〔非公開〕

ＴＲ－Ｃ－００６２

顔画像の特徴抽出に関する検討

木原　正人　Masasto Kihara
阿川　弘　Hiroshi Agawa
永嶋　美雄　Yoshio Nagashima

１９９１．３．１

ＡＴＲ通信システム研究所

テーマ　顔画像の特徴抽出に関する検討

実習期間　１９９１年１月８日～２月２８日

１　はじめに

現在、遠隔地間での会議ができるテレビ会議が注目され普及しつつある。それに伴い、臨場感のある仮想空間会議システムの研究が進められている［１］［２］。仮想空間会議システムは、遠隔地にいる人をＣＧによって、あたかも同一場所を共有している感覚を作り出す事で実現できる。

そのための技術課題として、顔画像の特徴抽出の前処理の一つに領域分割がある。［２］［３］［４］。今回の実務訓練では、画像処理手法について学習して、特に顔画像の領域分割にＷａｌｓｈ－Ｈａｄａｍａｒｄ変換［４］［５］［６］によるスペクトル画像を直接利用する手法に関して検討した。

２　顔画像の領域分割

２．１　画像の領域分割手法

画像を意味のある部分画像に分割することを画像の領域分割という。領域分割は、画像の特徴抽出の前段階処理として非常に重要なものである。領域分割手法は多数有るが、以下にいくつかについての簡単な説明を述べる。詳しくは、［３］［４］を参照していただきたい。

（１）　濃度ヒストグラムによる領域分割

濃度ヒストグラムよりしきい値を求め画像を２値化する。

しきい値決定法は多く提案されおり、濃度ヒストグラムのピークとピークの谷に決めるモード法や判別自動しきい値選択法、クラス間分散の差分に基づくしきい値設定法などがある。

（２）　クラスタリングによる領域分割

色度を属性値として用いる。各領域は、色が異なりかたまった分布をする。このかたまりをクラスタといい、これを求めることをクラスタリングという。クラスタリングには、階層的クラスタリングやＫ平均クラスタリングなどがある。

（３）　勾配弛緩法による領域分割

画像を構成する各画素の属性を確率に置き換えて、弛緩過程を更新していき

画像のコントラストを強調していく方法。

2．2　髪領域分割とその問題点

顔画像の領域分割アルゴリズムは既に提案されている［1］［2］。顔領域の分割処理には、入力されたＲＧＢ画像を明度、色相、彩度の知覚表色系に変換した画像を用いる。これは、光源、影の影響を受けにくく領域分割が容易になるからである。この画像のヒストグラム分布によるしきい値処理をすることで顔領域を分割することができる。

この手法は、日本人のように黒髪や白髪を持つ人に対しては容易に領域分割できるが、金髪などを持つ人などの場合には、一般的に難しい。これは、金髪が顔の色と近いことなどが原因といえる。

2．3　スペクトル画像による領域分割

前述の問題点を解決するために、本検討ではスペクトル画像をそのまま領域分割に利用する。スペクトル解析には、今回、Ｗａｌｓｈ－Ｈａｄａｍａｒｄ変換（ＷＨＴ）［5］［6］を用いた。後述するが、ＷＨＴはフーリエ変換と同じ様なスペクトル解析が可能である。

髪領域は、他の顔領域に比べて多く高周波数成分を含んでいる。この特徴を利用して小ブロックごとのスペクトル解析をおこない、そのスペクトル画像を直接処理することにより領域分割をする。髪領域の領域分割手法は、表色系画像を小ブロックごとにＷＨＴを行い正規化したスペクトル画像を求める。その画像を直接、適当なしきい値で2値化処理すると多くの周波数成分を含む髪領域と、低周波成分だけの他の領域が区別できる。後は、従来の2値画像処理により領域分割できる。この手法は、変換画像をそのまま利用するために処理を簡略化することができる。

処理過程の概要を図1に示す。

処理過程

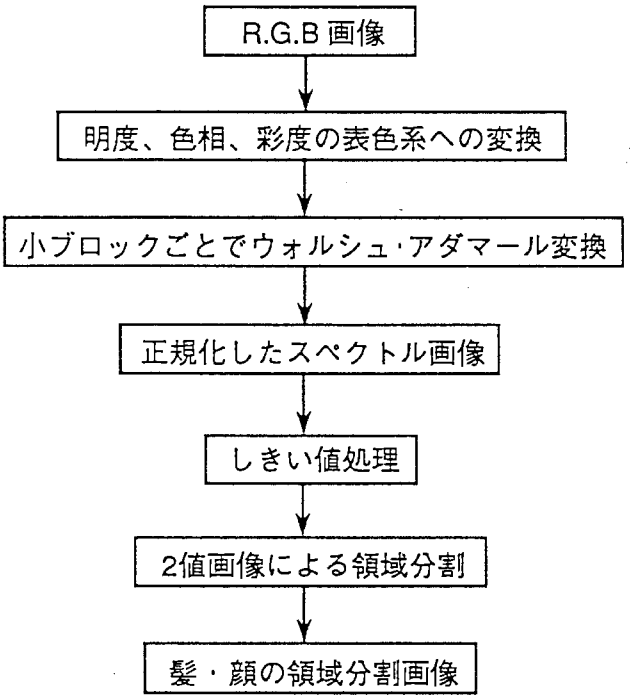


図１　顔画像の領域分割過程の概要

3　高速Ｗａｌｓｈ－Ｈａｄａｍａｒｄ変換（ＦＷＨＴ）

本検討では、スペクトル解析にＷＨＴを用いた。ＦＷＨＴは、ＦＦＴと同じく直交変換であり、高速でスペクトル解析が可能である。ここでは簡単にＦＷＨＴの計算法とＦＦＴとの違いについて説明する。また、ＦＷＨＴのプログラムを付録２に示す。

3．１　ＦＷＨＴの計算法

Ｗａｌｓｈ－Ｈａｄａｍａｒｄ変換を、数列の長さをＮ＝２^{n}として説明する。

Ｎ×ＮのＨａｄａｍａｒｄ行列Ｈ（ｎ）は、Ｈ（０）＝１として、

$$H(1) = \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix}$$

$$H(k) = \begin{vmatrix} H(k-1) & H(k-1) \\ H(k-1) & -H(k-1) \end{vmatrix}$$

$$k = 1,\ 2,\ \ldots,\ n$$

と定義される。

$$X(n) = \begin{vmatrix} x(0) \\ \cdot \\ \cdot \\ \cdot \\ x(N-1) \end{vmatrix} \qquad Y(n) = \begin{vmatrix} y(0) \\ \cdot \\ \cdot \\ \cdot \\ y(N-1) \end{vmatrix}$$

