ATRテクニカルレポート表紙

[非公開]

TR-M-0041

多眼カメラを用いた任意視点現像の生成

千葉　正広
Masahiro TIBA

朴　鍾一
Jong-Il PARK

1999.2.26

ATR 知能映像通信研究所

多眼カメラを用いた任意視点映像の生成

Arbitrary View Generation Using Multiple Cameras

平成 11 年 2 月 26 日

概要

本研究の目的は、自然風景などの実写映像を基に image-based rendering によって任意の視点・視線方向の映像を生成することである。多眼カメラで撮影した映像と奥行きマップを利用して、任意視点の映像を生成する方法について検討した。また、実時間処理実現のために、処理の効率化による計算時間の短縮についても検討した。実験として、ユーザが指定した視点の映像を即時に生成して表示するプログラムを試作した。

ATR 知能映像通信研究所 第 3 研究室

学外実習生 千葉 正広[1]

実習期間：平成 10 年 10 月 12 日－平成 11 年 2 月 26 日

[1]長岡技術科学大学 電気・電子システム工学課程

1　任意視点映像生成の概要

1.1　はじめに

任意視点映像とは、ある視点で撮影した映像を基に創り出した別の視点の映像。言い換えると、仮想的なカメラを設置し、あたかもそのカメラで撮影したかのように合成された映像のことである。

本研究の目的は、多眼カメラ映像と奥行きマップだけを用いたimage-based renderingにより任意視点映像を生成する手法を検討することと、処理の効率化により計算時間を短縮する方法について検討することである。

1.2　任意視点映像の生成

1.2.1　image-based renderingによる任意視点映像生成

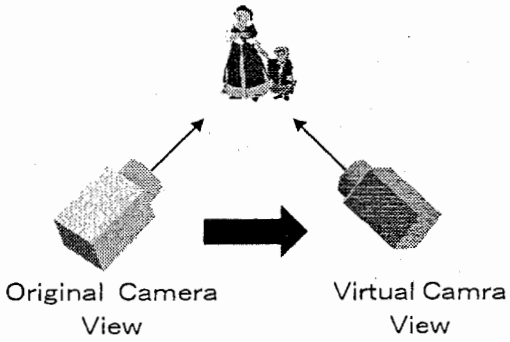


図 1: 任意視点映像生成の概念

任意視点映像生成の概念を図1に示す。

ＣＧなどで任意の視点の映像を生成する場合、被写体に関する3次元モデルを取得し、モデルベースでレンダリングを行うのが一般的である。しかし、自然風景などの実写映像から被写体に関する3次元モデルを抽出するのは非常に困難であり、モデルベースのレンダリングは難しい。これにたいし本研究では、多眼カメラで撮影した多視点の映像と、これらの映像から奥行き情報を抽出した奥行きマップのみを利用し、image-based rendering によって任意視点映像を生成する手法を検討する。3次元モデルを必要としないイメージベースのレンダリング処理を行うことによって、実写映像に基づいた任意視点映像を生成できるのである。

1.2.2　多眼カメラシステム

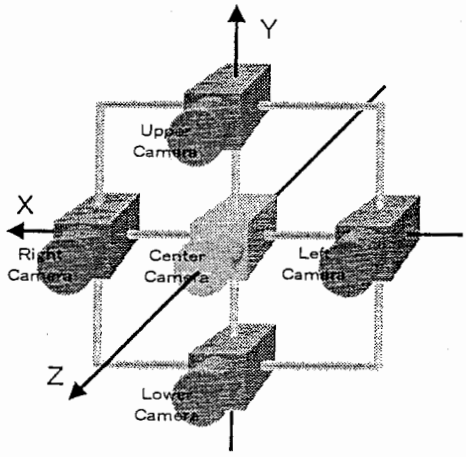


図 2: 多眼カメラシステム

本研究で使用した多眼カメラシステムは図2のようである。中央に基準カメラを設置し、その周りに参照カメラを設置する。カメラキャリブレーション [1] を行うことにより、参照カメラの配置情報は厳密に取得することができるので、参照カメラの位置・角度はある程度[2]自由である。

また、動画像へ対応するために、全てのカメラは同期させる。

1.2.3 奥行きマップ

本手法で利用する奥行きマップは“密”かつ“シャープ”なものでなければならない。(図3) ここで、“密”とは画面全体すべての画素について奥行き値があること、“シャープ”はオブジェクトの境界が明確であることを、それぞれ意味する。本手法では、朴らが提案する奥行き抽出法 [2][3] による奥行きマップを採用している。これは、多眼カメラシステムを利用した奥行き抽出法であり、階層的な推定によるオブジェクト境界でのマッチング精度の向上、多眼カメラによるオクルージョンへの対策を実現している。

図 3: “密”で“シャープ”な奥行きマップ

1.2.4 カメラの移動とオクルージョン領域

視点すなわち仮想カメラの移動に伴い、画面内では図4のような現象が起こる。遠くのものはあまり動かず、近くのものは大きく動く。また、奥のオブジェクトが手前のオブジェクトに隠されたり (隠される領域 occluded area)、今まで見えなかったオブジェクトが新たに見えてきたり (現れる領域 uncovered area) する。

これらの現象を image-based rendering によって再現するのが、本研究のアプローチである。

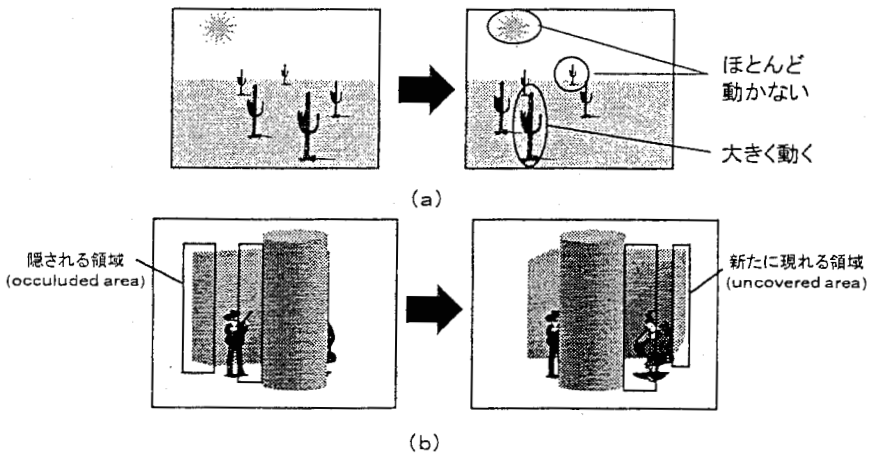


図 4: カメラの移動に伴い画面内で起こる現象

[2]奥行きマップ抽出・任意視点映像生成の際にオクルージョン問題が解決できる範囲内

2 任意視点映像生成に関する基礎

2.1 カメラの移動と3次元座標変換

2.1.1 空間座標から平面座標への投影

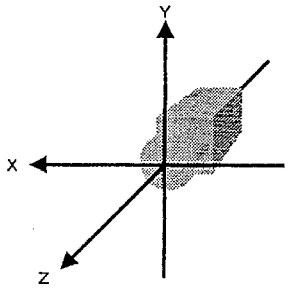


図 5: カメラ中心座標系

まず、カメラ中心の座標系 $\boldsymbol{X}$ を考える。(図 5) カメラの焦点距離を F とすると、カメラ座標系 $\boldsymbol{X}$ の点 $P(X,Y,Z)$ は、撮像平面上の点 $p(x,y)$ に投影される。

