色素による類似画像の検索

執筆者：山村　一起

指導教官：新納　浩幸

平成１５年３月15日

目次

図目次

第１章　序論

１．１　はじめに

インターネットの真価はWWWの登場によって認められるようになった。写真、動画、グラフィックス、音声など、各種の情報を取り扱えるようになったからである。しかし、このようなマルチメディア技術が使われているにもかかわらず、何万ものウェブサイトから欲しいものを見つけ出すには、いまだにテキストや数値による検索に頼らざるをえない。もし、ある画像を持っていてそれが何の画像か知りたい時や、自分で絵を書いてそれと似た画像をウェブ上から見つけたい時、検索のキーワードとして画像を検索エンジンに渡すとそれと類似する画像をウェブ上から検索できるシステムが望まれている。

画像の検索には様々な手法が提案されているが、まだ決定的と呼べるものはない。本研究ではそれらの手法の中でも比較的うまくできる手法である、色素を特徴量としたアプローチ[1]を試み、色素の情報だけでどの程度類似性を判断できるかを調べる。

１．２　本論文の構成

第２章において画像同士の類似度を測るための画像の特徴量ベクトル化、ベクトル間の類似度の求め方について説明する。

第３章では第２章で述べた方法を用いて、複数の条件のもとで画像同士の類似度を求める実験とその考察について述べる。

第４章では今回作成したプログラムを用いて作成したＶＢによるインターフェースを紹介する。

第２章　特徴量ベクトル

本実験では画像データを特徴量ベクトル化して、それらのベクトル間の類似度を計算することによって、画像間の類似度を求めるという作業を行った。

ここではその過程を順に説明していく。

２．１　ｂｍｐファイル

本実験で用いた画像ファイルのフォーマット形式は、２４ｂｐｐのビットマップ（ｂｍｐ）である。ここではｂｍｐファイルの構造について説明する。

ｂｍｐはＷｉｎｄｏｗｓで用いられている標準の画像フォーマットで、基本的に無圧縮なため扱いやすい。

ｂｍｐファイルのデータは図２．１に示した通りに並んでいる。

オフセット（１０進数）	バイト数	内容
０００００	2	“ＢＭ”の２文字（ｂｍｐファイルの識別子）
０００２	4	ファイルのバイト数
０００６	2	予約（＝０）
０００８	2	予約（＝０）
００１０	4	画像データ本体へのオフセット ２６：下のヘッダサイズが１２の場合 ４０：下のヘッダサイズが４０の場合 １２２：下のヘッダサイズが１０８の場合
００１４	4	ｂｍｐ情報のヘッダサイズ １２：シンプルなヘッダ ４０：通常のヘッダ １０８：くどいヘッダ
００１８	4	画像の横ピクセル数
００２２	4	画像の縦ピクセル数
００２６	2	プレーン（レイヤー）の数 １：通常

００２８	2	ｂｐｐ（ピクセル当りのビット数）
００３０	4	圧縮モード ０：無圧縮のベタデータ １：ＲＬＥ８ ２：ＲＬＥ４ ３：ビットフィールド
００３４	4	データ本体の長さ
００３８	4	横方向の解像度
００４２	4	縦方向の解像度
００４６	4	パレットの数 ０：ｂｐｐで表されるＭＡＸの色数 それ以外：実際のパレット数
００５０	4	重要なパレットの数
００５４ ・ ・	・ ・ ・	データ本体（下記を参照）

図２．１　ｂｍｐファイルのフォーマット

画像データ内で、ユーザーが制御できる最小の単位をピクセルと言う。

画像データの本体であるピクセルデータは、画面上で下側のラインから上のラインへと格納されている。１ラインの中では左から右である。

２４ｂｐｐの場合、図２．２のように３バイト一組のデータがピクセルの数だけ並んでいる。その内容はそれぞれＢ（青の強さ）、Ｇ（緑の強さ）、Ｒ（赤の強さ）を示しており０～２５５の値を取る。この３つの数値からピクセルの色は決まる。