とすると、Ｘ（ｎ）のＷＨＴ　Ｙ（ｎ）は

$$Y(n) = H(n) * X(n)$$

で表される。

Ｈａｄａｍａｒｄ行列は、±１の値のみをとるので実際の計算は、加減算のみで計算できる。

高速変換ＦＷＨＴは、次のように計算できる。Ｎ＝２^{3}＝８とした場合、

$$\begin{vmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \\ \hline y(4) \\ y(5) \\ y(6) \\ y(7) \end{vmatrix} = \begin{vmatrix} H(2) & H(2) \\ H(2) & -H(2) \end{vmatrix} \begin{vmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ \hline x(4) \\ x(5) \\ x(6) \\ x(7) \end{vmatrix}$$

半分で区切って考えると、

$$x^1(l) = x(l) + x(l+4) \qquad l = 0,\ 1,\ 2,\ 3$$
$$x^1(l) = x(l-4) + x(l) \qquad l = 4,\ 5,\ 6,\ 7$$

とできるので、

$$\begin{vmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{vmatrix} = H(2) \begin{vmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{vmatrix}$$

$$\begin{vmatrix} y(4) \\ y(5) \\ y(6) \\ y(7) \end{vmatrix} = H(2) \begin{vmatrix} x_1(4) \\ x_1(5) \\ x_1(6) \\ x_1(7) \end{vmatrix}$$

と分けることができ、同様にn回繰り返すことで長さ2のＷＨＴの計算まで分解できる。この様にして高速変換ができる。

2次元の場合は、数列が$N_1 \times N_2 = 2^{n1} \times 2^{n2}$とすると、

$$Y(n_1, n_2) = H(n_1)\, X(n_1, n_2)\, H(n_2)$$

となり、１次元の計算を繰り返すことで求められ、高速変換も可能である。

上述の説明でのＨａｄａｍａｒｄ行列は、Ｈａｄａｍａｒｄ順序と呼ばれるもので、この他にも、波数順序に並んだＷａｌｓｈ順序がある。２つの関係をＮ＝８の場合で示す。１を"＋"、－１を"－"で表してある。

＋ ＋ ＋ ＋ ＋ ＋ ＋ ＋	＋ ＋ ＋ ＋ ＋ ＋ ＋ ＋
＋ － ＋ － ＋ － ＋ －	＋ ＋ ＋ ＋ － － － －
＋ ＋ － － ＋ ＋ － －	＋ ＋ － － － － ＋ ＋
＋ － － ＋ ＋ － － ＋	＋ ＋ － － ＋ ＋ － －
＋ ＋ ＋ ＋ － － － －	＋ － － ＋ ＋ － － ＋
＋ － ＋ － － ＋ － ＋	＋ － － ＋ － ＋ － ＋
＋ ＋ － － － － ＋ ＋	＋ － ＋ － － ＋ － ＋
＋ － － ＋ － ＋ ＋ －	＋ － ＋ － ＋ － ＋ －
Ｈａｄａｍａｒｄ順序	Ｗａｌｓｈ順序

３．２　ＦＷＨＴとＦＦＴの違い

ＦＷＨＴは、ＦＦＴでの正弦波をＨａｄａｍａｒｄ行列で見られるように、±１の２値で表せる矩形波で近似していると解釈できる。そのため、ＦＦＴが、周波数という概念で、複素数を扱うのに対して、ＦＷＨＴは、波数という概念で実数を扱う。また、計算は、ＦＦＴが乗算と加減算で行うのに対して、ＦＷＨＴは加減算のみで計算できる。

計算量を比較すると、Ｎ×Ｎの２次元変換に対して、

ＦＦＴ　：　$2N\log_2 N$の加減算と
　　$N\log_2 N$の乗算

ＦＷＨＴ　：　$2N\log_2 N$の加減算のみ

であるから、ＦＷＨＴの方が計算量が少なく変換できる。

４　実験

４．１　使用装置

実験には、ＦＷＨＴ等の数値計算処理にはＳＵＮ３を使用し、他の画像処理にはＶＩＣＯＭ画像処理装置を用いた。

4．2　サンプル顔画像

実験に用いた画像は、実際の人物の顔画像をテレビカメラにより撮影したものを利用した。光源は、実験室の室内灯だけで、特別な照明は行っていない。撮影条件を付録1に示す［7］。入力画像は、５１２×５１２画素／フレーム、8ビット／画素にデジタル化されてある。

今回の実験では、日本人（黒髪）と金髪の人をそれぞれ1人を対象として右横顔画像を使用した。

4．3　実験方法

①　ＲＧＢ画像を明度、色相、彩度（Ｌ’ＨＳ）画像に変換する［2］。

②　Ｌ’ＨＳ画像を小ブロックごとにＦＷＨＴした、スペクトル画像で領域分割を行う。

③　③より良好な結果が出た表色系画像を用いて、画像サイズに対する小ブロックサイズの依存性を検討する。
対象画像サイズは、５１２×５１２、３８４×３８４、２５６×２５６、１２８×１２８で、小ブロックサイズは、２×２、４×４、８×８に対して検討をする。

④　ＦＷＨＴした画像を用いた領域分割法との比較のためにＦＷＨＴせずに従来法［1］を用いてＬ’ＨＳ画像から直接領域分割をする。

①から④を日本人（黒髪）、金髪の顔画像の両方に対して行う。ＦＷＨＴした画像からの領域分割方法は、2．3で述べたように表色系画像をＦＷＨＴを用いて正規化したスペクトル画像を求める。その画像を直接に、最適なしきい値で2値化処理を行う。その後、ＶＩＣＯＭ画像処理装置の2値画像処理機能によって髪領域分割を行う。

