J-044

FFT を用いたパノラマ映像生成について

On the Panorama Image Generation based on FFT Technique

米本 良 † **上杉 徹** † **川村 尚生** ‡ **菅原 一孔** ‡
Ryo Yonemoto Toru Uesugi Takao Kawamura Kazunori Sugahara

1. はじめに

最近ではネットワークの通信速度の高速化や高性能の映像撮影装置の低価格化に伴い，テレビ会議が開催される場面が増えてきている．ところが現在のテレビ会議では，1 台のビデオカメラで会場全体を撮影するのが通常である．このため，撮影された映像から話者の表情を読み取ることが困難であるなどの問題がある．この問題を解決するため，複数台のビデオカメラで会場全体の映像と話者の映像を別々に撮影する場合もあるが，複数のビデオカメラを用意するにはコストがかかる上にその切替操作には人手がかかるなどの問題が出てくる．そこで，我々は会議会場全体の広範囲の映像を表示するために，３つの NTSC ビデオカメラを用いて撮影した映像から生成したパノラマ映像と，話者を映している映像をカメラ内部で合成したものを映像信号として出力するテレビ会議用ビデオカメラを開発している．このような装置を構成する際には，話者追従の手法と共に複数の映像からのパノラマ映像の合成手法が重要な機能として上げられる．本稿では，パノラマ映像を合成する手法として，Alistair J. Fitch らが提案した Fast Robust Correlation 手法 [1] を取り上げそのハードウエア実現について述べる．

2. システムの構成

システムは，3 つの NTSC ビデオカメラを用いた映像撮影装置，NTSC ビデオデコーダ LSI，メモリ，FPGA ボード，デジタルビデオエンコーダ，モニタから構成されている．

処理の流れは以下のとおりである．まず，NTSC ビデオカメラから送られてきた NTSC アナログ映像信号をデコーダによってデジタルの RGB 各 8bit のデジタル映像信号に変換し FPGA ボード上のメモリに蓄積する．そして FPGA 上の各回路において，入力された映像データを基にパノラマ映像生成の処理を行い，エンコーダで NTSC アナログ映像信号に変換しモニタに表示する．構成している各装置を図 1 に示す．

3. パノラマ映像生成

パノラマ映像生成では，会議会場全体の広範囲の映像を表示するために，視野角の違う 3 つの NTSC ビデオカメラで撮影した入力映像を基にパノラマ状の映像を生成する．しかし，撮影した入力映像の映像サイズ (640×480[pixel]) のままでは，パノラマ映像を生成する際，想定している出力映像の映像サイズ (640×480[pixel]) より大きくなり，パノラマ映像を生成することはできない．そこで，それを防ぐために入力映像を 3/8 サイズの 240×180[pixel] に縮小し一部分を重ねて合成し，640×180[pixel] で表示することにした．

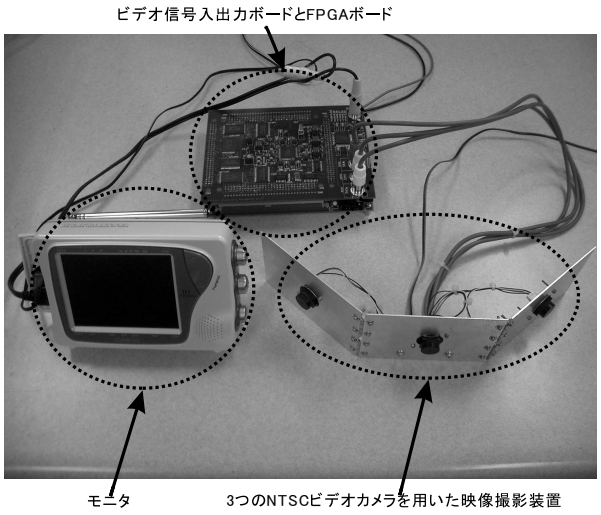

図 1: 装置

映像サイズの縮小方法として，まず，撮影した入力映像をメモリに書き込む時に，1 画素ずつ飛ばして書き込むことで，入力映像の映像サイズを 1/2 に縮小する．さらに，入力映像を 3/4 サイズにするために，パノラマ映像を生成する際に縮小前の映像から画素を縦横方向にそれぞれ 4 画素に 1 画素間引くという手法を用いた．またパノラマ映像の生成方法は，入力映像データをメモリの空き領域にコピーすることによって行われる．まず，映像サイズを縮小しながらメモリの決められたアドレスに 3 つの映像データの横 1 行分をコピーする．続いて 2 行目，3 行目とコピーするが，縦方向も 4 回に 1 回行を飛ばして縮小しながら全映像をコピーしてパノラマ映像を生成する．