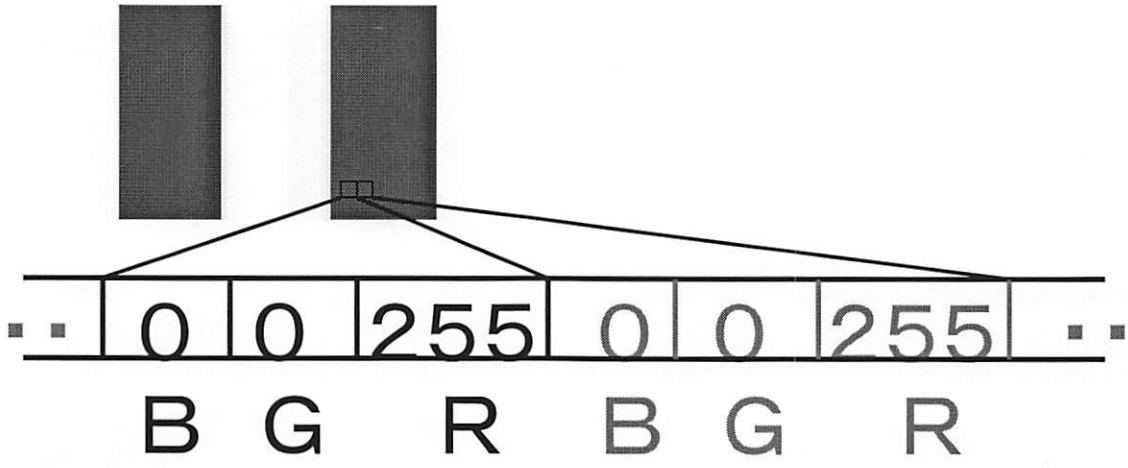


図２．２　画像データ本体

例として図２．３に示した２×２ピクセルのｂｍｐ画像のバイト列を示す。

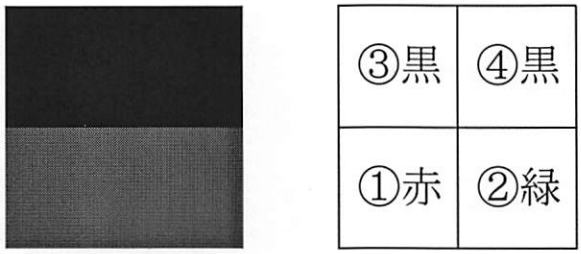


図２．３　ｂｍｐファイル例

このファイルの場合データは図２．４に示した通り並んでいる

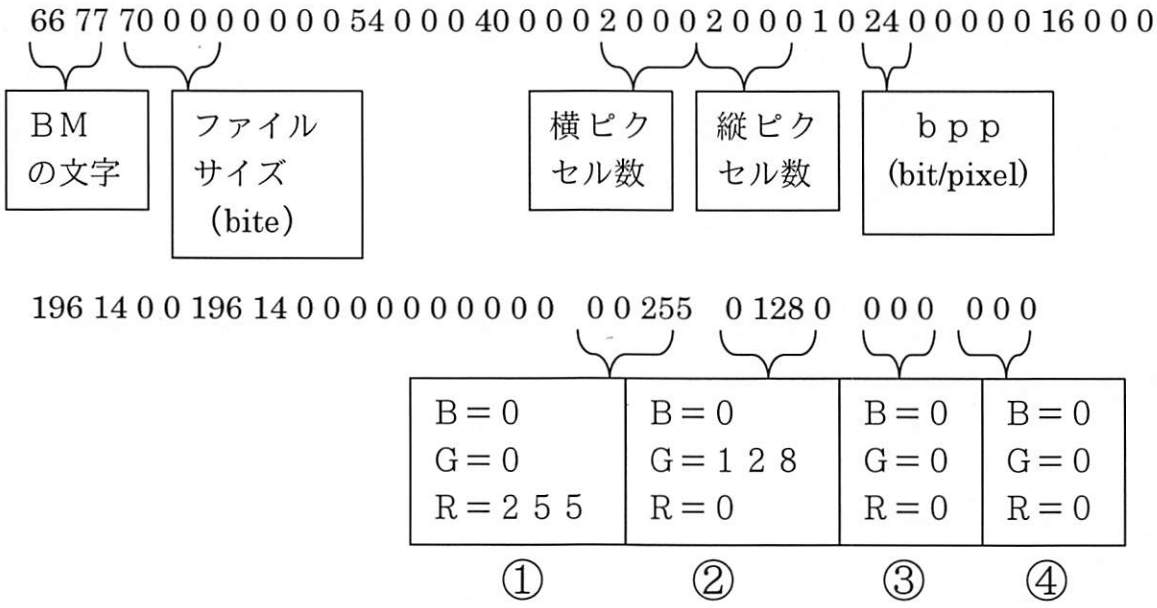


図２．４　バイト列

データの５４バイトまでは値は逆順に格納されている。例えば横ピクセル数の部分に５１，１２，１，０の順で並んでいたら、横ピクセル数は

$$51+(255\times12)+(255^2\times1)+(255^3\times0)=68136$$

である。

２．２　ＢＧＲとＨＳＶ

ＢＧＲ表色系がそれぞれ青、赤、緑を表すのに対し、ＨＳＶは色相（Ｈｕｅ）、彩度（Ｓａｔｕｒａｔｉｏｎ）、明度（Ｖａｌｕｅ）を表す。

色相は、色がスペクトルに沿ってどこにあるか、あるいはそれがどの「虹の色」であるかを示す。虹のように、始まって終わる色は赤である。色値はカラーサークル（色相環）（図２．５）に組織され、０度で赤、６０度でイエロー、そして時計回りに緑、シアン、青、マゼンタ、再び３６０度で赤と続く。

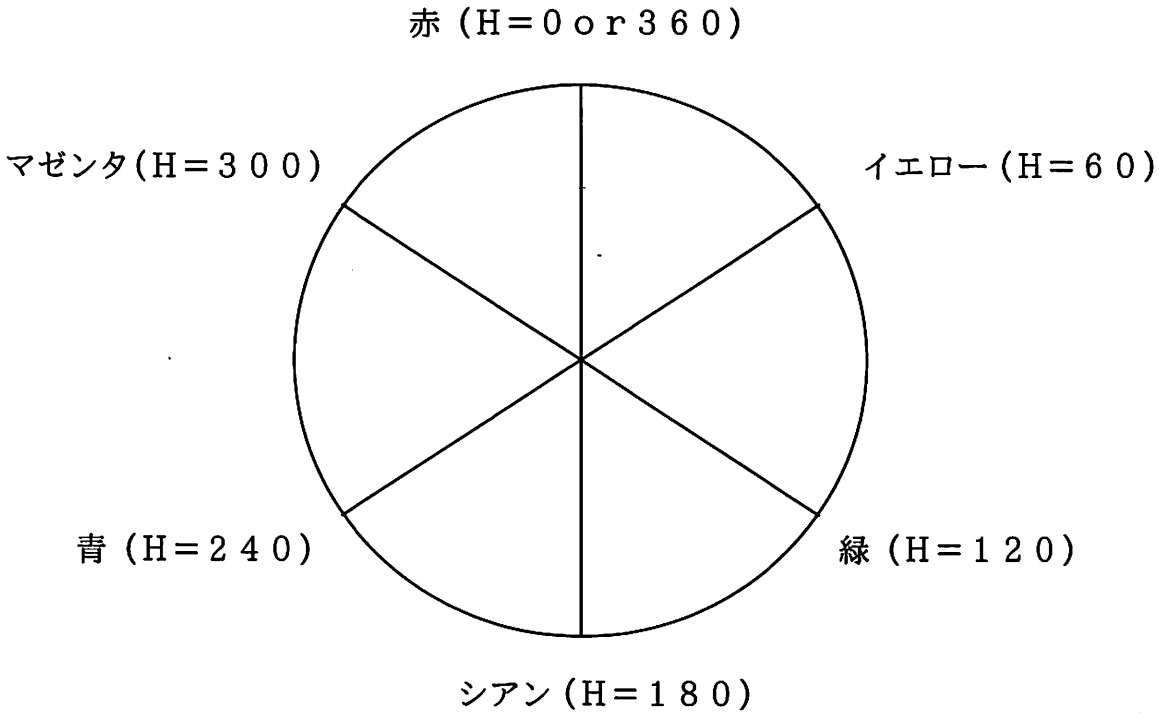


図２．５　カラーサークル

彩度は、いかに純粋か、つまり「派手な」色かである。彩度の値は０（グレイスケール）から１００（最大純度）まである。低い値は中間色、鈍い色を表す。高い値は、強い純粋な色を意味する。

明度は、明るさを表現する別の言い方である。０の値は完全な黒を意味している。１００の値は色が持つことのできる最も明るい値だが、最大値は白を意味しない。しかし（彩度が０でない限り）、最大値は色が特定の彩度で持つことのできる単に最も明るい値である。