予備実験より、エッジ方向を考慮してＦＷＨＴした小ブロックを転置してから領域分割を行った。

Ｌ’ＨＳ画像およびスペクトル画像データは、１６ビット／画素とした。

5　実験結果および考察

5．1　ＦＷＨＴしない画像からの領域分割

Ｌ’ＨＳ画像から直接領域分割を行うと、黒髪の顔画像に対しては、彩度画像でうまく領域分割ができた。これを図2．1に示す。（ａ）が、2値化処理した結果、（ｂ）が、（ａ）より抽出した顔領域の輪郭、（ｃ）が、領域分割結果である。　金髪の顔画像に対しては、どの画像に対してもうまく領域分割

ができなかった。２値化処理をすると図２．２のようにどの表色系画像でも２．１（ａ）の様にはならなず、はっきりと髪領域と顔領域の区別ができていない。

５．２　ＦＷＨＴしたスペクトル画像からの領域分割

Ｌ’ＨＳ画像を４×４ブロックごとにＦＷＨＴし、その画像を２値化処理をした画像を図３に示す。黒髪の顔画像では、色相画像と彩度画像とによい境界線が見られた。金髪の顔画像では、色相画像がよい結果であった。

これらの画像から領域分割を行うと、色相画像において黒髪、金髪の顔画像ともに領域分割ができた。これは、色相画像が明度画像等と違い、光源や影の影響を受けにくいことなどが理由に考えられる。

また、黒髪の顔画像に関しては、彩度画像からも領域分割ができた。

図４は、ＦＷＨＴした色相画像からの領域分割の結果である。上段が黒髪、下段が金髪の顔画像でそれぞれ、小ブロックサイズが、２×２、４×４、８×８の場合の結果である。。

５．３　画像サイズと小ブロックサイズの依存性

この検討には、５．２より、黒髪、金髪ともによい結果が出た色相画像を用いた。

図５は、金髪の顔画像を画像サイズ２５６×２５６でブロックサイズを変えたときの結果である。ブロックサイズ８×８では、毛の生え際がうまく出ずおかしな結果になっている。画像サイズに対して、ブロックサイズが大きいために荒い処理しかできないためである。また、２値化処理をした時点で輪郭部分の成分がブロックで現われ最後までつながって残ってしまった。

図６は、黒髪の顔画像をブロックサイズ２×２として画像サイズを変えたときの結果である。どの画像も良好な領域分割ができている。

図７は、金髪の顔画像をブロックサイズ８×８として画像サイズを変えたときの結果である。画像サイズが、小さくなると領域分割がうまくできなくなった。

これら、領域分割結果の評価を表１に示す。評価方法は、領域分割処理結果を原画像と見比べ、うまく領域分割ができているかを主観的に判断した。これは、個人の主観で判断したもので定量的ではないが一つの評価方法として用いた。評価は、３段階に分け以下のようにした。

○　：　良好な領域分割ができた
△　：　一応、領域分割ができた
×　：　領域分割ができなかった

表１　画像サイズと小ブロックサイズを変化したときの領域分割の評価

髪の色	画像サイズ	小ブロックサイズ		
		２×２	４×４	８×８
日本人（黒髪）	１２８×１２８	○	△	×
	２５６×２５６	○	○	△
	３８４×３８４	○	○	○
	５１２×５１２	○	○	△
金髪	１２８×１２８	○	△	×
	２５６×２５６	○	○	×
	３８４×３８４	○	○	△
	５１２×５１２	○	○	△

表１より、どのサイズの画像に対しても小ブロックサイズが２×２で良好な領域分割がされた。４×４は、画像サイズが小さくなるにつれて少々の難が出てくる。８×８にいたっては、画像サイズが小さいと、領域分割ができない。画像サイズに対して、ブロックサイズが大きくなると領域分割が荒くなり他の領域とくっついたりしてしまった。輪郭線付近には、多くの波数成分が存在するために輪郭線が髪領域につながってしまったりすることもあった。

結果としては、黒髪の顔画像の方が良好な領域分割をしやすかったが、黒髪、金髪のどちらの顔画像に対しても、本方法で領域分割が可能であった。小ブロックサイズを２×２とすれば、実験した範囲の画像サイズでは良好な領域分割ができ、画像サイズが大きければ４×４でも良好な結果が出た。実験した範囲の画像サイズでは、８×８では大きすぎた。

6　今後の課題

今回の検討は、黒髪、金髪の人の顔画像をそれぞれ１画像だけで、評価も主観的なものであった。今後は、もっと多くの画像に対して行い、抽出誤差の定量的な評価をする必要がある。そして、最適な画像サイズと小ブロックサイズの関係を決定する必要がある。

また、今回、スペクトル画像を求めるのにＷａｌｓｈ－Ｈａｄａｍａｒｄ変換を用いたが、フーリエ変換など、他の直交変換を用いた場合と精度、速度などを比較する必要がある。

領域の境界（輪郭線など）付近は、多くの周波数成分を含むために、スペクトル解析を行うと成分がエッジのように出てくる。この特徴を活かした顔画像処理の検討をすることなどが今後の課題である。

7　むすび

顔画像の特徴抽出の前処理としての領域分割において、金髪は肌と色が近く領域分割が困難であった。そこで、髪領域に高周波成分が多いことに注目して、Ｗａｌｓｈ－Ｈａｄａｍａｒｄ変換を用いたスペクトル画像から直接に領域分割する方法を検討した。本手法は、Ｗａｌｓｈ－Ｈａｄａｍａｒｄ変換が、加減算だけで変換でき簡単であるとともにスペクトル画像をそのまま領域分割に利用しているために処理の簡略、スピード化が図れる利点を持っている。また、髪領域の高周波成分に注目しているために、髪や顔の色にとらわれる事なく行うことができた。

今後は、スペクトル解析画像をエッジ処理などの他の顔画像処理へ利用するなど、顔画像の特徴点抽出手法と組合せることが考えられる。。

謝辞

今回の実務訓練で研究の機会を与えていただき、御指導下さいましたＡＴＲ通信システム研究所知能処理研究室の方々に感謝致します。

参考文献

[1]永嶋、他：”顔の３次元モデルの自動生成法に関する検討”，信学技報 IE90-46（1990.9）

[2]H.Agawa,et.al:"Image Analysis for Face Modeling and Facial Image Reconstruction",Visual Communications and Image Processing '90, Vol.1360,pp1184-1197（1990）

[3]田村：”コンピュータ画像処理入門”，総研出版（1985）

[4]尾崎、谷口：”画像処理（第２班）”，共立出版（1988）

[5]有木：”信号・画像のデジタル処理”，産業図書（1980）

[6]R.C.Gonzalez,P.Wintz:"Digital Image Processing",Addison-Wesley Publishing Company（1977）