$$x = F\frac{X}{Z}, \quad y = F\frac{Y}{Z} \quad 。 \tag{1}$$

2.1.2 カメラ座標系

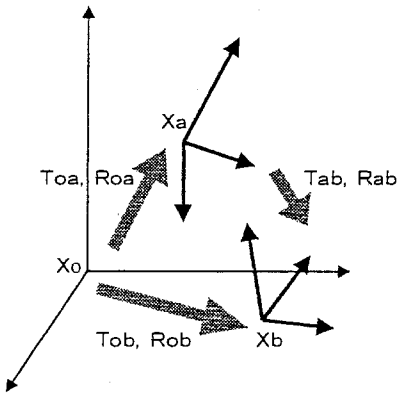


図 6: 基準座標系とカメラ座標系

全ての基準となる座標系を $\boldsymbol{X}_o$, カメラ a の座標系を $\boldsymbol{X}_a$, カメラ b の座標系を $\boldsymbol{X}_b$ とする。$\boldsymbol{X}_a$ は $\boldsymbol{X}_o$ と平行移動 $\boldsymbol{T}_{oa}$, 回転 $\boldsymbol{R}_{oa}$ の関係にあり、$\boldsymbol{W}_b$ は $\boldsymbol{X}_o$ と平行移動 $\boldsymbol{T}_{ob}$, 回転 $\boldsymbol{R}_{ob}$ の関係にあるとする。(図 6)

$$\boldsymbol{X}_a = \boldsymbol{R}_{ob}\boldsymbol{X}_o + \boldsymbol{T}_{oa} \tag{2}$$

$$\boldsymbol{X}_b = \boldsymbol{R}_{ob}\boldsymbol{X}_o + \boldsymbol{T}_{ob} \tag{3}$$

ただし、$\boldsymbol{T}$ はカメラの平行移動

$$\boldsymbol{T} = (T_x, T_y, T_z), \tag{4}$$

$\boldsymbol{R}$ は3次元回転行列、

$$\boldsymbol{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{5}$$

である。

$\boldsymbol{X}_a$ と $\boldsymbol{X}_b$ の関係は以下のように導かれる。

式 (2) より、

$$\boldsymbol{X}_o = \boldsymbol{R}_{oa}^{-1}(\boldsymbol{X}_a - \boldsymbol{T}_{oa}) \tag{6}$$

式 (3) に式 (6) を代入して、

$$\begin{aligned} \boldsymbol{X}_b &= \boldsymbol{R}_{ob}\boldsymbol{R}_{oa}^{-1}(\boldsymbol{X}_a - \boldsymbol{T}_{oa}) + \boldsymbol{T}_{ob} \\ &= \boldsymbol{R}_{ob}\boldsymbol{R}_{oa}^{-1}\boldsymbol{X}_a + \boldsymbol{T}_{ob} - \boldsymbol{R}_{ob}\boldsymbol{R}_{oa}^{-1}\boldsymbol{T}_{oa} \end{aligned} \tag{7}$$

ここで、

$$\boldsymbol{R}_{ob}\boldsymbol{R}_{oa}^{-1} = \boldsymbol{R}_{ab}, \quad \boldsymbol{T}_{ob} - \boldsymbol{R}_{ob}\boldsymbol{R}_{oa}^{-1}\boldsymbol{T}_{oa} = \boldsymbol{T}_{ab} \tag{8}$$

とすると、

$$\boldsymbol{X}_b = \boldsymbol{R}_{ab}\boldsymbol{X}_a + \boldsymbol{T}_{ab} \tag{9}$$

となる。

2.1.3 カメラの平行移動と回転

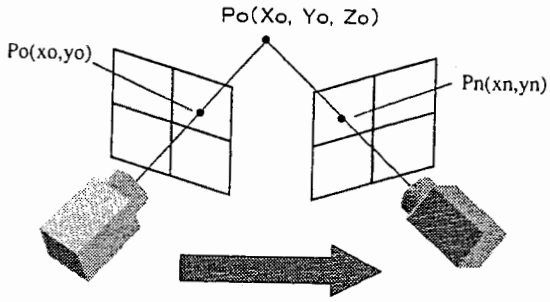


図 7: カメラの平行移動と回転

移動前のカメラ o の座標系を $\boldsymbol{X}_o$, 移動後のカメラ n の座標系を $\boldsymbol{X}_n$ とする。なお、ここからは簡単のため、移動前のカメラ座標系を基準の座標系として扱うことにする。

カメラの移動が、平行移動 $\boldsymbol{T}_{on}$, 回転 $\boldsymbol{R}_{on}$ で与えられるとすると、 $\boldsymbol{X}_n$ は式 (9) より、

$$\boldsymbol{X}_n = \boldsymbol{R}_{on}\boldsymbol{X}_o + \boldsymbol{T}_{on} \tag{10}$$

となる。

座標系 $\boldsymbol{X}_o$ の点 $P_o(X_o, Y_o, Z_o)$ は、カメラ o の撮像平面上の点 $p_o(x_o, y_o)$ に投影される。

$$x_o = F_o\frac{X_o}{Z_o}, \quad y_o = F_o\frac{Y_o}{Z_o} \quad 。 \tag{11}$$

同様に、座標系 $\boldsymbol{X}_n$ の点 $P_n(X_n, Y_n, Z_n)$ は、カメラ n の撮像平面上の点 $p_n(x_n, y_n)$ に投影される。

$$x_n = F_n\frac{X_n}{Z_n}, \quad y_n = F_n\frac{Y_n}{Z_n} \quad 。 \tag{12}$$

式 (10) より、

$$\begin{pmatrix} X_n \\ Y_n \\ Z_n \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X_o \\ Y_o \\ Z_o \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} \tag{13}$$

これを展開すると、

$$\begin{cases} X_n = r_{11}X_o + r_{12}Y_o + r_{13}Z_o + T_x \\ Y_n = r_{21}X_o + r_{22}Y_o + r_{23}Z_o + T_y \\ Z_n = r_{31}X_o + r_{32}Y_o + r_{33}Z_o + T_z \end{cases} \tag{14}$$

となる。

式 (14) に式 (12) を代入すると、

$$x_n = F_n\frac{X_n}{Z_n} = F_n\frac{r_{11}X_o + r_{12}Y_o + r_{13}Z_o + T_x}{r_{31}X_o + r_{32}Y_o + r_{33}Z_o + T_z} \tag{15}$$

$$y_n = F_n\frac{Y_n}{Z_n} = F_n\frac{r_{21}X_o + r_{22}Y_o + r_{23}Z_o + T_y}{r_{31}X_o + r_{32}Y_o + r_{33}Z_o + T_z} \tag{16}$$