ＨＳＶを使うと、色相と明度を分離できるので、それぞれに重みを持たせた検索が可能となる。例えば、明度の特徴ベクトルの類似度計算時の重みを減らすことで、影で暗く写った部分の影響を少なくすることができる。

これらのＨＳＶ要素は、ＢＧＲ要素を用いて図２．６の計算式から求める。

$$
\begin{aligned}
&V=\max(R,G,B),X=\min(R,G,B)\\
&S=(V-X)/V\\
&r=(V-R)/(V-X)\\
&g=(V-G)/(V-X)\\
&b=(V-B)/(V-X)\\
&R=V\text{の場合},H=(\pi/3)*(b-g)\\
&G=V\text{の場合},H=(\pi/3)*(2+r-b)\\
&B=V\text{の場合},H=(\pi/3)*(4+g-r)
\end{aligned}
$$

図２．６　ＢＧＲ→ＨＳＶ変換式

例として図２．３の②のピクセルのＨＳＶを求めてみる。

$B=0,G=128,R=0$を256進から100進に直すと

$B=0,G=50,R=0$

$V=50,X=0$

$$S=\frac{50-0}{50}\times 100=100$$

$$r=\frac{50-0}{50}=1,g=\frac{50-50}{50}=0,b=\frac{50-0}{50}=1$$

$G=V$より

$$H=(\pi/3)*(2+1-1)=120$$

よってピクセル②のＨＳＶは１２０，１００，５０と求まる。

２．３　特徴量ベクトル

画像のピクセルから抽出したＢＧＲＨＳＶ要素を、それぞれ１６次元のベクトルで表す。Ｂの場合、取りうる範囲０～２５５を

０～１５，１６～３１，３２～４７，・・・・・，２４０～２５５

と分け、それぞれを

Ｂ［０］，Ｂ［１］，Ｂ［２］，・・・・・，Ｂ［１５］

とする。Ｂの値が０だったら、Ｂ［０］＝１として、Ｂ［１］～Ｂ［１５］は０とする。他の要素についても同様の作業を行い、全ての要素を横に並べて９６次元のベクトルとする。

それらを横に並べたＢ［０］～Ｖ［１５］をそのピクセルの特徴量ベクトルP_iとする。

$$P_i = (\underbrace{B[0], B[1], B[2], \cdots, V[14], V[15]}_{96\text{次元}})$$

画像全体の特徴量ベクトルを足し算したのがその画像の特徴量ベクトルである。

$$P_1 + P_2 + \cdots + P_i + \cdots + P_n$$

（ｎ：画像のピクセル数）

ここで例として図２．３の画像の特徴量ベクトルを作ってみる。ピクセル①～④の特徴量ベクトル P_1, P_2, P_3, P_4 はそれぞれ

P_1 = (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1)

P_2 = (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0)

P_3 = (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

P_4 = (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

従ってこの画像の特徴量ベクトルは

$P_1 + P_2 + P_3 + P_4$
= (3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,3,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
3,0,0,0,0,1,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1)

となる。

２．４　コサイン尺度

２．３では画像情報を特徴量ベクトル化した。それらのベクトル間の類似度を計算することによって画像同士の類似度とする。

ベクトル間の類似度の定義としてはさまざまなものが考えられるが、その中でよく用いられているのがコサイン尺度（２つのベクトルのなす角度）である[2]。コサイン尺度は内積を正規化したもので、式は以下の通り。

$\vec{a},\vec{b}$ を m次元のベクトルとすると

● ベクトルの長さ

$$|a|=\sqrt{\sum_{i=1}^{m} a_i}$$

● 内積

$$\vec{a}\cdot\vec{b}=\sum_{i=1}^{m} a_i b_i$$

● コサイン尺度

$$\cos\left(\vec{a},\vec{b}\right)=\frac{a\cdot b}{|a||b|}$$

第３章　　実験

画像から特徴量を抽出してベクトル表現するプログラム、それらベクトルのコサイン尺度を求めるプログラムをＣ＋＋言語で作成した。またそのプログラムを繰り返し起動させるためのシェルプログラムも作成した。