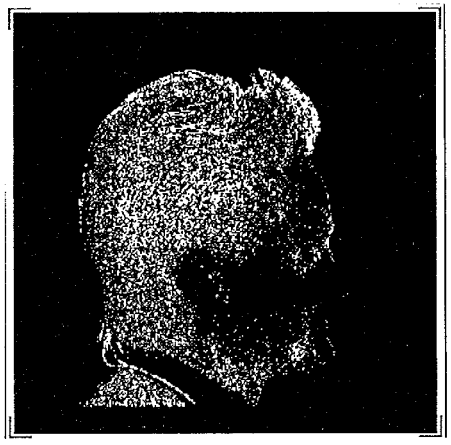
[7]金沢、他：”顔画像の特徴点抽出に関する研究”，ＡＴＲテクニカルレポート,TR-C-0044（1990.3）

（ａ）２値化処理結果

（ｂ）輪郭抽出結果

（ｃ）領域分割結果

図２．１　黒髪の彩度画像からの領域分割

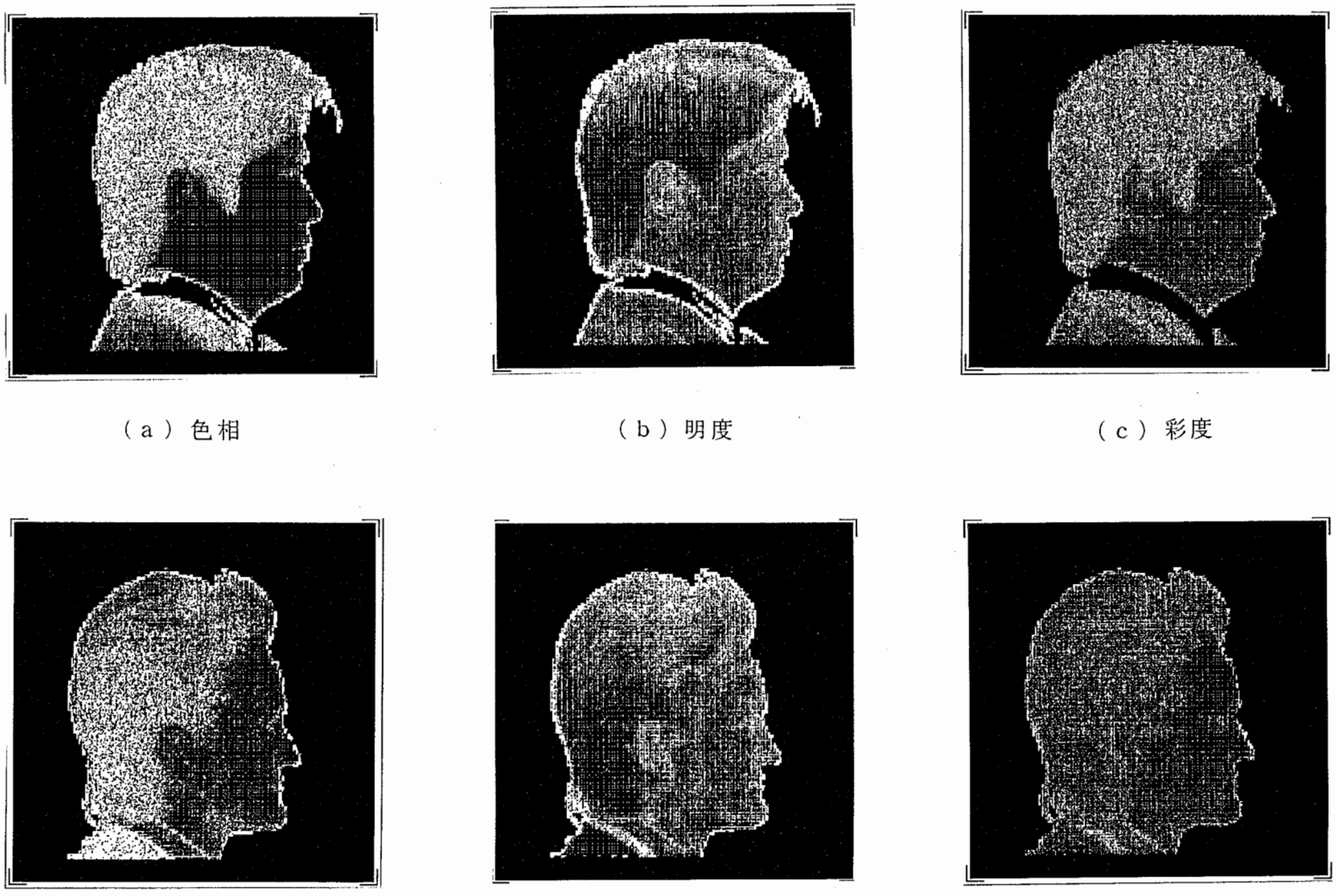


（ａ）色相

（ｂ）明度

（ｃ）彩度

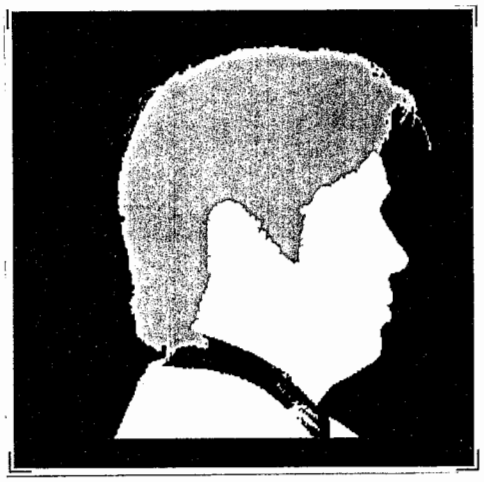
図２．２　金髪の顔画像の２値化処理結果



（ａ）色相　　（ｂ）明度　　（ｃ）彩度

（ａ）色相　　（ｂ）明度　　（ｃ）彩度

図３　ＷＨＴ画像の２値化処理結果
上段：黒髪　下段：金髪



（a） 2×2

（b） 4×4

（c） 8×8

（a） 2×2

（b） 4×4

（c） 8×8

図4　WHT画像からの領域分割結果
上段：黒髪　下段：金髪
図下数字はブロックサイズ

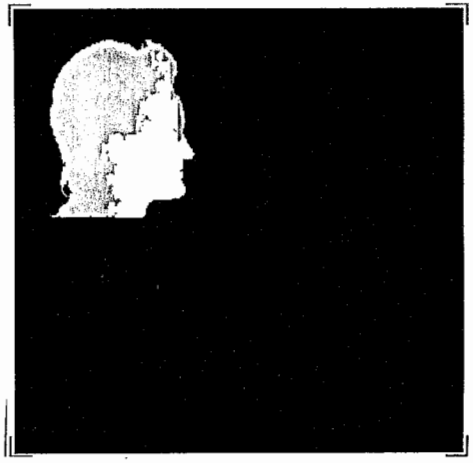
（a） 2×2

（b） 4×4

（c） 8×8

（a） 2×2

（b） 4×4



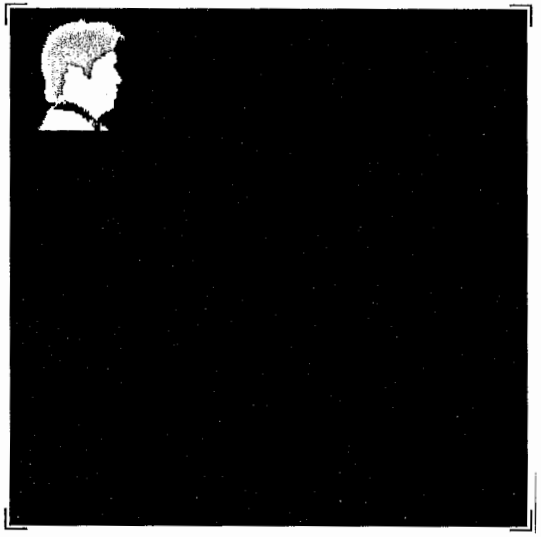
（c） 8×8

