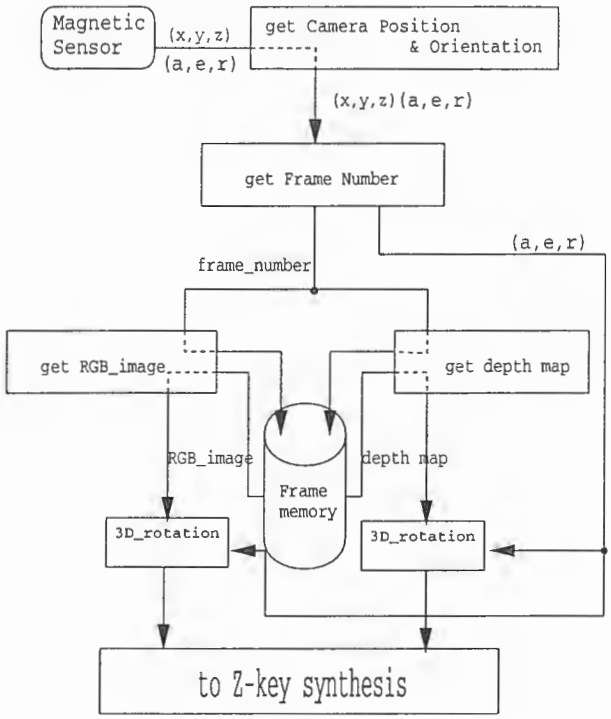


図 31: システムB

合成の結果を以下（図32, 図33）に示す。フレームの回転が実現できたことがわかる。

図 32: 実験結果2 (a)

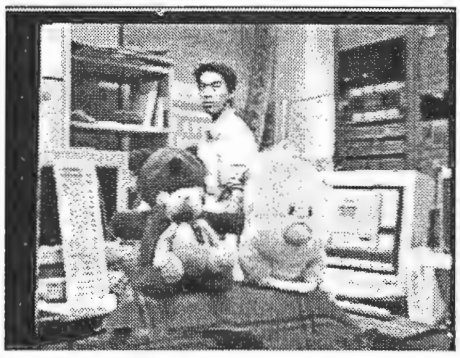


図 33: 実験結果2 (b)

3.4 考察

システムAについて

実験の結果、スタジオ映像と任意視点背景映像とで違和感を感じた。これは、フレームメモリに蓄積した映像のカメラ移動に対するサンプリング間隔が荒いためである。通常、映像は1秒間に30フレームと言われるが、実験で生成した背景映像は1秒間に約2フレーム程度であった。スタジオの映像がスムーズであるのに対し、背景映像はストロボ映像の様であるため、違和感を感じたのだと考えられる。フレームレートの問題を除けば、期待されるシステムを実現できた。

システムBについて

実験の結果、スタジオ映像と奥行きマップとイメージ映像との間に、時間差が生じてしまった。この時間差の原因を考える。

本システムは、3台のコンピュータを使用している。まず、磁気センサーによって現在のカメラ位置及び方向のデータを取得する。このデータが1台目のコンピュータ（コンピュータA）におくられ、位置データからフレームナンバーを取得する。フレームナンバーと方向のデータが同時に2台のコンピュータに送れればよいのだが、送信の関係で、まず1台に送ることになった（コンピュータB）。なるべく時間差をなくすように、コンピュータBはデータを受信するとすぐに残りのコンピュータ（コンピュータC）に送信するようにした。コンピュータBとコンピュータCはデータを受信するとそれぞれイメージフレーム及び、奥行きフレームをフレームメモリから取得してくる。取得したフレームは、Open-GLテクスチャとしてはりつけられ、方向データによって回転処理が行われる。その後、合成される。

以上が合成処理の流れである。ここで、時間差が生じる理由が2つあげられる。
1つめは、データの送受信の部分である。受信後すぐに送信を行うようにしてはいるが、やはりコンピュータBとコンピュータCの間には時間差が生じてしまう。
2つめは、Open-GLである。回転の処理を行うためにOpen-GLを使用した。コンピュータBについては、ライブテクスチャ機能があるため問題ないが、コンピュータCはライブテクスチャ機能が映像をテクスチャとして使用するのにかなりの時間がかかってしまう。ここが、時間差が生じる一番の原因と考えられる。また、コンピュータB及びコンピュータCは別の機種を使用している。そのためCPUの処理速度の違いの影響も考えられる。

4　まとめ

4.1　おわりに

任意視点ステレオ映像生成について

多眼カメラを使って奥行きを推定し、それに基づいて任意視点から見たステレオ映像を生成する手法を提案した。これにより、両眼視差による奥行き感が向上できたことが確認できた。

3次元映像合成法（システムA）について

ユーザーのカメラ操作に合わせて任意視点映像を生成し3次元合成するアプローチとして、カメラ操作を上下左右の方向に限定した場合の任意視点背景映像の生成を試みた。フレームメモリに蓄積したフレームのサンプリング間隔が荒いため、違和感を感じた。フレームレートの問題を除けば、期待のシステムは実現できた。しかしカメラの回転が許容されないため、応用性は限定されると考える。

3次元映像合成法（システムB）について

ユーザーのカメラ操作に合わせて任意視点映像を生成し3次元合成を試みた。実験の結果、カメラ移動や回転の原理は正しいことが確認できた。使用したコンピュータの種類の違い、コンピュータ間のデータ通信の時間差、テクスチャの回転処理の影響のため、スタジオ映像と奥行きマップとイメージとで時間差が生じてしまった。今後は、任意視点映像のリアルタイム生成や任意視点映像生成専用の環境の構築に向けて研究の必要がある。