実験の対象となる画像はＷＥＢ上から集めた図３．１，図３．２に示したような写真画像である。

● ６７種１３２枚の花画像

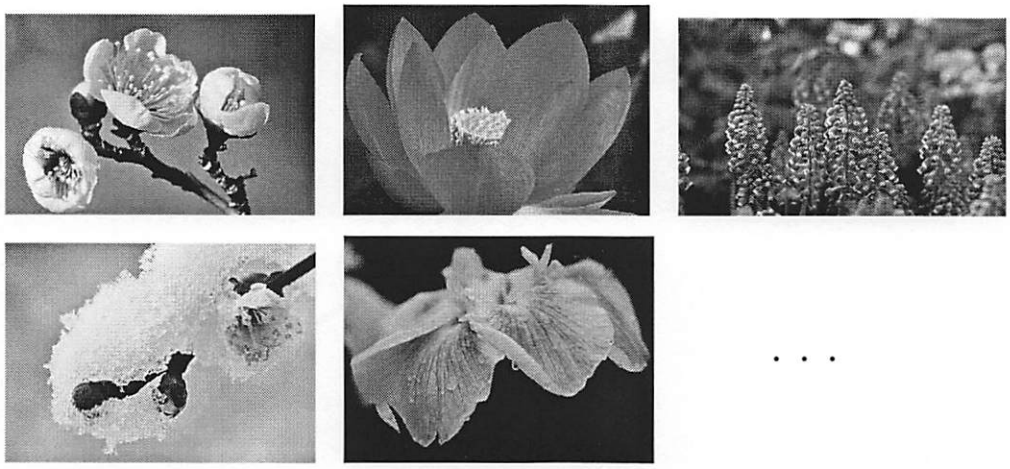


図３．１　花画像

● １１２枚のスニーカー画像

図３．２　靴画像

これらの画像を対象に、以下の実験を行った。

（実験１）１３２枚の花画像全ての組み合わせに対してＢＧＲＨＳＶ要素の９６次元でコサイン尺度を求める。

（実験２）１３２枚の花画像全ての組み合わせに対してＢＧＲ要素４８次元でコサイン尺度を求める

（実験３）１３２枚の花画像全ての組み合わせに対してＨＳＶ要素４８次元だけでコサイン尺度を求める。

（実験４）１３２枚の花画像全ての組み合わせに対して、画像を９分割したうち中央の部分だけを対象にしてＢＧＲＨＳＶ要素９６次元でコサイン尺度を求める。

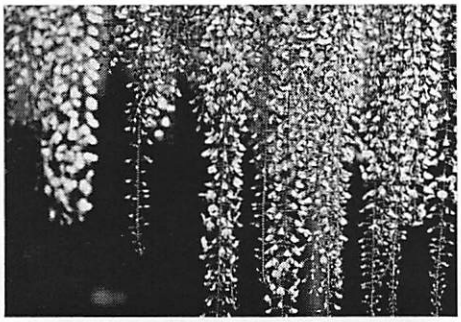
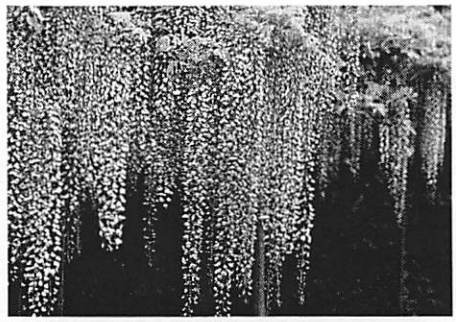
（実験５）１１２枚の靴画像を対象に、指定した画像に対する全画像のコサイン尺度をＢＧＲＨＳＶ、ＢＧＲ、ＨＳＶの３通りで計算する。

３．１　実験１（花画像ＢＧＲＨＳＶ）

今回花画像を選んだ理由は、花は色が豊富であり花の種類が色によって区別することができるので、本実験のテーマである色素による検索に適した材料だと思ったからである。

類似度の高さを計ることによって同種類の花画像を見つけ出すことを目的に、全ての画像の組み合わせに対しＢＧＲＨＳＶ要素９６次元でコサイン尺度を求めた。

コサイン尺度の高い上位３組と下位３組をそれぞれ図３．３，図３．４に示す。

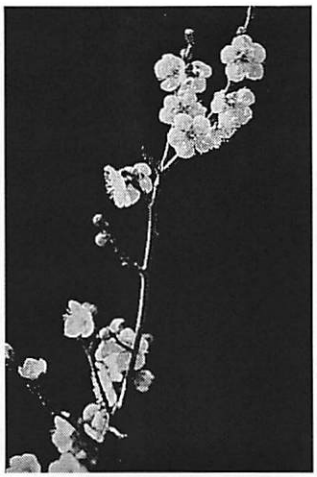



藤 01　　0.982759　　藤 03


ボリジ 02　0.954069　ボリジ 03

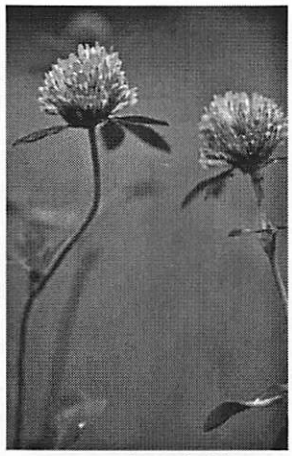

ベルガモット 01　0.949966　ボリジ 03

図３．３　ＢＧＲＨＳＶ上位３組




梅 03　0.0419713　はす 02


梅 03　0.0389996　菜の花 01

梅 03　0.0234859　赤詰草 01

図３．４　ＢＧＲＨＳＶ下位３組

第１位、第２位を見て分かる通り、同じ種類の花が高い類似度を持つという想定した通りの結果になった。色による画像認識はこの時点では成功したと言える。

しかし第３位には違う種類の組み合わせが現れている。この組み合わせは花ではなく背景が似ていることから類似度が高くなったと思われる。

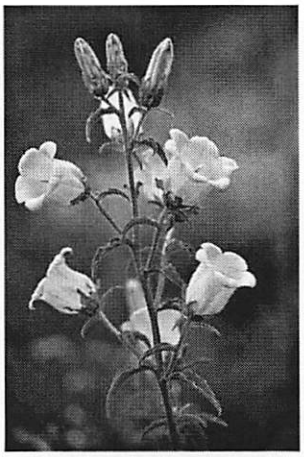
一方下位３組を見るとそれぞれ背景が極端に異なる組み合わせだった。

３．２　実験２（花画像ＢＧＲ）

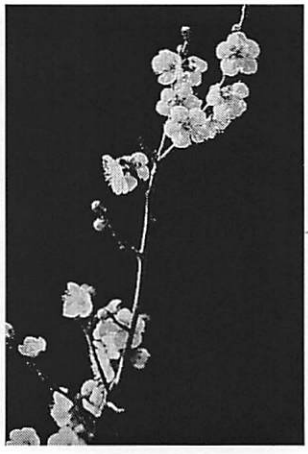
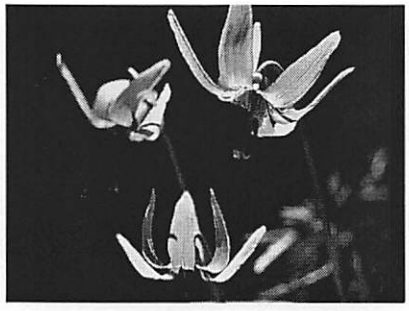
実験１ではＢＧＲＨＳＶ要素９６次元で特徴量ベクトルを作成しコサイン尺度を求めたが、作業の効率化を考えて特徴量の次元変化による違いを調べるために、花画像の全ての組み合わせに対しＢＧＲ要素４８次元だけで特徴量ベクトルを作成しコサイン尺度を求めた。

$$P_i = (\underbrace{B[0], B[1], B[2], \cdots, R[14], R[15]}_{48\text{次元}})$$

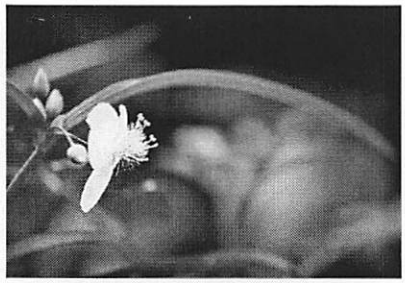
コサイン尺度の高い上位３組を図３．５に示す。




カンタベリ 01　0.992386　ハイビスカス 01

梅 03　0.991881　かたくり 02

菖蒲 01　0.991014　露草 01

図３．５　ＢＧＲ上位３組

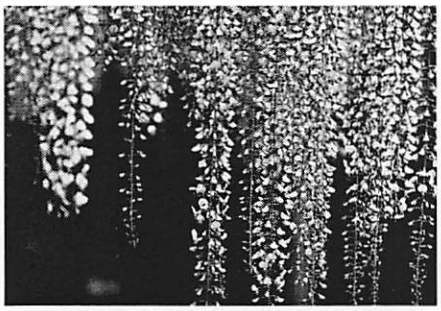
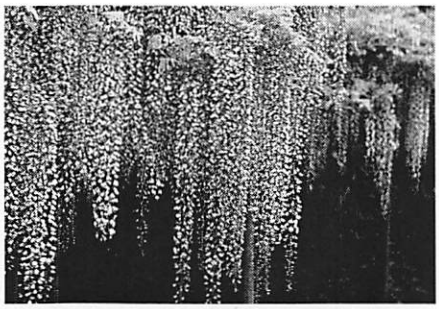
今回の結果は、上位３組に限らずどの組み合わせも視覚的に見るとコサイン尺度の割に似ているとは言い難い結果となった。