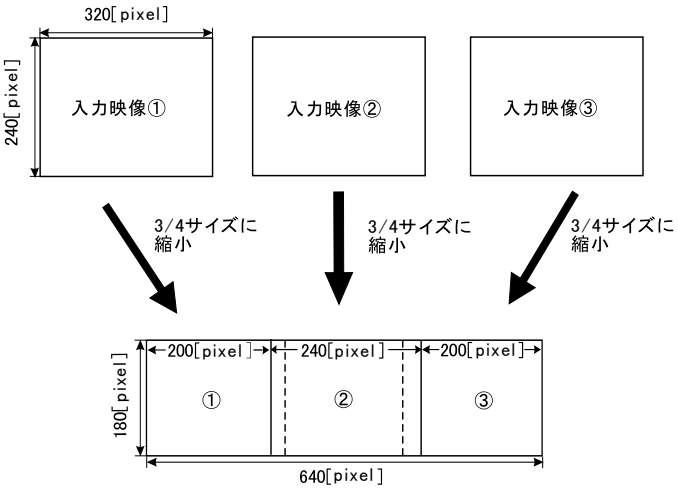

図 2: パノラマ映像を生成する様子

†鳥取大学 大学院 工学研究科 知能情報工学専攻
‡鳥取大学 工学部 知能情報工学科

4. パノラマ映像生成の自動補正処理

4.1 自動補正処理

自動補正処理とは，上記で述べたようにパノラマ映像を生成する際に，映像同士の最適な結合位置を検出する必要があり，その結合位置を自動で検出し最適な位置に映像を補正する処理を行うことである．この補正処理を図 2 に示す入力映像①と入力映像②，入力映像②と入力映像③の各々の境界領域で行う．

4.1.1 Fast Robust Correlation 手法 [1]

今，図 3 に示すように，画像 g を画像 f に対して x 軸方向に m，y 軸方向に n ずらすと仮定する．このとき，$\boldsymbol{x}, \boldsymbol{m}$ を $\boldsymbol{x} = (x, y), \boldsymbol{m} = (m, n)$ と定義すると，2 つの画像 f, g の画素差は以下のように表すこととする．

$$R = f(x) \cdot \alpha_f(x) - g(x-m) \cdot \alpha_g(x-m) \qquad (1)$$

また，2 つの画像 f, g の全体としての画像の差は次のようになる．

$$R(m) = \sum_x h\left(f\left(x\right) - g\left(x-m\right)\right) \cdot \alpha_f(x) \cdot \alpha_g(x-m) \quad (2)$$

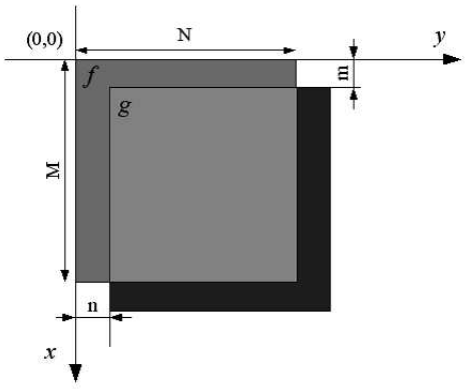

図 3: 画像比較

ここで，式 (1) と式 (2) に示している $\alpha_f(x)$ と $\alpha_g(x)$ はアルファマスクと呼ばれ, 任意の形状の領域に柔軟に誤差評価の際の重みを持たせることができる．本研究では画像全面に対して比較するので，$\alpha_f(x)$，$\alpha_g(x)$ 共に，全面が 1 の値を持つものとする．また，式 (2) 中の，画素ごとの誤差 r に対する評価関数 $h(r)$ は，偶関数である cos 関数を用いて式 (3) で与える．

$$h(r) \approx \sum_{p=1}^{P} b_p \left(1 - \cos\left(a_p \pi r\right)\right) \qquad (3)$$

ここで，式 (3) は b_p と a_p というパラメータを含んでいるが，$P = 1$ の場合には $b_1 = 1, a_1 = 1.0$ となる．式 (3) を式 (2) に代入して整理すると

$$\begin{aligned} R(m) &= \sum_x [(\alpha_f(x) \cdot \alpha_g(x-m)) \sum_{p=1}^{P} b_p (1 - \cos(a_p \pi \\ &* \quad (f(x) - g(x-m))))] \end{aligned} \qquad (4)$$