・謝辞

実務訓練期間中、多大な御指導をしていただいた朴 鍾一研究員に心から感謝いたします。また、数々の御助言ならびに支援していただいた井上 誠喜室長をはじめとする知能映像通信研究所第3研究室の方々に御礼申し上げます。特に、多大なご助言をいただいた石若 通利 研究員、古屋 隆志 氏には深く感謝いたします。最後に、私に貴重な体験の場を提供してくださったＡＴＲ知能映像通信研究所ならびに長岡技術科学大学の方々にこの場をかりて御礼を申し上げます。

参考文献

[1] 朴 鍾一, 井上 誠喜, "多視点映像から任意視点映像の生成" , 電子情報通信学会,IE96-121,1997 年 2 月

[2] Jong-Il Park and Seiki Inoue, "*ArbitraryViewGenerationfromMultipleCameras*" ,Proc.IEEE ICIP'97,Santa Barbara,Oct.1997.

[3] J.Park, N.Yagi, K.Enami, K.Aizawa,and M.Hatori, "*EstimationofCameraParameters fromImageSequenceforModel−BasedVideoCoding*" ,IEEE TOCASFVT,VOL.4,NO.3,JUNE.1994

[4] 朴 鍾一, 大川 慶, 井上 誠喜, "イメージ表現ルームにおける 3 次元映像合成法" , 電子情報通信学会,,1998 年 2 月

[5] 松井 明, 朴 鍾一, 井上 誠喜, "多視点画像と奥行きマップを用いた任意視点映像生成についての検討", ATR Technical Report TR-M-0016, 1997 年 3 月

[6] 青木 利道, 朴 鍾一, 井上 誠喜, "多視点画像と奥行きマップを用いた三次元シーンの再構築", ATR Technical Report TR-M-0029, 1997 年 9 月

プログラムについて

プログラム実行方法 及び 必要ファイル一覧

- 任意視点ステレオ映像生成

```
dir   : miris37&miris69 % /home/ohkawa/vs/vs_stereo/vs_stereo_demo2/
Usage: vss3 imagefilename_list depthfile vpfile mode
```

表 2: 実験で使用したファイル

vss3	
imagefilename_list	kh_list
depthfile	kh.dsp
vpfile	vp_xxx03
mode	2

mode について

mode 1 は、カメラがその位置で回転する映像を生成する。

mode 2 は、注目点を見続けるようにカメラが回転する。

vpfile の形式が mode によって変わるので注意が必要

表 3: 必要ファイル

dmake3.c	視点移動軌跡生成
dmake4.c	視点移動軌跡生成
img_utl.c	画像処理を扱うルーチン
img_utl.h	ヘッダ
nr_utl.c	配列に関するルーチン
nr_utl.h	ヘッダ
osflt.c	そのほかのルーチン
osflt.h	ヘッダ
vs_stereo3.c	メインプログラム

- カメラ動作に合わせた任意視点映像生成と実時間合成 システム A

```
dir  : miris37 % /home/ohkawa/vs/vs_demo/open97_2/
Usage: perl vs_frame_0.pl
```

表 4: 必要ファイル (ONYX)

fastrak.c	磁気センサー用ソース
fastrak.h	ヘッダ
ispudef.h	フレームメモリ用ヘッダ
vddef.h	フレームメモリ用ヘッダ
vs_frame_1.c	データ取得プログラムソース
vs_frame_2.c	イメージフレーム取得プログラムソース
vs_frame_3.c	デプスフレーム取得プログラムソース
vs_frame_0.pl	perl によるプログラム間のデータ転送

- カメラ動作に合わせた任意視点映像生成と実時間合成 システム B

ソケット通信を使用しているため、 accept を先に立ち上げる必要がある。
(以下に示した順に立ち上げていく。)

```
    dir  : miris69 % /home/ohkawa/vs/vs\_tex\_rot/vs\_texrot\_demo\_dep/
1. Usage:accept | dmtexcube

    dir  : miris61 % /home/ohkawa/vs_texrot_demo_img/
2. Usage:accept | screen_INDG2 |connect miris69

    dir  : miris37 % /home/ohkawa/vs/vs_tex_rot/vs_texrot_demo_frak/
3. Usage:vs_dat_send | connect miris61
```

表 5: 必要ファイル (ONYX)

connect.c	ソケット通信 データ送信側
fastrak.c	磁気センサー 用ソース
fastrak.h	ヘッダ
ispudef.h	フレームメモリ用ヘッダ
vddef.h	フレームメモリ用ヘッダ
vs_dat_send.c	データ送信メインプログラム

表 6: 必要ファイル (O2)

accept.c	ソケット通信受信側
ispc.c	フレームメモリ用ソース
ispudef.h	フレームメモリ用ヘッダ
vddef.h	フレームメモリ用ヘッダ
ispc.h	
oglwindow.h	ヘッダファイル
fastrak.h	
myimage.h	
texture.h	
unitsquare.h	
ideas.c++	ソースファイル
oglwindow.c++	
reftex.c++	
tex_cube.c++	
twave.c++	
atlantis.c++	
jplay.c++	
scube.c++	
texture.c++	
unitsquare.c++	
vtex.c++	
olympic.c++	
spots.c++	

表 7: 必要ファイル (INDIGO2)

accept.c	ソケット通信受信側
connect.c	ソケット通信送信側
ispudef.h	フレームメモリ用ヘッダ
vddef.h	フレームメモリ用ヘッダ
screen_INDG2.c	ソースファイル