となり、ここで

$$X_o = x_o\frac{Z_o}{F_o}, \quad Y_o = y_o\frac{Z_o}{F_o} \tag{17}$$

を代入して整理すると、

$$x_n = F_n\frac{r_{11}X_o + r_{12}Y_o + r_{13}F_o + \frac{F_o}{Z_o}T_x}{r_{31}X_o + r_{32}Y_o + r_{33}F_o + \frac{F_o}{Z_o}T_z} \tag{18}$$

$$y_n = F_n\frac{r_{21}X_o + r_{22}Y_o + r_{23}F_o + \frac{F_o}{Z_o}T_y}{r_{31}X_o + r_{32}Y_o + r_{33}F_o + \frac{F_o}{Z_o}T_z} \tag{19}$$

となる。

2.2 マッピング

2.2.1 forward mapping

forward mapping は入力画像の各画素に対応する出力画像上の座標を求め、そこに入力画像の情報を複写する方法である。出力画像上の対応座標は、入力画像から出力画像へのマッピング関数で計算できる。

入力画像・出力画像とも画素は離散に配置しているため、forward mapping 処理によって得られる出力画像では、埋まらない画素 (mapping crack) が生じる、複数の情報が 1 つの画素に複写される、といった問題が起こる。(図 8)

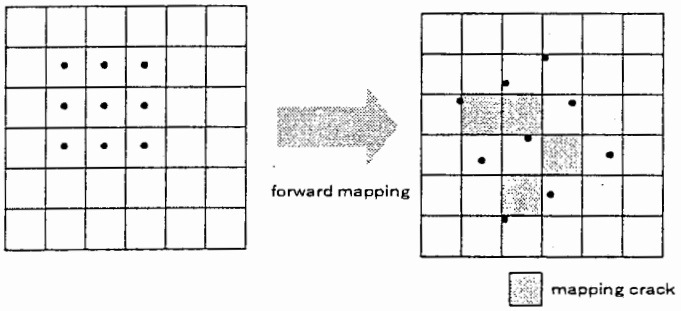


図 8: forward mapping と mapping crack

2.2.2 backward mapping

backward mapping は出力画像の各画素に対応する入力画像上の座標を求め、そこから情報を複写する方法である。入力画像上の対応座表は、出力画像から入力画像へのマッピング関数で計算できる。

この方法の特色は、各画素に対する 1 度のマッピング処理で、出力画像の全ての画素が確実に埋まるということである。出力画像から入力画像へのマッピング関数は、入力画像から出力画像へのマッピング関数の逆関数である。この逆関数が得られる場合は、forward mapping よりも有利な方法である。

3 具体的な処理方法

3.1 処理全体の流れ

本手法は、全体の処理を大きく 2 つの手順に分けることができる。

手順 1: 基準カメラで撮影した映像 (基準映像) を仮想カメラの映像に 3 次元座標変換する。

手順 2: 基準映像には映っていない領域について、周りの参照映像から情報を取得する。

3.2 基準映像から仮想映像への 3 次元座標変換

3.2.1 奥行きマップを利用した 3 次元座標変換

式 (18),(19) で示したように、3 次元座標変換の計算には Z 軸の値、すなわち奥行き情報が必要である。この奥行き情報は、奥行きマップから取得する。3 次元座標変換の実際の処理は、基準映像を仮想映像にマッピングすることである。マッピング方法として望ましいのは、一度の走査で仮想映像の全ての画素が埋まる backward mapping である。しかし、仮想映像の奥行きマップはもともと存在しないので、仮想映像から基準映像へのマッピング関数を計算することができない。従って、基準映像から仮想映像への 3 次元座標変換は、基準映像から仮想映像への forward mapping で行う。

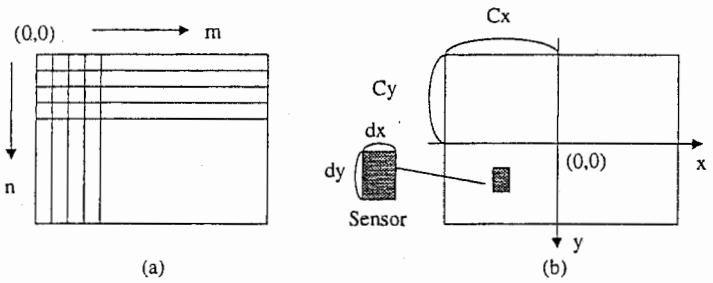


図 9: (a) メモリ上のイメージデータと (b)CCD カメラ

本研究で使用した多眼カメラシステムを構成するＣＣＤカメラは、図 9(b) のようである。それに対し、メモリ上に確保された映像データは図 9(a) のようになっている。これらの間には、

$$x = \frac{dx(m - C_x)}{sx}, \quad y = dy(n - C_y) \tag{20}$$

の関係がある。
ここで、各パラメータは以下の通りである。

C_x,C_y: カメラの光軸が撮像平面と交わる点 [pixel]
dx,dy:1sel に相当する長さ [mm/pixel]
sx:dx の誤差を補正するための係数

式 (20) を式 (18),(19) に代入して整理すると次式が得られる。

$$m_v = \frac{a_1 m_o + a_2 n_o + a_3 Z_o(m_o, n_o)^{-1} + a_4}{a_9 m_o + a_{10} n_o + a_{11} Z_o(m_o, n_o)^{-1} + a_{12}} + C_{x_v} \tag{21}$$

$$n_v = \frac{a_5 m_o + a_6 n_o + a_7 Z_o(m_o, n_o)^{-1} + a_8}{a_9 m_o + a_{10} n_o + a_{11} Z_o(m_o, n_o)^{-1} + a_{12}} + C_{y_v} \tag{22}$$

$Z_o(m_o, n_o)$ は $p_o(m_o, n_o)$ に対応する奥行き値

ただし、

$$
\begin{aligned}
a_1 &= F_v \frac{sx_v}{dx_v} \frac{dx_o}{sx_o} r_{11} \\
a_2 &= F_v \frac{sx_v}{dx_v} dy_o r_{12} \\
a_3 &= F_v \frac{sx_v}{dx_v} F_o T_x \\
a_4 &= -F_v \frac{sx_v}{dx_v} \left(-r_{13} F_o + \frac{dx_o}{sx_o} r_{11} C_{x_o} + dy_o r_{12} C_{y_o} \right) \\
a_5 &= F_v \frac{1}{dy_v} \frac{dx_o}{sx_o} r_{21} \\
a_6 &= F_v \frac{1}{dy_v} dy_o r_{22} \\
a_7 &= F_v \frac{1}{dy_v} F_o T_y \\
a_8 &= -F_v \frac{1}{dy_v} \left(-r_{23} F_o + \frac{dx_o}{sx_o} r_{21} C_{x_o} + dy_o r_{22} C_{y_o} \right) \\
a_9 &= \frac{dx_o}{sx_o} r_{31} \\
a_{10} &= dy_o R_{32} \\
a_{11} &= F_o T_z \\
a_{12} &= r_{33} F_o - \left(\frac{dx_o}{sx_o} r_{31} C_{x_o} + dy_o r_{32} C_{y_o} \right)
\end{aligned}
$$