となる．これを整理すると

$$\begin{aligned} R(m) &= \sum_x [(\alpha_f(x) \star \alpha_g(x)) \sum_{p=1}^{P} b_p \\ &- \sum_{p=1}^{P} b_p ((\alpha_f(x) e^{j a_p \pi f(x)}) \\ &\star \quad (\alpha_g(x) e^{j a_p \pi g(x)}))] \end{aligned} \qquad (5)$$

の関係を得る．ただし，式 (5) 中の $\star$ は畳み込み演算を示している．

4.1.2 周波数領域での画像比較

式 (5) を空間領域で直接求めずに，空間周波数領域で間接的に求めることにする．また，その計算の流れを図 4 に示す．まず，f, g には RGB 成分の R 値を与え，上の 4 つの項に対してそれぞれ二次元 FFT を行う．結果を F, G とし FG^* を求める．ただし，G^* は G の複素共役を表す．その後，$\times \sum b_p$ や $\times b_1$ という演算があるが，上記で述べたように，$P = 0$ として $b_1 = 1$ なので，そのまま差を取る．そして最後に，その差の値に対して二次元 IFFT を行い，$R(m)$ を得る．この得られた $R(m)$ について最小値を求め，最小値となる (m, n) により，画像 f, g の合成画像となるよう画像 g を移動させることで自動補正処理を行う．また，画像サイズ $N \times M$[pixel] の画像を FFT を用いずに比較した場合の計算量は $O(M^2N^2)$ であるが，FFT を用いて周波数領域で処理を行うと計算量は $O(MN \log_2 N)$ となる．

4.2 FFT のハードウエア化

4.2.1 バタフライ演算

図 4 に示すとおり，本手法は FFT の処理が中心的な役割を果たす．FFT をハードウエア化するために，バタフライ演算をハードウエア化する必要がある．バタフライ演算とは，図 5 に示す計算処理である．

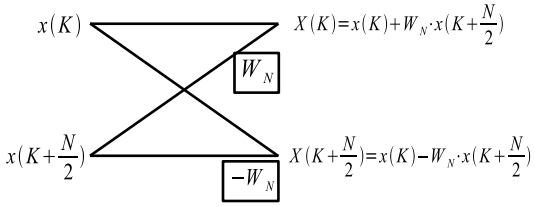

図 5: バタフライ演算

バタフライ演算をハードウエア化するにあたり，記述方法により FPGA 上に形成されるハードウエア量が大きく異なるので，その記述には注意が必要である．

本稿で扱うデータはすべて仮数部 23 ビット，指数部が 8 ビット，符号ビット 1 ビットの合計 32 ビットの浮動小数点で表現されているもとし，FPGA 上でこのような浮動小数点の計算を行うために，ALTERA 社の QuartusII がもつ Mega Wizard の機能により，乗算器，加減算器を実現した．