３．３　実験３（花画像ＨＳＶ）

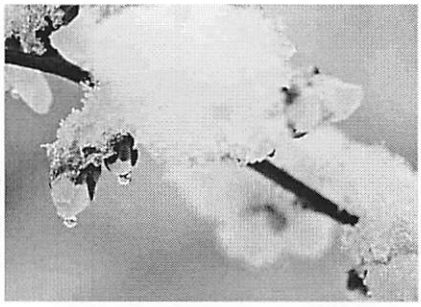
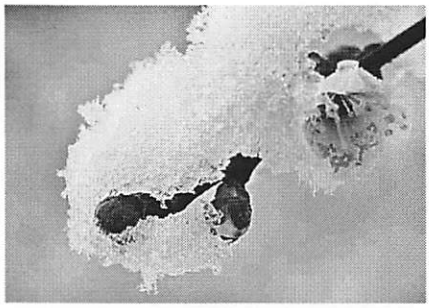
今度は実験２と逆に花画像の全ての組み合わせに対してＨＳＶ要素４８次元で特徴量ベクトルを作成しコサイン尺度を求めた。

$$P_i = (\underbrace{H[0], H[1], H[2], \cdots, V[14], V[15]}_{48\text{次元}})$$

コサイン尺度の高い上位３組を図３．６に示す。

藤 01　　0.985207　　藤 03

梅 04　　0.96576　　梅 05


ベルガモット 01　0.940578　ボリジ 03

図３．６　ＨＳＶ上位３組

この結果には、実験１の結果と似た傾向が見られ、視覚的にも似ていると思える組み合わせが上位に見られた。

３．４　実験４（花画像中央部分ＢＧＲＨＳＶ）

実験１で画像の背景の影響が大きく影響した結果となったため、対象物の重みが小さくなってしまい計算結果に響いた。そこで対象物は画像の真中に位置取りされていることを狙って、図３．７のように花画像を９分割したうちの中心部分だけを対象にＢＧＲＨＳＶ要素９６次元で特徴量ベクトルを作成しコサイン尺度を求めた。

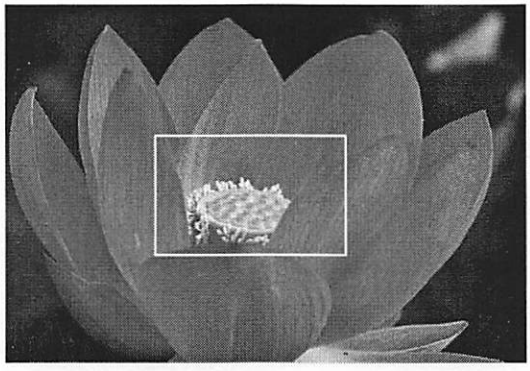
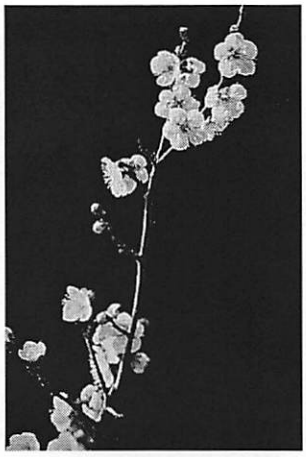
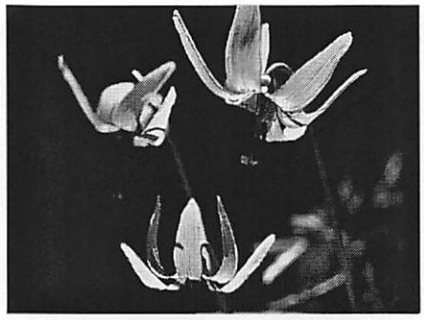
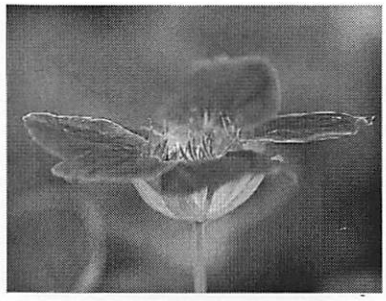


図３．７　画像の中心部分

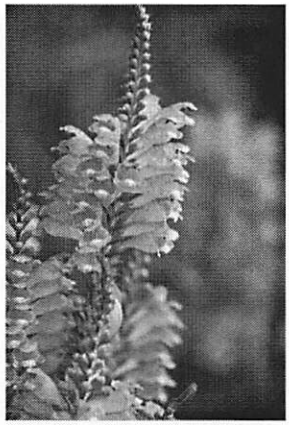
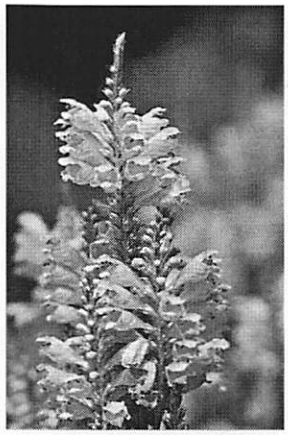
コサイン尺度の高い上位３組を図３．８に示す。

梅 03　0.919684　かたくり 02

キンセンカ 01　0.908841　ナンテン 02

カクトラノオオ 01　0.901062　トラノオオ 01

図３．８　中央部分ＢＧＲＨＳＶ上位３組

画像の対象個所を絞ることによって画像によって結果の違いが見られたが、それでも良い結果になったとは言えない。

３．５　実験５（靴画像）

ここまでは花画像を対象に実験を行ってきたが、今度はスニーカー画像１１２枚を対象に実験を行うことを試みた。

これらの画像の中から一枚を選び、その一枚に対する他画像のコサイン尺度をＢＧＲＨＳＶ、ＢＧＲ、ＨＳＶの３通りで特徴量ベクトルをそれぞれ作り求めた。(図３．９)