である。

上式により、基準映像上の座標 $p_o(m_o, n_o)$ に対応する仮想映像上の座標 $p_v(m_v, n_v)$ が計算できる。

3.2.2 mapping crack と super sampling

図 8に示すように、メモリ上の映像データの各画素は離散に配置しており、このまま forward mapping を行うと埋まるべきなのに埋まらない画素 (mapping crack) が生じることがある。この mapping crack を防ぐために、super sampling 手法 [4] を用いる。

super sampling とは、図 10のように各画素に内で 1 点だけ参照するのではなく、もっと密にサンプリングする手法である。

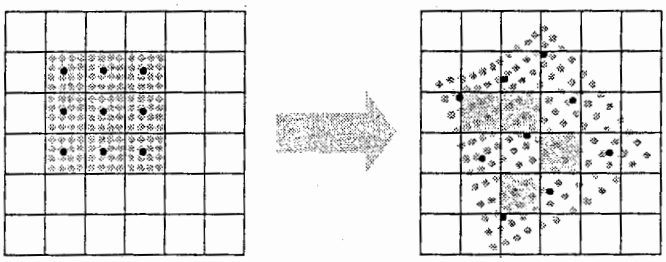


図 10: super sampling

本来の super sampling 手法は、入力画像の各画素内のサンプリング点は画素の中央の値を元に線形補間して用い、出力画像の色情報は各画素に複写された色情報を適切に評価して決定するものである。しかし、本手法では計算の簡便化のため、入力画像のサンプリング点では各画素の中央値をそのまま用い、出

力画像の色情報は次に述べるように各画素に複写された色情報のうち奥行きが最も手前のものを選択するものとする。

この手法によって mapping crack が生じにくくなる。

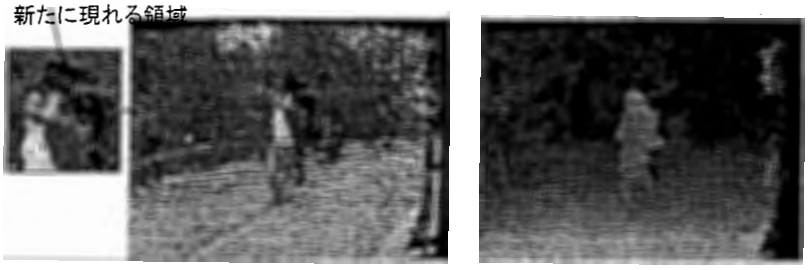
3.2.3 隠れる領域に対する処理

forward mapping を行うと、仮想映像では同一の画素に複数のカラー情報がマッピングされることがある。これは、手前のオブジェクトによって隠されてしまう領域で起こる。

この場合は、奥行き値を比較して常識的に手前のものを優先させればよい。

3.3 新たに現れる領域に対する処理

3.3.1 仮想映像から参照映像へのマッピング関数



(a) (b)

図 11: forward mapping 直後の (a) 仮想映像と (b) 仮想映像の奥行きマップ

新たに現れる領域に関しては、基準映像には何の情報もないので、周りに配置した参照カメラの映像を利用する。

基準映像から仮想映像への forward mapping 処理が終了した直後の仮想映像は、図 11(a) のようである。この図 11(a) で、マッピングされずに残っている画素が、新たに現れる領域である。この領域の各画素に対応する参照映像の座標を計算し、そこからカラー情報を持ってくる。このときの処理は、仮想映像から参照映像への backward mapping である。

仮想映像から各参照映像へのマッピング関数を計算するためには、仮想映像の奥行きマップが必要となる。基準映像から仮想映像への forward mapping の際に、奥行きマップも同時にマッピングしておくと、図 11(b) のようになる。この奥行きマップも、現れる領域に関してはマッピングされていない。現れる領域に関する奥行き情報は、どこからも得ることができないので、周りの奥行き情報を用いて補間する。

3.3.2 奥行きマップの補間

奥行きマップの補間は、仮想カメラの移動の仕方を考慮して行う。カメラの移動による画面内オブジェクトの移動はエピポーラ線に沿うので、補間はエピポーラ線に沿った方向から画素を選択して行う。(図12) 新たに現れる領域は、手前オブジェクトに隠されていた奥の領域と推察できるから、エピポーラ線の方向にある隣接画素のうち、最も奥の奥行き値で補間することとする。ただし、計算の簡便化のため、ここでの補間は線形補間ではなく、単純に隣接した画素から値を複写するものとする。

図13に補間した後の奥行きマップを示す。

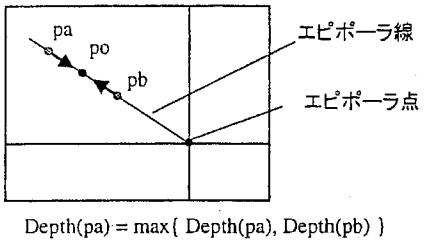


図 12: 画素の補間方向

図 13: 補間後の仮想映像の奥行きマップ

3.3.3 参照映像の選択

補間により仮想映像の奥行きマップを得たので、仮想映像から各参照映像へのマッピング関数が計算できる。次に、どの参照映像から情報を持ってくるかを決定する。

まず、仮想カメラの位置から、利用する参照カメラの候補を選ぶ。エピポーラ点を計算し、この点が基準映像を 8 つに分割した領域のいずれに属するかによって参照映像の候補を選定する。(図 14)

次に、新たに現れる領域の各画素について、候補映像のうちどれから情報を持ってくるかを決定する。ここでは、対応する参照映像の座標を計算し、その座標には基準映像にはない参照映像特有の情報が確かに存在するか否かを判定することによって、用いる映像を決定する。どの候補映像にも有効な情報がないと判定した場合は、まわりのカラー情報を用いて補間する。この時の補間方法は、奥行きマップの補間と同様である。

対応する座標の有効かどうか判定する方法について述べる。まず、基準映像の奥行きマップを各参照映像の奥行きマップに forward mapping する。マッピング後の各参照映像の奥行きマップでは、基準映像に情報がない領域についてはマッピングされていない。この、マッピングされていない領域にこそ各参照映像にしかない情報が存在するのである。つまり、対応する座標が有効かどうかは、その座標に奥行きマップがマッピングされているかどうかで判定できるのである。(図 15)

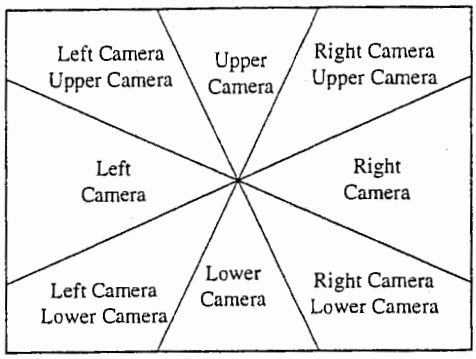


図 14: 参照カメラの候補選択

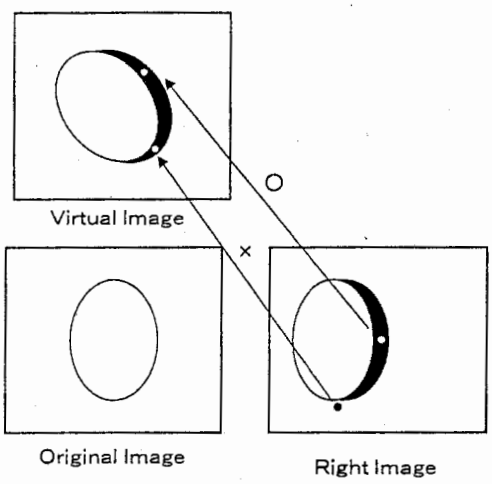


図 15: 参照映像の有効度判定

これまでの基準映像から仮想映像への forward mapping、仮想映像から参照映像への backward mapping によって、任意視点映像が生成できる。