図5　ブロックサイズの検討（画像サイズ：256×256）
上段：黒髪　下段：金髪
図下数字はブロックサイズ

（a）５１２×５１２

（b）３８４×３８４

（c）２５６×２５６



（d）１２８×１２８

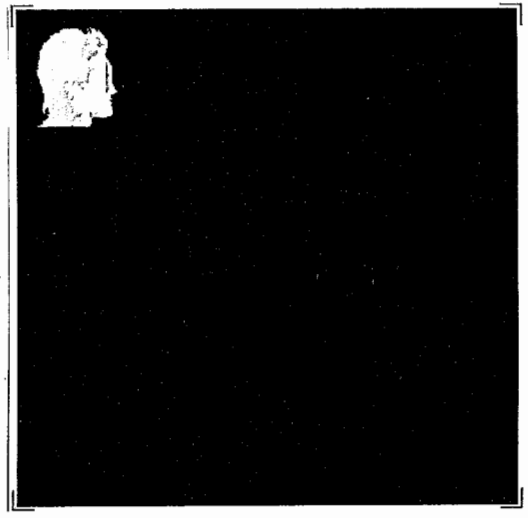
図６　画像サイズの検討１（ブロックサイズ：２×２）

図下数字は画像サイズ

（ａ）５１２×５１２

（ｂ）３８４×３８４

（ｃ）２５６×２５６



（ｄ）１２８×１２８

図７　画像サイズの検討２（ブロックサイズ：８×８）

図下数字は画像サイズ

付　録

付録１　サンプル画像の撮影条件

サンプル画像は、撮影されたデータを利用させてもらった［７］。撮影は、図８の様にセットして撮影されてある。光源は、実験室の照明だけである。点線部分は、天井の照明器具である。また、ホワイトボード付近に示されている値は、すべて上向きで計測された照度計の値である。

使用器具

1. カメラ　ＤＸＣ－７５０　、　ＳＯＮＹ
2. ＤＸＣ－７５０　ＣＡＭＥＲＡ　ＣＯＮＴＲＯＬ　ＵＮＩＴ、ＳＯＮＹ
3. ＦＭＵ－２０８８ＳＰ　、　Ｇｒａｐｈｉｃａ
4. デジタル照度計　Ｔ－１Ｍ　、　ミノルタ
5. ４０形白色蛍光灯３６［Ｗ］×３　、ナショナル
6. ホワイトボードは、ついたてに白紙を貼ったものを使用