対象画像

ＢＧＲＨＳＶ　　ＢＧＲ　　ＨＳＶ

他の画像１１１枚

図３．９　実験５

３通りで行ったそれぞれのコサイン尺度上位３枚を図３．１０に示す。

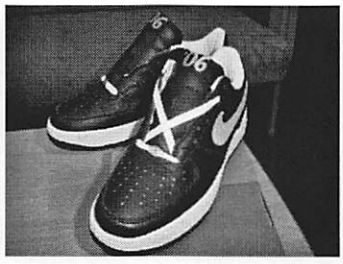
対象画像

●ＢＧＲＨＳＶ上位３枚



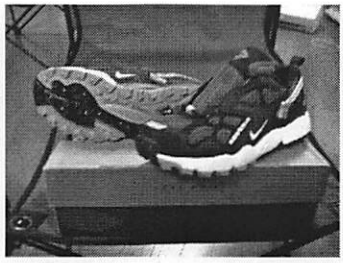
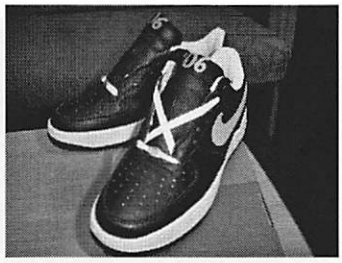
0.881333　　0.874246　　0.831927

●ＢＧＲ上位３枚


0.932345　　0.929451　　0.915383

●ＨＳＶ上位３枚

0.859592　　0.842546　　0.837956

図３．１０　靴画像実験結果

３通りの結果を見比べてみると、ＢＧＲＨＳＶとＨＳＶで行った結果に似た傾向が見られた。これは実験１～３で行った結果と同じことである。

上位の組み合わせの画像はどこが似ているのかと言うと、全体的な色合いや明るさであり、画像の構図や対象物の形ではなかった。

その点で視覚的に見て正しいと思える結果になったのはＢＧＲＨＳＶとＨＳＶであり、このことからＢＧＲ尺度だけで実験を行うよりもＨＳＶ尺度を使った方が良いと思われる。

第４章　考察

今回５つの実験を行い、それぞれの結果から以下のことが言える。

（実験１）花の色が同じということから、同種の花画像同士が高い類似度を持つという狙い通りの結果になった。しかし上位には別種の花画像同士もあり、それらの中には違う色の花の組み合わせもあった。その理由は花以外の背景にあって、画像中で対象物の花よりも背景の占める割合が大きいために背景による類似度が計算された。
（実験２）次元をＢＧＲの４８次元だけにすることにより作業速度は早まった。しかし計算結果の精度は実験１に比べ悪かった。
（実験３）同じくＨＳＶの４８次元だけにすることにより作業速度は早まった。精度も実験２のＢＧＲのように悪くはならず、実験１で行ったＢＧＲＨＳＶに似た結果になった。
（実験４）実験１において問題となった背景による影響は、画像によっては小さくなり結果に大きな違いが見られた。しかしこの方法では対象となる画像に制限がでてきてしまうために有効とは言えない。
（実験５）花画像から靴画像に変えて行ったが、花画像で行った結果と同様のことが言える。

本実験では画像同士の類似度を測るために、画像をベクトル化してコサイン尺度を用いた。この方法では画像全体の色の平均で類似度を求めているために、画像の構図や対象物の形までは判断することができなかった。図４．１の２枚の絵はまったく同じ画像ということになる。

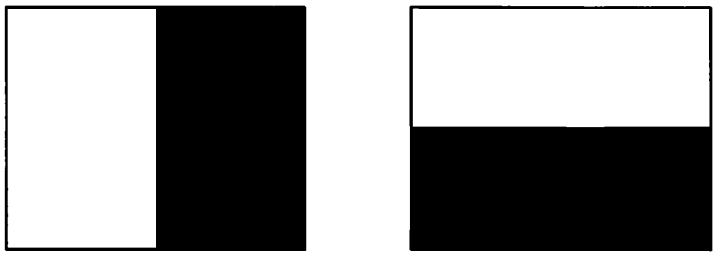


図４．１　同じ画像

画像検索としてはまだまだ精度の低い方法ということがわかった。実用性も考えて精度を上げるとするなら、画像の構図まで考慮して類似度を測るという方法が必要となってくる。

今後はその手法として新たな画像間距離ＥＭＤ（EarthMover'sDistance）[3]を適用したい。ＥＭＤは２つの分布間の距離尺度であり、一方の分布を他方の分布に変換するための最小コストにより距離を定義するという方法である。これは色の分布具合から計算を行うので、本実験の課題となった画像の構造も考慮された類似度計算を行ってくれると思う。

第５章　ＶＢによるインターフェース

誰もが気軽にデジカメで大量の写真を撮る時代になり、人によっては１日に１０枚のペースで集めていくと１年で数千枚の写真がたまってしまうことになる。画像ファイル数千、数万枚という量になってくると整理するのも一苦労になる。このような場合にも画像検索は役立ってくる[4]。