3.4 高速化のための手法

3.4.1 補間アルゴリズムの単純化

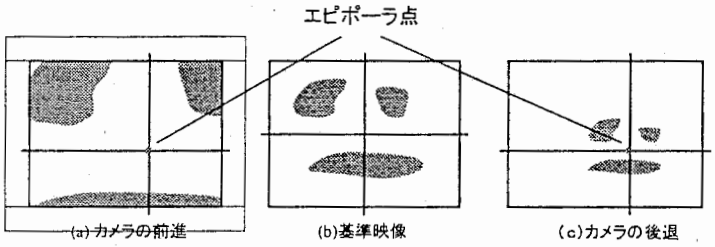


図 16: camera の (a) 前進・(c) 後退に伴う画面内オブジェクトの移動

奥行き・カラー情報の補間は、エピポーラ線に沿った方向から画素を選択して行うと述べた。しかし、補間すべき全ての画素に関してエピポーラ線を計算することはかなりの計算量である。そこで、高速化のために補間のアルゴリズムを単純化することにした。

カメラの前進・後退により、画面内のオブジェクトはエピポーラ点を中心に、図 16のように収縮したり

拡散したりする。このことから、仮想映像をエピポーラ点中心に領域分割し、各領域ごとに補間する方向を統一する方法で、アルゴリズムを単純化することにした。

補間すべき全ての画素についてエピポーラ線を計算した場合でも、補間処理は隣接画素の値をそのまま持ってくるだけ (線形補間しているわけではない) なので、単純化したアルゴリズムと画質の面では大差がない。

3.4.2 super sampling の最適化

基準映像から仮想映像への forward mapping 処理は、全体の処理の中で最も計算量の多い行程である。この処理の計算量は、super sampling のサンプリング数に大きく影響される。

カメラの移動による画面内オブジェクトの移動量は、手前のものは大きく、遠くのものは小さい。これは、手前のオブジェクトに関してはサンプリング数が十分でないと mapping crack が生じるが、遠くのオブジェクトはサンプリング数が少なくても mapping crack は生じにくいということである。このことから、各画素の奥行き値に応じてサンプリング数を調整し、forward mapping 処理の効率向上を図った。また、カメラの前進・後退、ズームイン・アウトに応じたサンプリング数の調整も行った。

3.5 実験

本手法を用いて、生成した任意視点映像を図 17 に示す。隠れる領域、現れる領域が処理されているのがわかる。

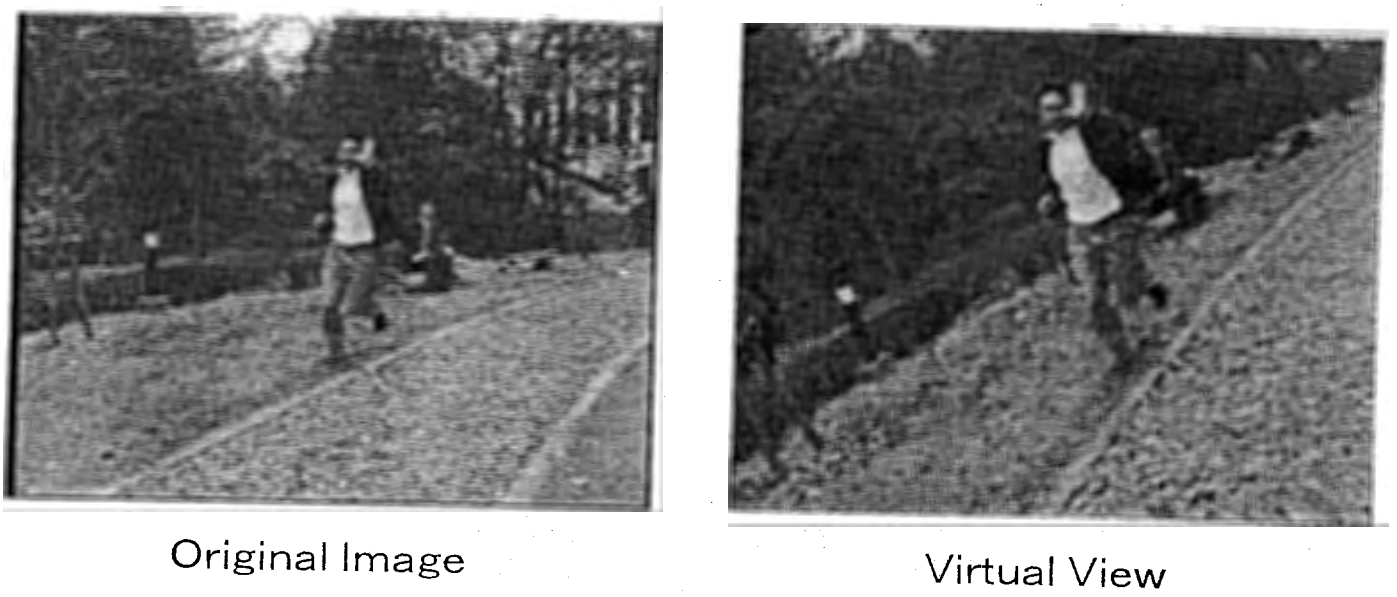


図 17: 実験結果

4　任意視点映像をリアルタイムで生成・表示するシステムの試作

4.1　システム構成

ユーザからの仮想カメラ位置の入力に応じて、即時に任意視点映像を生成し表示するシステムを試作した。

4.1.1　実装

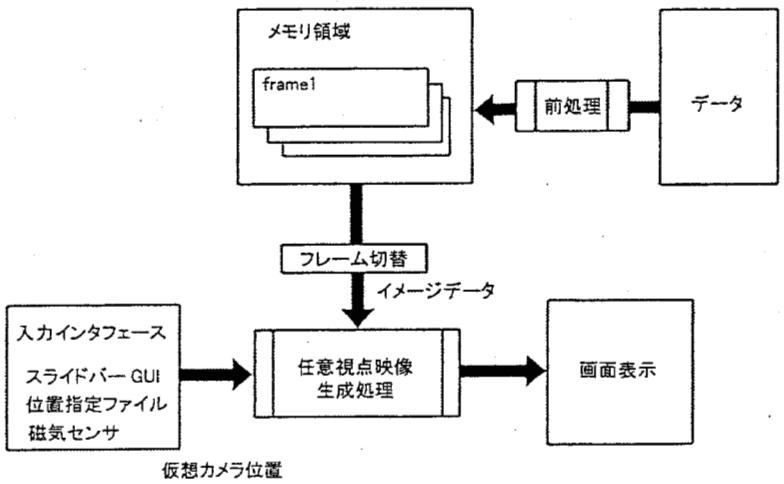


図 18: プログラムの構成図

作成したプログラムの構成図を図 18 に示す。開発環境は、 Microsoft Visual C++ ver.5.0 RISC edition である。

図 18のように、使用する全ての画像データをメモリ上に読み込んでおき、必要な前処理 (画像のレンズ歪み補正、基準映像の奥行きマップを各参照映像にマッピングするなど) を施しておく。動画を使用する場合も、全てのフレームの映像データについて同様にする。