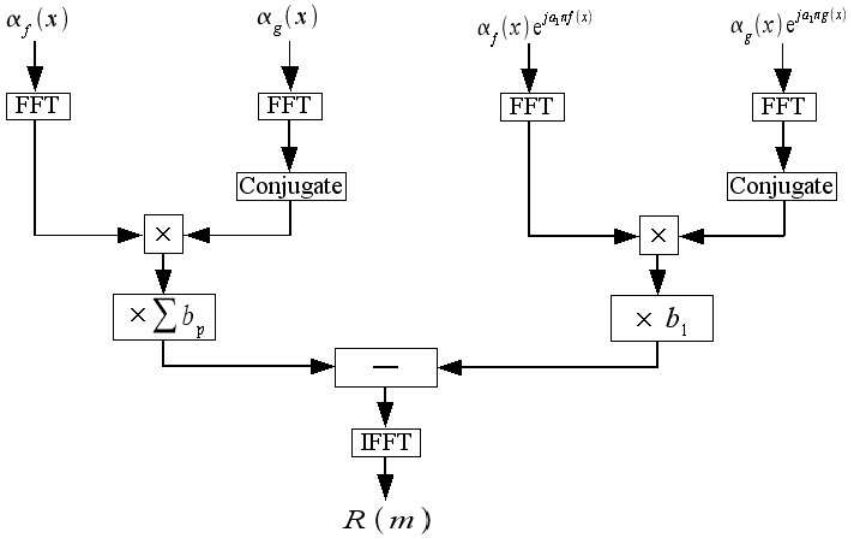

図 4: 画像比較の流れ図

バタフライ演算の流れは，以下のとおりである．図 6 にバタフライ演算の様子を示す．

1. 複素数 W_N の値を予めレジスタ W_{Nr} と W_{Ni} に保存しておく．
2. メモリから 2 つのデータとして，それぞれの実部虚部合せて 4 つのデータを読み込み FPGA 内のレジスタ x_0, x_1, x_2, x_3 に保存する．
3. これらのデータから，$(x_2 + jx_3)(W_{Nr} + jW_{Ni})$ を求め，一時レジスタ t_0, t_1 に保存する．
4. $(x_0 + jx_1) + (t_0 + jt_1)$ を求め，その結果をレジスタ X_0, X_1 に保存する．
5. レジスタ X_0, X_1 の内容を，定められたメモリのアドレスに格納する．
6. $(x_0 + jx_1) - (t_0 + jt_1)$ を求め，その結果をレジスタ X_0, X_1 に保存する．
7. レジスタ X_0, X_1 の内容を，定められたメモリのアドレスに格納する．

このように 1 つのバタフライ演算につき，実数の乗算 4 回と実数の加減算が 6 回必要である．1 回の乗算には 5 クロック，1 回の加減算には 8 クロック必要である場合には 68 クロックで 1 つのバタフライ演算を実行できる．

M 点の 1 次元 FFT を実装するためには $\frac{M}{2}log_2M$ 回のバタフライ演算が必要である．さらに $M \times N$ 点の 2 次元 FFT を実現するためには $\frac{MN}{2}(log_2M + log_2N)$ 回のバタフライ演算が必要となる．本稿では後述するように 128×256 の大きさの画像の合成を行なうので，245760 回のバタフライ演算が必要となる．本手法ではこのような 2 次元の FFT を 4 回必要とする．その結果，FFT 以外の処理を考慮して約 3 秒の処理時間が必要となる．

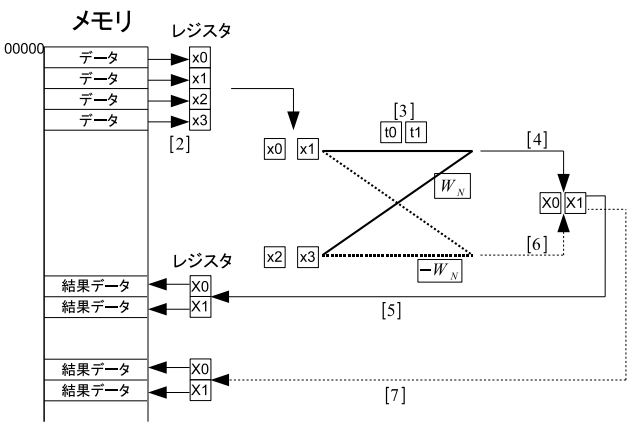

図 6: バタフライ演算の流れ ([ ] 内は本文中の処理の番号)

5. 実験

5.1 実験方法

3 つの NTSC カメラより撮影した入力映像①と入力映像②，入力映像②と入力映像③の映像を各々比較し合成する．上述のとおり 1 つの映像の大きさは 240×180[pixel] であるが，FFT を用いる場合，映像サイズを 2 のべき乗することを考慮して，本システムでは映像サイズとして，128×256[pixel] とする．ただし，x 方向には画像の半分の領域を合成に利用するものと考えた．なお，y 方向には 180 点の画素の外に 0 の値を埋めて 256 点の画像を生成した．

5.2 実験結果

実際に 3 つの NTSC ビデオカメラの映像図を図 7，その入力映像②と入力映像③を 128×240[pixel] の映像サイズに切り出した映像を図 8 に示す．さらに，図 8 の入力映像②と入力映像③の合成映像について，合成結果を図 9 に示す．

6. おわりに

本研究では，パノラマ映像生成の自動補正処理を行うために Fitch らの Fast Robust Correlation 手法のハードウエア実現を試みた．この手法は，FFT を用いることで計算量を低減することができた．今後の課題としては，補正処理の精度向上を目指したい．

図 7: パノラマ映像前の３つの入力映像

図 8: 入力映像②と入力映像③の合成前

図 9: 合成結果

参考文献

[1] Alistair J. Fitch，Alexander Kadyrov，William J. Christmas and Josef Kittler : Fast Robust Correlation，IEEE (2005)

[2] 長谷川 裕恭 : VHDL によるハードウェア設計入門，CQ 出版社 (2002)

[3] 森岡 澄夫 : HDL による高性能ディジタル回路設計，CQ 出版社 (2002)

[4] 大浦 拓哉 : FFT (高速フーリエ・コサイン・サイン変換) の概略と設計法 : http://www.kurims.kyoto-u.ac.jp/~ooura/fftman/index.html