そこで本実験で作成したプログラムを利用して、Microsoft VisualBasic6.0 でインターフェースを作成した。

一つの選んだ画像と類似度の高い画像をディレクトリ内から見つけ出して、上位３枚を表示する。図５．１，図５．２，図５．３にその外見を示す。

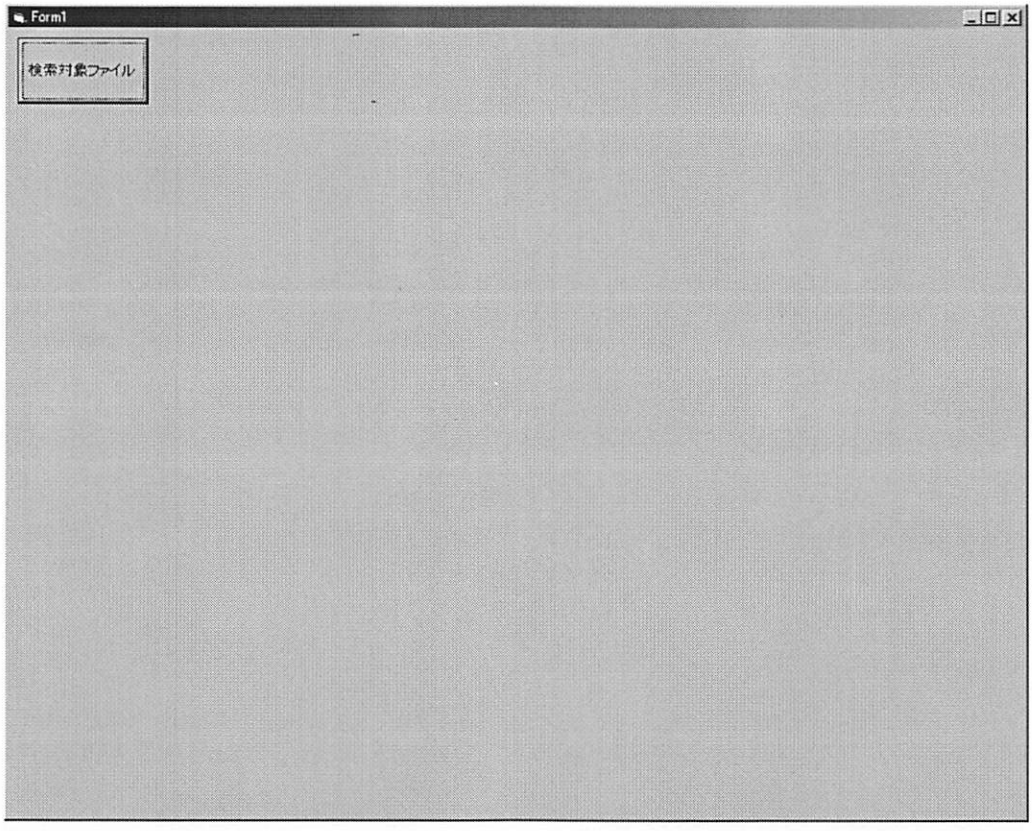


図５．１　インターフェース１

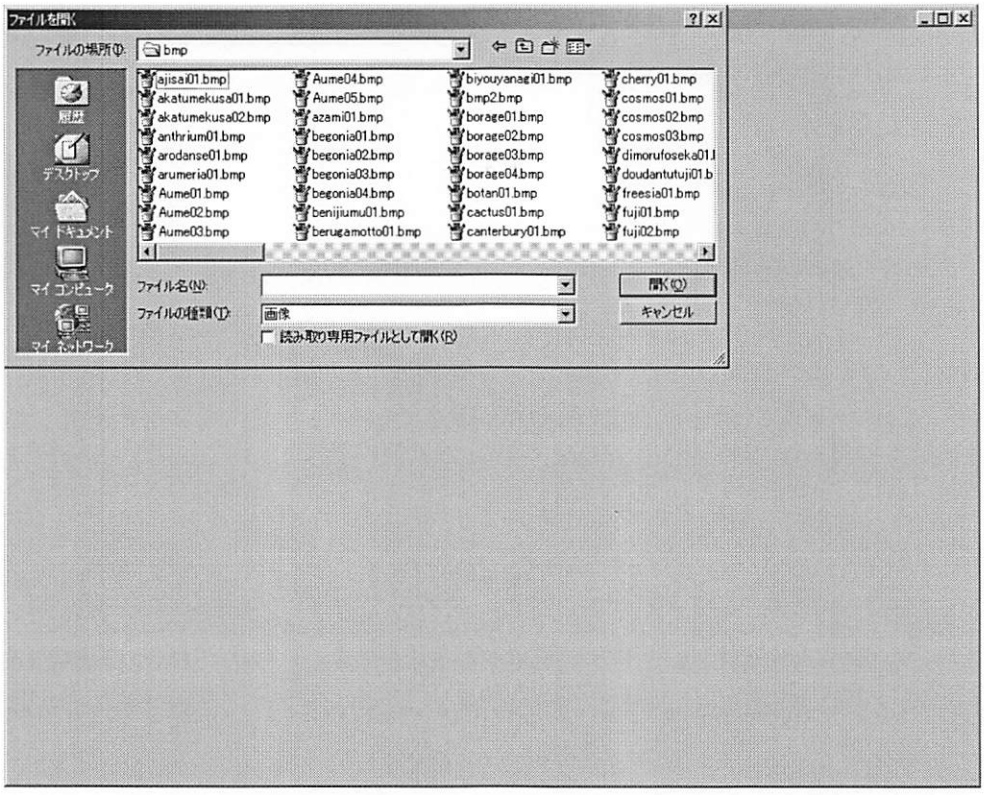


図５．２　インターフェース２

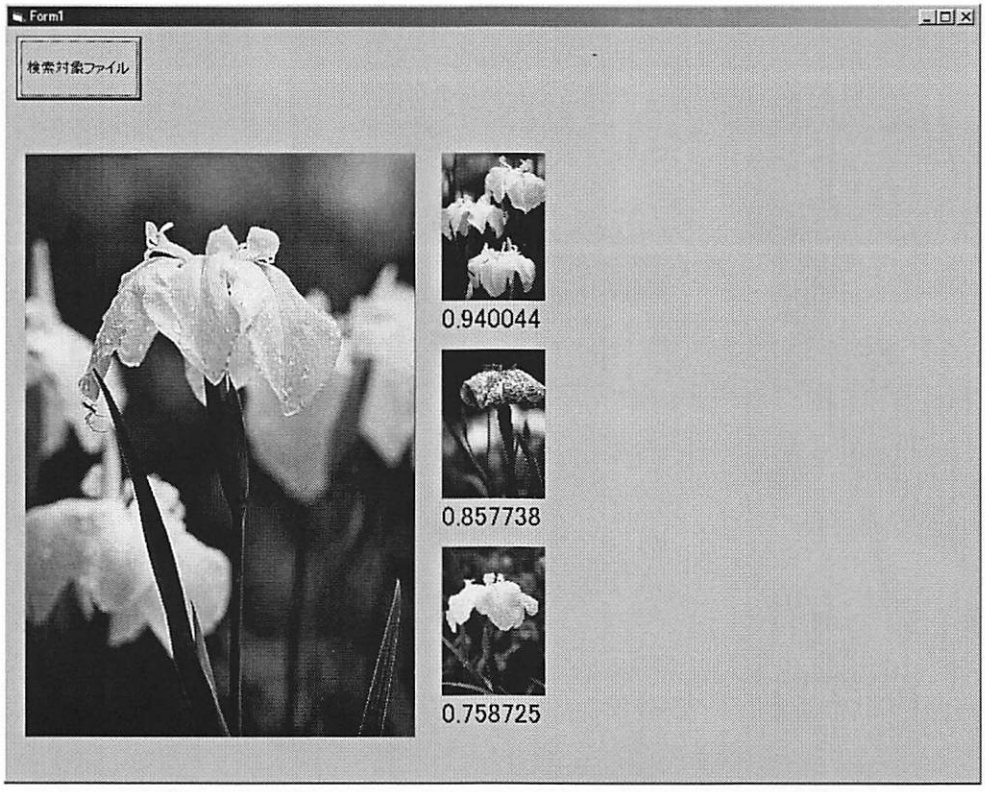


図５．３　インターフェース３

対象画像を指定してから他画像の類似度を計算して、画面表示するまでの処理速度にとても時間がかかるため実用にはまだ難しい。今後改良を加え、より使いやすくして多くの人に使ってもらえるプログラムにしたい。