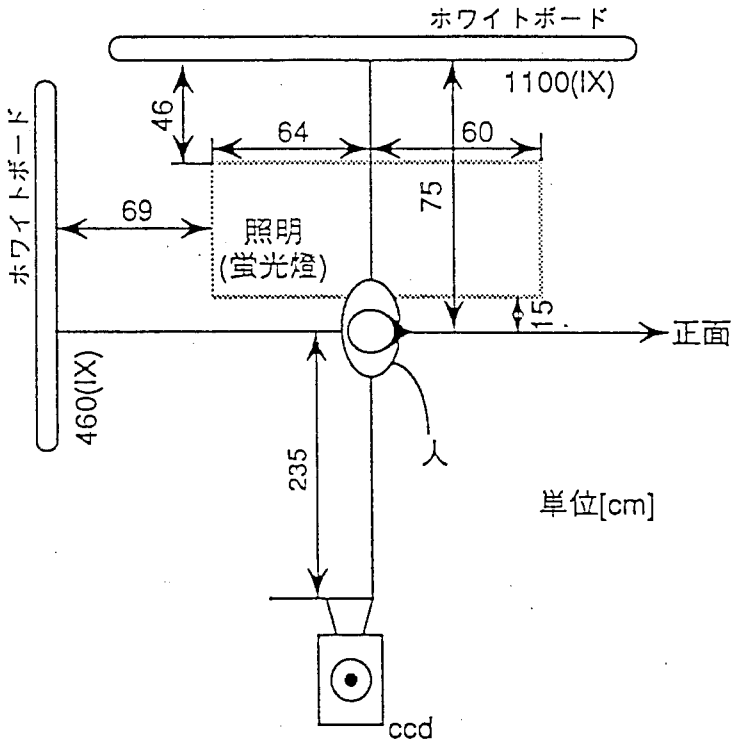


図８　サンプル画像の撮影条件

付録２　高速Ｗａｌｓｈ－Ｈａｄａｍａｒｄ変換プログラム

今回、作成した高速Ｗａｌｓｈ－Ｈａｄａｍａｒｄ変換プログラムを示す。

```
/*************************************************************************/
/* The 2-Dimensional Fast Walsh-Hadamard Transform (FWHT)                */
/*      File Name : fwht.c                                               */
/*      Include   : <stdio.h>,<math.h>,<sys/types.>,<sys/uio.h>          */
/*                      <sys/file.h>,<string.h>,<sys/fcntlcom.h>         */
/*      OS        : SUN3 os4.0.3                                         */
/*      Header    : "imgtype.h"                                          */
/*      Link      : "imgread.c","imgwrite.c"                             */
/*      Typedef   : IM_TYPE     ,In/Out Image Data Type                  */
/*                              Must Define In "imgtype.h"               */
/*      Parameter : Gloval                                               */
/*                  int im_size        ,Image Size                       */
/*                  double x[][]       ,Image Data For Calcuration       */
/*                  IM_TYPE im[][]     ,Input Image Data Type            */
/*                : Local Variable Name                                  */
/*                  int N              ,Block Size                       */
/*                  int b_size         ,N = 2^b_size                     */
/*                  int t              ,Transposition Parameter          */
/*                  int inv            ,Inverse FWHT Parameter           */
/*                  int nor            ,Normarise Parameter              */
/*                  int spec           ,Spectrum Parameter               */
/*                  int opt            ,Similar Optical Parameter        */
/*                  int order          ,Orederd Parameter                */
/*                  char infile[80]    ,Input Image File Name            */
/*                  char outfile[80]   ,Output Image File Name           */
/*************************************************************************/

#include <stdio.h>
#include <math.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <sys/file.h>
#include <string.h>
#include <sys/fcntlcom.h>
#include "imgtype.h"

/**********  Gloval Variable ***********************************************/

int im_size = 512;
double x[512][512];
IM_TYPE im[512][512];


/*************************************************************************/
/********** Main Top *****************************************************/

void main(argc,argv)
int argc;
char *argv[];
{
  void setpara();
  void p_change();
  double fwht2();
  void normarise();
  void optical();
  void usage();

  double max;
  int   N, b_size, t, inv, nor,
        spec, opt, order;
  char  infile[80],outfile[80];


                                        /***** Argument reaad *****/
  (void)setpara(argc,argv,&N,&b_size,&t,&inv,&nor,&spec,&opt,&order,infile,outfi
le);
```

```
                                        /***** Image Read From File *****/
  (void)printf("Reading Imagefile : %s¥n",infile);
  if( imgread(infile,(int)im_size,&im[0][0]) != 0 ) {
        (void)printf("Image Read Error in imgread, Exit!!¥n");
        (void)usage();
   }

                                        /***** TransPosition *****/
  (void)p_change( N, b_size, order, t );

                                        /***** 2-d(FWHT) *****/
  if( ( max=fwht2( N, b_size, inv ) ) <= 0. )
    (void)printf("Caution !! Maximum Value : %lf¥n",max);

                                        /***** Normarise *****/
  (void)normarise( max, nor, spec );

                                        /***** Similar Optical Position *****/
  if( opt==1 ) (void)optical(N);

                                        /***** Image Write To File *****/

  (void)printf("Writing file : %s¥n",outfile);
  if( imgwrite(outfile,(int)im_size,&im[0][0]) != 0 ) {
        (void)printf("Image Write Error in imgwrite, Exit!!¥n");
        (void)usage();
   }
}

/************************ main end ************************************/
/**********************************************************************/



/**********************************************************************/
/******************** Set Parameter ***********************************/

void setpara(argc,argv,N,b_size,t,inv,nor,spec,opt,order,infile,outfile)
int argc;
char *argv[];
int *N,*b_size,*t,*inv,*nor,*spec,*opt,*order;
char infile[],outfile[];
{
  int i,j;
                                /****** Default Parameter ******/
  *N = 4;                       /**       Block Size          **/
  *b_size = 2;                  /**       N = 2^b_size        **/
  *t = 0 ;                      /**  Transposition Parameter  **/
  *inv = 0;                     /**  Inverse FWHT Parameter   **/
  *nor = 1;                     /**    Normarise Parameter    **/
  *spec = 1;                    /**    Spectrum Parameter     **/
  *opt =0;                      /** Similar Optical Parameter **/
  *order = 0;                   /**     Orederd Parameter     **/

  (void)strcpy( infile , "image.im");
  (void)strcpy( outfile , "image.fwht");

                                        /***** Argument Read *****/
  i=1;
  while( i<argc ) {
    if( argv[i][0] == '-' ) {
      switch( argv[i][1] ) {
        case 'b' : if( ( argc==i+1) ||  ( *N = atoi(argv[++i]) ) <= 0 ) {
                     (void)printf("Argument Error in setpara : Block Size ¥n");
                     (void)usage();
```

```
                    }
                    if( *N > im_size) {
                      (void)printf("Argument Error in setpara : Lage Block Size =
%d¥n",N);
                      (void)usage();
                    }
                    break;
        case 'i' : if( ( argv[i+1][0] != '-' ) || ( argc != i ) )
                      (void)strcpy(infile,argv[++i]);
                    else {
                      (void)printf("Argument Error in setpara : Input File Name¥n
");
                      (void)usage();
                    }
                    break;
        case 'o' : if( ( argv[i+1][0] != '-' ) || ( argc != i ) )
                      (void)strcpy(outfile,argv[++i]);
                    else {
                      (void)printf("Argument Error in setpara : Output File Name¥
n");
                      (void)usage();
                    }
                    break;
        default  : j=1;
                    while( j<strlen(argv[i]) ) {
                      switch( argv[i][j] ) {
                        case 'W' : *order = 0;
                                   break;
                        case 'H' : *order = 1;
                                   break;
                        case 't' : *t = 1;
                                   break;