入力インタフェースの種類に応じて適切な入力処理を行い、仮想カメラ位置をもとに任意視点映像を生成・表示する。動画の場合は、適当なタイミングで処理の対象となるデータを、次のフレームのものに差し替える処理を行う。

仮想カメラ位置の入力インタフェースとして、スライドバーコントロール、位置指定ファイル、磁気センサの三つを用意した。

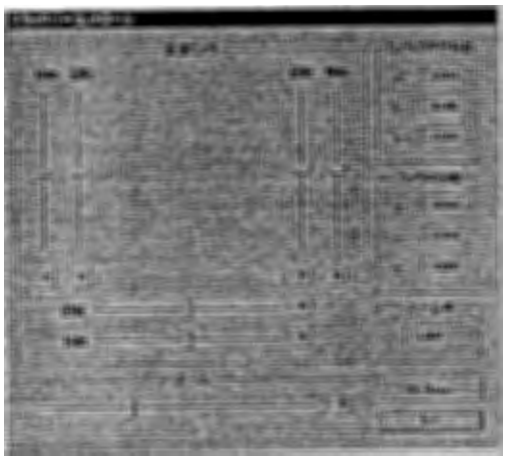


図 19: スライドバーコントロールＧＵＩ

スライドバーコントロールは、図19のように位置決定のための各パラメータを、スライドバーを使って入力するインタフェースである。

位置指定ファイルは、あらかじめ各パラメータを書き込んでおいたファイルである。

磁気センサを用いたインタフェースは、ダミーのカメラ模型に磁気センサを取り付け、ダミーカメラの位置・方向を検出してカメラ位置を入力するものである。

なお、詳しい内容は、ドキュメントファイルを参照のこと。

4.1.2 磁気センサからのカメラ位置入力

磁気センサによって、ダミーカメラの位置・回転角が取得できる。基準位置を任意に決め、その基準位置に対する現在のダミーカメラの位置・回転角は以下のようにして求める。磁気センサユニットの座標系を X_o、任意に決めた基準座標系ををを X_r、現在のダミーカメラ座標系を X_n とする。

$$\boldsymbol{X}_r = \boldsymbol{R}_{or}\boldsymbol{X}_o + \boldsymbol{T}_{or} \tag{23}$$

$$\boldsymbol{X}_n = \boldsymbol{R}_{on}\boldsymbol{X}_o + \boldsymbol{T}_{on} \tag{24}$$

$$\boldsymbol{X}_n = \boldsymbol{R}_{rn}\boldsymbol{X}_r + \boldsymbol{T}_{rn} \tag{25}$$

式 (23) より、

$$\boldsymbol{X}_o = \boldsymbol{R}_{or}^{-1}\boldsymbol{X}_r - \boldsymbol{T}_{or}。 \tag{26}$$

式 (24) に式 (26) を代入すると

$$\begin{aligned} \boldsymbol{X}_n &= \boldsymbol{R}_{on}\boldsymbol{R}_{or}^{-1}(\boldsymbol{X}_r - \boldsymbol{T}_{or}) + \boldsymbol{T}_{on} \\ &= \boldsymbol{R}_{on}\boldsymbol{R}_{or}^{-1}\boldsymbol{X}_r + \boldsymbol{T}_{on} - \boldsymbol{R}_{on}\boldsymbol{R}_{or}^{-1}\boldsymbol{T}_{or} \end{aligned} \tag{27}$$

となる。

式 (27) = 式 (25) であるから、

$$\boldsymbol{R}_{rn} = \boldsymbol{R}_{on}\boldsymbol{R}_{or}^{-1}, \quad \boldsymbol{T}_{rn} = \boldsymbol{T}_{on} - \boldsymbol{R}_{on}\boldsymbol{R}_{or}^{-1}\boldsymbol{T}_{or} \tag{28}$$

である。

上式で得られる $\boldsymbol{R}_{rn}$ と $\boldsymbol{T}_{rn}$ を、仮想カメラ位置を与えるパラメータとして入力する。

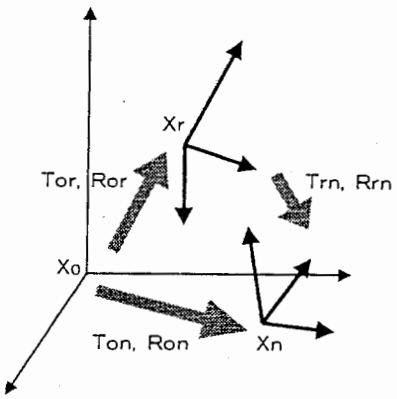


図 20: 磁気センサユニット座標系とダミーカメラ座標系

4.2 実験

試作したシステムを用いて、任意視点映像を生成する実験を行った。実験環境は以下の通りである。

使用コンピュータ: DEC Alpha CPU 533MHz, メモリ 256MB, OS Windows NT 4.0

映像データ: 320 × 240 pixel RGB 各 8 ビットカラーイメージ

4.2.1 生成速度

映像の生成速度は、4-10[frame/sec] であった。この速度は、映像の奥行きの分布ぐあい、カメラの移動量によって変化する。静止画と動画では、生成速度にほとんど差がなかった。

4.2.2 スライドバーコントロール

スライドバーによる入力インタフェースを用いた場合、入力に応じてインタラクティブに任意視点の映像が表示された。しかし、一度に操作できるスライドバーは1つなので、様々なパラメータを組み合わせたカメラの移動を再現できなかった。例えば、カメラを右上方向に移動させるには、一度右方向へ移動した後、上へ移動させなければならず、斜めの移動は再現できない。

4.2.3 位置指定ファイル

位置指定ファイルを用いた場合は、様々なカメラの移動を再現できた。しかし、ユーザが望む位置を数値データとして入力するのは難しい。カメラに同じ動きを繰り返させる場合は便利である。

4.2.4 磁気センサ

磁気センサを用いたインタフェースでは、ダミーカメラが移動した通りの映像を生成することができた。ただし、磁気センサのコントロールユニットとの通信速度が遅いため、映像の生成速度は若干低下する。

5 考察

5.1 仮想カメラの移動範囲

基準位置からの仮想カメラの移動量が、小さい場合と大きい場合の生成映像を示す。(図 21) 移動量が小さい場合は良好な画質であるが、移動量が大きい場合はカメラの可動範囲が狭いため画質はよくない。

現在のカメラの可動範囲は x y 方向の平行移動が 40[cm] 程度、パン・チルト角が＋－5[deg] 程度である。(基準カメラと各参照カメラとの距離は約 20[cm])

様々な応用を考えた場合、可動範囲はもっと広いほうがよい。カメラの可動範囲を拡張する方法として、参照カメラを増やす、パノラマ映像を利用するなどが考えられる。

Original Image

(a)

(b)

図 21: 移動量の違いによる画質の差, (a) カメラ位置：右約 10cm (b) カメラ位置：右約 30cm, 上約 30cm

5.2 応用

任意視点映像生成技術の応用について考察する。

5.2.1 ＶＲへの応用

本手法による任意視点映像は奥行きマップ付きで生成されるので、Z-Key 法 [5] などを用いた 3 次元映像合成が可能である。人物を撮影するカメラのカメラワークに合わせて、任意視点の背景映像を生成して合成することなどが考えられる。