                        case 'n' : *nor = 0;
                                   break;
                        case 'I' : *inv = 1;
                                   break;
                        case 'O' : *opt =1;
                                   break;
                        case 's' : *spec =0;
                                   break;
                        default :  (void)printf("Argument Error in setpara : Type
Miss >> %s¥n",argv[i]);
                                   (void)usage();
                                   break;
                      }
                      j++;
                    }
                    break;
      }
    }
    else {
      (void)printf("Argument Error in setpara : No Symbol '-'¥n");
      (void)usage();
    }
    i++;
  }

                                        /***** Parameter Display *****/

  *b_size = (int)(log((double)*N)/log(2.));
  (void)printf("***** Condition Table *****¥n");
  (void)printf("  Image Size   : 512*512¥n");
  (void)printf("  Block Size   : 2^%d = %d¥n",*b_size,*N);
  (void)printf("  Input File   : %s¥n",infile);
  (void)printf(" Output File   : %s¥n",outfile);
  (void)printf("   Orderd      : ");
```

```
  if( *order==1 ) (void)printf("%s¥n","Hadamard");
    else  (void)printf("%s¥n","Walsh");
  (void)printf("***** Option Condition *****¥n");
  if( *inv==1 ) (void)printf(" Inverse FWHT¥n");
  if( *t==1 ) (void)printf(" Column , Row TransPosition¥n");
  if( *opt==1 ) (void)printf(" Similar Optical Position¥n");
  if( *nor==1 ) (void)printf(" Normarise");
  if( *spec==1 ) (void)printf(" Spectrum Value");
  (void)printf("¥n¥n");

}


/*************************************************************************/
/************** Set Ordered and Column ,Row TransPosition ***************/

void p_change(N,n,order,t)
int N,n,order,t;
{
  int grey();

  int i,j,k,l;
  if( order==1 ) {                        /***** Hadamard Ordered *****/

    if( t!=1 )                            /*** Normal ***/
      for( i=0 ; i<im_size ; i+=N )
       for( j=0 ; j<im_size ; j+=N )
        for( k=0 ; k<N ; k++ )
         for( l=0 ; l<N ; l++ )
           x[i+k][j+l] = (double)im[i+k][j+l];

    else                                  /*** Transposition ***/
      for( i=0 ; i<im_size ; i+=N )
       for( j=0 ; j<im_size ; j+=N )
        for( k=0 ; k<N ; k++ )
         for( l=0 ; l<N ; l++ )
           x[i+l][j+k] = (double)im[i+k][j+l];
  }
  else {                                  /***** Walsh Ordered *****/

    if( t!=1 )                            /*** Normal ***/
      for( i=0 ; i<im_size ; i+=N )
       for( j=0 ; j<im_size ; j+=N )
        for( k=0 ; k<N ; k++ )
         for( l=0 ; l<N ; l++ )
           x[i+grey(k,n)][j+grey(l,n)] = (double)im[i+k][j+l];

    else                                  /*** Transposition ***/
      for( i=0 ; i<im_size ; i+=N )
       for( j=0 ; j<im_size ; j+=N )
        for( k=0 ; k<N ; k++ )
         for( l=0 ; l<N ; l++ )
           x[i+grey(l,n)][j+grey(k,n)] = (double)im[i+k][j+l];
  }

}


/********************** Gray Code Generation ***************************/
/***** Function Used in p_change for Walsh Ordered *********************/

int grey(v,n)
int v,n;
{
  int i,r,t,bf,b;                 /** r : Return Value, other : Templral **/
```

```
  t=2;
  bf = (1&(v>>(n-1)));
  r = bf;
  for( i=n-2 ; i>=0 ; i-- ) {
    r += t*( bf^( b=( 1&( v>>i ) ) ) );
    bf = b;
    t *=2;
  }
  return r;
}


/*************************************************************************/
/******************** 2-Dimensional FWHT *********************************/
/***** max : Return Value , Maximum FWHT Value for Normarise ************/

double fwht2( N, n, inv )
int N,n,inv;
{
  void fwht1r();
  void fwht1c();

  int i,j,k;
  double max = 0.;

                                          /***** FWHT Par Block Size *****/
  (void)printf("Calculating FWHT¥n");
  for( i=0 ; i<im_size ; i+=N )
    for( j=0 ; j<im_size ; j+=N ) {
                                          /*** For Column ***/
      for( k=0 ; k<N ; k++ )
        fwht1c(N,n,k+j,i);
                                          /*** For Row    ***/
      for( k=0 ; k<N ; k++ )
        fwht1r(N,n,k+i,j);

                                          /*** Maximum Cofficient Search ***/

        if( ( i==0 ) && ( j==0 ) ) max = fabs(x[0][0]);
        if( max < fabs(x[i][j]) ) max = fabs(x[i][j]);

    }

                                          /***** Inverse Transform *****/
  if( inv==1 ) {
    (void)printf("Inverse FWHT¥n");
    max = max/(double)(N*N);
    for( i=0 ; i<im_size ; i++ )
      for( j=0 ; j<im_size ; j++ )
        x[i][j] = x[i][j] / (double)(N*N);
  }

  return max;
}


/****************** 1-Dimensional FWHT For Column ************************/
/**** l : Column Point, p : Left Upper Row Point of Block, in image ******/

void fwht1c(N,n,l,p)
int N,n,l,p;
{
  int i,ii,j,k;
  int step;
  double xx;
```

```
  step = N/2;

  for( k=1 ; k<=n ; k++ ) {

    for( j=p ; j<N+p ; j+=step*2 )
      for( i=j ; i<j+step ; i++ ) {
        ii = i + step;
        xx = x[i][l];
        x[i][l] = x[i][l] + x[ii][l];
        x[ii][l] = xx - x[ii][l];
      }
    step /= 2;
  }
}


/****************** 1-Dimensional FWHT For Row ***************************/
/**** l : Row Point, p : Left Upper Column Point of Block, in image *****/

void fwht1r(N,n,l,p)
int N,n,l,p;
{ int i,j,k; int ii,step;
  double xx;

  step = N/2;

  for( k=1 ; k<=n ; k++ ) {

    for( j=p ; j<N+p ; j+=step*2 )
      for( i=j ; i<j+step ; i++ ) {
        ii = i + step;
        xx = x[l][i];
        x[l][i] = x[l][i] + x[l][ii];
        x[l][ii] = xx - x[l][ii];
      }
    step /= 2;
  }
}


/*****************************************************************************/
/****************** Normarise and Spectrum Transform *************************/

void normarise(max,nor,spec)
double max;
int nor,spec;
{
  int   i,j;

  if( nor==1 ) {
    (void)printf("Normarise : Max Value = %lf¥n",max);
    max = 32767./max;
  }
  else
    max = 1.;
                                        /***** Spectrum *****/
  if( spec==1 ) {
    for( i=0 ; i<im_size ; i++ )
      for( j=0 ; j<im_size ; j++ )
        im[i][j] = (IM_TYPE)fabs(x[i][j]*max);
  }
  else {                                /***** Non Spectrum *****/
    for( i=0 ; i<im_size ; i++ )
      for( j=0 ; j<im_size ; j++ )
        im[i][j] = (IM_TYPE)(x[i][j]*max);
  }
```

```
}

/*****************************************************************************/
/****************** Change To Similar Optical Position ***********************/

void optical(N)
{
  int i,j,k,l;
  int ia,ib,ja,jb;                /** Temporal **/
  IM_TYPE m1,m2;                  /** Temporal **/

  for( i=0 ; i<im_size ; i+=N )
    for( j=0 ; j<im_size ; j+=N )