このことを利用して、次のようなシステムに応用できる。図 22のような、クロマキーステージと大画面スクリーンを備えたスタジオを用意する。ステージ上のユーザは、奥行き付きの背景映像と自分の姿が 3 次元合成された映像を、スクリーンで見ながら自由に行動することができる。ユーザあるいはカメラの動きに応じて任意視点の背景映像を生成し、人物像と合成する。ユーザは、自分あるいはカメラの動きに応じてインタラクティブに変化する仮想世界にいる、自分自身を見ることができるのである。

仮想世界の構築にはＣＧを利用するのが一般的である。しかし、モデルベースのＣＧで自然風景などを 3 次元的に再現するためには、複雑な 3 次元モデルを扱う必要があり、多大なコストがかかる。これに対し、本手法はイメージベースの処理であるので、実写映像をもとに比較的手軽に 3 次元映像処理が行える。このことから、自然風景などを用いた仮想世界の構築など、ＶＲへの応用が期待できる。

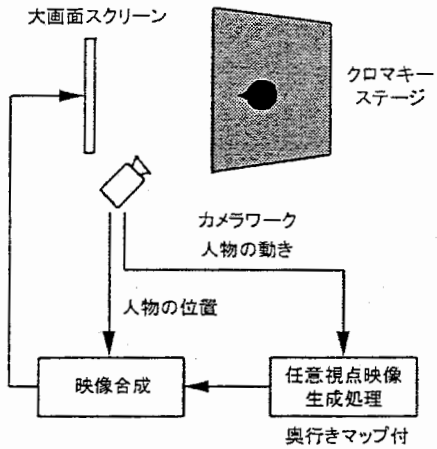


図 22: バーチャルスタジオ

5.2.2 マルチメディアデータベース

通常の映像や音声に加え、様々な属性情報をマルチメディアデータベースに蓄積し、それらを自由に組み合わせて出力することで、ユーザがイメージ通りのコンテンツを柔軟かつ効率的に作成・加工・編集するのを支援するシステムの研究が進められている [6]。ここで属性情報とは、映像・音声を構成する、あるいはそれらに付随する情報、例えば、形状や奥行きなどの 3 次元情報、テクスチャ、被写体の動き情報、カメラワーク、印象キーワードなどである。ユーザはそれらをマルチメディア部品として、専門的な知識なしに簡単な操作で利用することができるのである。

本手法による任意視点映像は、このマルチメディアデータベースに応用できるものと考える。例えば、いろいろな風景の多視点映像と奥行きマップをデータベースに保存しておけば、ユーザは様々な視点からの好きな風景を合成用の素材として利用することができるのである。映像データを全て保存しておくのに比べて、はるかに少ないデータ量で様々な視点の映像を提供できることから、マルチメディアデータベース構築の実現に貢献するものであると考える。

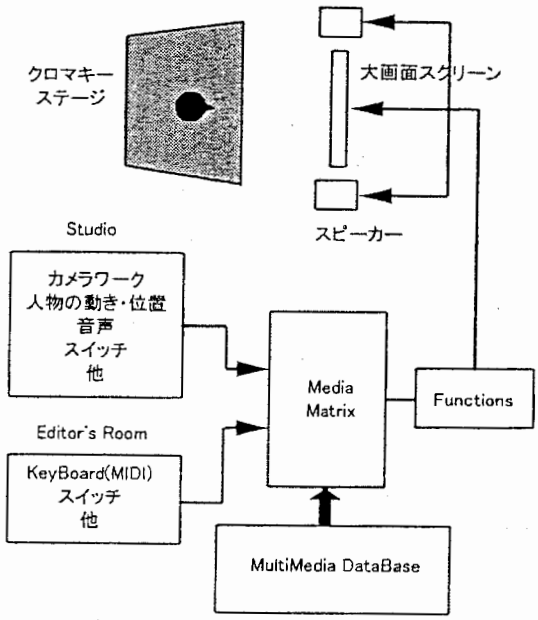


図 23: マルチメディアデータベースを利用したバーチャルスタジオ

謝辞

実習を行うにあたり、多大な御指導をして下さいました朴 鐘一研究員に心から感謝いたします。また、数々の御助言・御支援をしていただきました、岩舘 祐一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。最後に、充実した研究環境と貴重な経験の場を提供して下さった、ATR知能映像通信研究所ならびに長岡技術科学大学にこの場を借りて御礼申し上げます。

参考文献

[1] R.Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the shelfTV cameras and lenses," *IEEE J. Robotics and Automation*, vol.RA-3, no.4, pp.323-344, Aug 1987.

[2] 朴、井上、"多眼カメラを用いた奥行き情報抽出とその映像表現への応用," 第3回画像センシングシンポジウム,pp.247-252, 1997年6月

[3] J.Park and S.Inoue, "Toward occlusion-free depth estimation for video production," *Proc. Intel' Workshop on New Video Media Technology'97"*, pp.131-136, Tokyo, Japan, Jan 1997.

[4] G.Wolberg, *Digital Image Warping*, IEEE Computer Society Press, 1990.

[5] T.Kanade *et al.*, "A stereo machine for video-rate dense depth mapping and its new applications," *Proc. IEEE CVPR'96*, pp.196-202, San Francisco, June 1996.

[6] S.Inoue, "Mental image expression by media integration - COMICS," *Proc. of 1st International Workshop on New Video Media Technology*, pp.47-52, Seoul, Korea, March 1996.

[7] J.Park and S.Inoue, "Arbitrary view generation using multiple cameras," *Proc. IEEE ICIP'97*, vol.I, pp.149-153, Santa Barbara, USA, Oct. 1997.

[8] 朴、井上、"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

[9] 朴、千葉、岩館、"多眼カメラを用いた効率的な任意視点映像生成", 信学技報, Vol.98, No.574, EID98-165, IE98-156, pp.7-12,Feb 1999.

付録 - 作成したプログラムについて

A　操作方法

A.1　ファイルを選択する

1. ［メニュー (M)］から［ファイル選択 (F)］を選択。
2. イメージファイルを選択。カラーイメージ (ppm 形式)、奥行きマップ (pgm 形式)。
3. キャリブレーションファイルを選択。
4. フレーム数を設定。

注 イメージファイルのファイル名のフォーマットは ooooxxx.ppm とすること。
(o: 任意,xxx: フレーム番号) 例： center_309.ppm

A.2　前処理を行う

1. ［メニュー (M)］から［PRIPROCESS(P)］を選択。
   - イメージデータをメモリに格納し、前処理 (レンズの歪み補正、各参照カメラの奥行きマップ生成) をする。

A.3　任意視点映像生成 (メイン処理) を行う

1. ［メニュー (M)］から［VSWC(V)］を選択。
   - ［Movie Control］と［インタフェース］ダイアログが表示される。

A.3.1　スライドバーインタフェースを使う場合

1. ［インタフェース］ダイアログで［スライドバー］を選択する。
   - ［スライドバーコントロール］ダイアログが表示される。
2. スライドバーを操作して任意視点映像を生成する。
3. ［Mpvie Control］ダイアログの［>］ボタンで、一定間隔でフレームを切り替えるようになる。