      for( k=0 ; k<N/2 ; k++ )
        for( l=0 ; l<N/2 ; l++ ) {
          ia = i+k;
          ib = ia+N/2;
          ja = j+l;
          jb = ja+N/2;
          m1 = im[ia][jb];
          im[ia][ja] = im[ib][jb];
          im[ib][jb] = m1;
          m2 = im[ib][ja];
          im[ib][ja] = im[ia][jb];
          im[ia][jb] = m2;
        }
}


/*****************************************************************************/
/********************* Usage *************************************************/

void usage()
{
  (void)printf("¥n Usage: fwht [ -b Block_size ] [ -i Infile ] [ -o Outfile ] ¥n
");
  (void)printf("                [ -WHIOnst ]¥n¥n");
  (void)printf("Block Size      Transform Small Block Size ( defalut : 4 )¥n");
  (void)printf("                  range : 1 to 512 (1,2,4,8,...,2^n ; n= 0 to 9)
]¥n");
  (void)printf("Infile          Input Image File Name ( defalut : 'image.im' )¥n
");
  (void)printf("Outputfile      Output Image File Name ( defalut : 'image.fwht'
)¥n");
  (void)printf("  -W / -H       Transform Value Orderd¥n");
  (void)printf("                        -W : Walsh / -H : Hadamard ( defalut : O
rder )¥n");
  (void)printf("  -I            Inverse FWHT¥n");
  (void)printf("  -t            Transposition of Samll Block Matrix¥n");
  (void)printf("  -O            Similar Optical Position of Transform Value¥n");
  (void)printf("  -s            Calculation of Normal FWHT Value( defalut : Spec
trum Value )¥n");
  (void)printf("  -n            Don't Normarise ( defalut : Normarise )¥n");
  exit(0);
}


/****************** Program End **********************************************/
/*****************************************************************************/
```

```
/*************************************************************************/
/*                                                                       */
/*   FuncName  : imgread( infile, size, img )                            */
/*                                                                       */
/*   Function  : read  image  file                                       */
/*                 1 pixel / sizeof(IM_TYPE) byte                        */
/*                                                                       */
/*                                                                       */
/*               return value   0 : normal end;                          */
/*                              1 : abnormal end;                        */
/*                                                                       */
/*   Parameter : Argument                                                */
/*                char infile[]        (i)  read image file name         */
/*                int  size            (i)  read image file size         */
/*               : Gloval                                                */
/*                IM_TYPE img          (o)  set buffer                   */
/*   define    :  IM_TYPE                read image data type          */
/*   include   : <sys/types.h>,<sys/uio.h>,<stdio.h>                     */
/*               <sys/file.h>,<string.h>,<sys/fcntlcom.h>                */
/*   Haeder    : "imgtype.h"                                             */
/*************************************************************************/

#include <stdio.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <sys/file.h>
#include <sys/fcntlcom.h>
#include "imgtype.h"

extern IM_TYPE im[512][512];

int imgread( infile, size, img )

char            infile[];
int             size;
IM_TYPE         *img;
{
        int iii;
        int ir;
        int filds;

        /****** FILE OPEN #######/

        filds = open( infile, O_RDONLY );
        if ( filds==-1 )
        {
                (void) printf ( "image file open error¥n" ) ;
                return(1);
        }

        /****** FILE READ #######/

        iii = sizeof(IM_TYPE)*size*size;
        ir  = read( filds,(char *)img, iii) ;
        if ( ir==-1 )
        {
                (void) printf ( "image file read error¥n" ) ;
                return(1) ;
        }

        /****** FILE CLOSE #######/

        (void) close ( filds ) ;
        return(0);
}
```

```
/*****************************************************************************/
/*   FuncName  : imgwrite( outfile, size , img )                             */
/*                                                                           */
/*   Function  : create image file from img[][] .                            */
/*                                                                           */
/*                                                                           */
/*               return value   0 : normal end                               */
/*                              1 : abnormal end                             */
/*                                                                           */
/*   Parameter : Argument                                                    */
/*                char outfile[]       (i)  output image file name           */
/*                int  size            (i)  output image file size           */
/*             : Gloval                                                      */
/*                IM_TYPE img          (o)  output image data                */
/*   define    :  IM_TYPE                Read Image Data Type                */
/*   include   : <sys/types.h>,<sys/uio.h>,<stdio.h>                         */
/*               <sys/file.h>,<string.h>,<sys/fcntlcom.h>                    */
/*   Haeder    : "imgtype.h"                                                 */
/*****************************************************************************/

#include <stdio.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <sys/file.h>
#include <sys/fcntlcom.h>
#include "imgtype.h"

extern IM_TYPE im[512][512];

int imgwrite( outfile,size ,img )

char            outfile[];
int             size;
IM_TYPE         *img;
{

        int  iii;
        int  ir;
        int  filds;
        static int mode = 0666;

        /****** FILE OPEN *******/

        filds = open( outfile, O_CREAT|O_TRUNC|O_WRONLY,mode );
        if ( filds==-1 )
        {
                (void) printf ( "Output image file open error¥n" ) ;
                return(1);
        }

        /****** DATA WRITE *******/

        iii = sizeof(IM_TYPE)*size*size;
        ir = write ( filds,(char *)img, iii ) ;
        if ( ir==-1 )
        {
                (void) printf ( "Output image file write error¥n" ) ;
                return(1) ;
        }

        /****** FILE CLOSE *******/

        (void) close ( filds ) ;
        return(0) ;
}
```

```
/* Header of Image Data Type Define      */
/*       File Name        imgtype.h      */

typedef short IM_TYPE;
```