A.3.2　磁気センサ (FASTRAK ユニット) を使う場合

1. ［インタフェース］ダイアログで［磁気センサ］を選択する。
   - ［磁気センサ］ダイアログが表示される。
2. FASTRAK ユニットの準備をする。(ディップスイッチは 11001001)
3. ［初期化］ボタンを押す。
   - FASTRAK ユニットの初期化処理を行います。
4. 磁気センサを動かして、適当な位置で [位置決め] ボタンを押す。
5. ［VSWC 同期］をチェックする。
   - 磁気センサの位置に応じた任意視点映像が表示される。
6. ［L］ボタンを押すと、そのパラメータは固定される。
7. ［Mpvie Control］ダイアログの［>］ボタンで、一定間隔でフレームを切り替えるようになる。

A.3.3 Vp(View point) ファイルを使う

1. [Movie Control] ダイアログの [use vp] をチェックする。
2. Vp ファイルを選択する。
3. [¿] ボタンで、Vp ファイルに対応した任意視点位置映像が表示される。
4. [R] ボタンは、Vp のリセット。

*** Vp ファイルのフォーマット ***
Frame Zoom Rx Ry Rz Delx DelY DelZ

A.3.4 スナップショット

1. [インタフェース] ダイアログから [スナップショット] を選択する。
   - [スナップショット] ダイアログが表示される。
2. [Brawse] ボタンを押して、出力ファイルを指定する。
3. [Write] ボタンを押す。
   - 現在表示されているイメージがファイルに書き込まれる。

B　クラスの概要

********** 主要クラス **********

CVswc_nApp -Developer Studio がつくるクラス

CVswc_nDoc -Developer Studio がつくるクラス

CVswc_nView -Developer Studio がつくるクラス

CMainFrame -Developer Studio がつくるクラス

CVs - 任意視点映像生成処理をするクラス

CPriProcess - 前処理をするクラス

CFrakCtrl -FASTRAK(磁気センサユニット) を制御するためクラス

********** ダイアログクラス **********

CSelFileDialog - ファイル入力用ダイアログ

CProgress -PROPROCESS 時にプログレスバーを表字するダイアログ

CIFSelect - インタフェースを選択するダイアログ

PlayDialog -[Movie Control] ダイアログ

CVswcCtrl -[スライドバーコントロール] ダイアログ

CFrakDlg -[磁気センサ] ダイアログ

VpDialog -[View Point File] ダイアログ

CDlgWriteFile -[スナップショット] ダイアログ

CErrDlg - エラー表示用ダイアログ

CAboutDlg -About ダイアログ

********** イメージクラス **********

CCIMGC - 基準カメラ用クラス (CCIMG から派生)

CCIMGI - 参照カメラ用クラス (CCIMG から派生)

CCIMGV - 仮想カメラ用クラス (CCIMG から派生)

CDIMG - 奥行きマップ (pgm) 用クラス (CDIMG から派生)

CIMGDISP - 画面に表示するイメージのクラス

CCIMG - カラーイメージ用 (ppm) 基本クラス (CIMG から派生)

CIMG - イメージ用基本クラス

********** 構造体 **********

CalibConstants

CameraPara

MapCoef

MapPara

InitData - フレームごとの焦点距離 F , カメラ間隔 L 等

VpData - 現在の ViewPoint

VpArray -VpFile 内の VpData を格納したもの

sPImgObj - 現在フレームのデータ

FrmData - ファイル入力時に指定したフレームデータ

PosData - 磁気センサから読み込んだ位置データ

RgbData -RGB データ構造体

C　処理の流れ

C.1　ファイルの入力 ～ 前処理

1. ファイル名を入力するダイアログが閉じた時点で、指定したフレーム分だけイメージオブジェクトが生成され、各々にファイル名が割り当てられる。
2. フレームごとに以下の処理をする。
   (a) 各々のファイルからイメージデータ、キャリブレーションデータを読み込む。
   (b) 各カメラ映像ごとにレンズの歪み補正をする。
   (c) マッピングパラメータを計算して、各参照映像の奥行きマップを生成する。

C.2　スライドバーコントロール

C.2.1　静止画

1. スライドバーを操作すると、MainFrame にメッセージを送る。
2. CMainFrame クラスはメッセージを受け取ると、スライドバーの位置から仮想カメラ位置を決定し、任意視点映像を生成して表示する。

C.2.2　動画

1. Movie Control の [¿] が押された場合、MainFrame にメッセージを送る。
2. CMainFrame クラスはメッセージを受け取ると、タイマ関数を発生させる。
3. CMainFrame クラスは、タイマ関数により一定間隔でスライドバーの位置から仮想カメラ位置を決定し、任意視点映像を生成して表示する。その後、フレームを切り替える。
4. フレームの切り替えは、CVs クラスが参照するデータ領域のアドレスを、該当するフレームのものに変更するだけ。

C.3　磁気センサ

C.3.1　静止画

- タイマ関数により、一定間隔で磁気センサからカメラ位置を読み取り、任意視点映像を生成して表示する。

C.3.2　動画

- 静止画と同様。フレームを一定間隔で切り替える。

C.4　VpFile

1. Vp ファイルから、Vp データを VpArray に格納する。
2. タイマ関数により、一定間隔で VpArray からカメラ位置を読み出し、任意視点映像を生成して表示する。

D　Ａｌｐｈａ以外のプロセッサ用にビルドする方法（VC++ 5.0)

(例) Ｉｎｔｅｌ ＰｅｎｔｉｕｍII用にビルドする。（普通のＤＯＳ／ＶのＷｉｎｄｏｗｓ用にビルドする）

1. Developer Studio でプロジェクトを読み込む。
2. ［ﾋﾞﾙﾄﾞ (B)］から［構成 (F)］を選択。
3. ［構成］ダイアログから［追加］を選択。
4. 該当するプラットフォーム（この場合は Win32）を選択して追加する。

E　必要なファイル

vswc_n.exe 任意視点映像生成プログラムＧＵＩ

vswc_n.dsp プロジェクトファイル

********** ソースファイル **********

IMG.cpp

CIMG.cpp

CIMGC.cpp

CIMGI.cpp

CIMGV.cpp

DIMG.cpp

DlgWriteFile.cpp

ERRDIALOG.cpp

ErrDlg.cpp

FileName.cpp

FrakCtrl.cpp

FrakDlg.cpp

IFSelect.cpp

Img_Dsip.cpp

InputFiles.cpp

MainFrm.cpp

PlayDialog.cpp

PriProcess.cpp

Progress.cpp

SelFileDialog.cpp

StdAfx.cpp

VpDialog.cpp

Vs.cpp

vswc_n.cpp

vswc_nDoc.cpp

vswc_nView.cpp

VswcCtrl.cpp

********** ヘッダーファイル **********

CIMG.h

CIMGC.h

CIMGI.h

CIMGV.h

DIMG.h

DlgWriteFile.h

ERRDIALOG.h

ErrDlg.h

FileName.h

Frak_Defines.h

FrakCtrl.h

FrakDlg.h

IFSelect.h

IMG.h

Img_Dsip.h

InputFiles.h

MainFrm.h

PlayDialog.h

PriProcess.h

Progress.h

resource.h

SelFileDialog.h

StdAfx.h

structs.h

structs2.h

VpDialog.h

Vs.h

vswc_n.h

vswc_nDoc.h

vswc_nView.h

VswcCtrl.h