第6章　結論

今回の実験では、色素の情報だけでどの程度画像の類似性を判断できるかを調べるために、花と靴の写真画像から特徴量ベクトルを計算して画像同士のコサイン尺度を求めた。

結果、ＢＧＲ尺度だけではうまくいかず、ＨＳＶ尺度を用いた方が良いことが分かった。ただし、ＨＳＶに更にＢＧＲ情報を加えＢＧＲＨＳＶにしても精度に変化はなかった。

本実験で用いた方法では対象物の判別まではできず、画像全体の色や明るさから類似性を判別していることを確認できた。

今後、新たな画像間距離を適用させて精度の高い検索を行いたい。

また、このプログラムを用いてＶＢによるインターフェースを作成した。実験を進めると同時にこちらも改良していきたい。

謝辞

本研究の遂行及び論文の作成に多大なご助言及び御指導を賜った新納浩幸教官（茨城大学システム工学科）に深い感謝の意を表します。

また、御指導を頂きましたシステム工学科計算機応用学講座の教官の方々にも深く感謝致します。

最後に、本研究を進めるにあたり助言、強力を頂きました同研究室の阿部修也氏（茨城大学大学院理工学研究科システム工学専攻)、高橋篤史氏（茨城大学大学院理工学研究科システム工学専攻)、紺野憲一氏（茨城大学システム工学科４回生)、藤枝雅一氏（茨城大学システム工学科４回生）に深く感謝致します。

参考文献

[1] 赤間浩樹,紺谷精一,鬼塚真,山室雅司,串間和彦: “データベース最前線-多次元空間索引と画像・音楽の内容検索-“,bit Vol32,No9,pp47-54,2000

[2] 北研二,津田和彦,獅々堀正幹:「情報検索アルゴリズム」,共立出版,pp.61,2001

[3] Y. Rubner, C. Tomasi, and L. J. Guibas. "A metric for distributions with applications to image databases", Proc. of IEEE International Conference on Computer Vision, pp. 59-66, 1998.

[4] “今月のコラム”,ASCII,Vol.26,#8 August 2002

付録　プログラムソースリスト

● ２枚の画像を読み込んでＢＧＲＨＳＶでコサイン尺度を計算

```
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>

////////////////////////////////////////////////////////////////
//HSV 変換関数
////////////////////////////////////////////////////////////////

void BGR_to_HSV(int B, int G, int R, int *h, int *s, int *v)
 {
   float H, S, V, X;
   float r = 0, g = 0, b = 0;
   int pi = 180;

   if(B > G)
    {
      if(B > R)
        V = B;
      else
        V = R;
    }
   else
    {
      if(G > R)
        V = G;
```

```
    else
      V = R;
  }

if(B < G)
  {
    if(B < R)
      X = B;
    else
      X = R;
  }
else
  {
    if(G < R)
      X = G;
    else
      X = R;
  }

if(V != X)
  {
    r = (V - R) / (V - X);
    g = (V - G) / (V - X);
    b = (V - B) / (V - X);
  }

if(R == V)
  H = (pi / 3) * (b - g);
else if(G == V)
  H = (pi / 3) * (2 + r - b);
else if(B == V)
  H = (pi / 3) * (4 + g - r);

if(H < 0)
  H = H + 360;
```

```
    V = (V / 256) * 100;
    X = (X / 256) * 100;

    if(V)
      S = ((V - X) / V) * 100;
    else
      S = 0;

    *h = H / 22.56;
    *s = S / 6.3;
    *v = V / 6.3;
  }

////////////////////////////////////////////////////////////////
//main 関数
////////////////////////////////////////////////////////////////

void main(int argc, char *argv[])
 {
    FILE *fp1, *fp2;
    char add1[100] = "c:¥¥bmp¥¥bmp¥¥";
    char add2[100] = "c:¥¥bmp¥¥bmp¥¥";
    unsigned int one, two;
    int i;
    long j;
    int h1, h2, h3, h4, w1, w2, w3, w4;
    int height, width, set;
    int B[16], G[16], R[16], H[16], S[16], V[16];
    int BB, GG, RR, h, s, v;
    int amari1, amari2;
    double length1, length2, Inner = 0.0;
    double inner[96];

    fp1 = fopen(strcat(add1, argv[1]), "r");
    fp2 = fopen(strcat(add2, argv[2]), "r");
```

```
//////////////////////////////////////////////////
//１個目ファイルデータの処理
//////////////////////////////////////////////////

    for(i = 0; i < 54; i++)
     {
        one = (unsigned int)fgetc(fp1);

        if(i == 18)
            w1 = one;                              //横 Pixel 数

        if(i == 19)
            w2 = one;

        if(i == 20)
            w3 = one;

        if(i == 21)
         {
            w4 = one;
            width = (pow(255, 3) * w4) + (pow(255, 2) * w3) +
                        (255 * w2) + w1;
         }

        if(i == 22)
            h1 = one;                              //縦 Pixel 数

        if(i == 23)
            h2 = one;

        if(i == 24)
            h3 = one;

        if(i == 25)
         {
            h4 = one;
```

```
            height = (pow(255, 3) * h4) + (pow(255, 2) * h3) +
                     (255 * h2) + h1;
        }

        if(i == 28)
        {
            if(one == 24)
              set = 3;
            else
            {
              cout << endl;
              exit(0);
            }
        }

        if(i == 46)
        {
            if(one)
              set = one;

            j = width * height * set + 54;
        }
    }

//画像データ本体の処理////////////////////////////////////

    for(int k = 0; k < 16; k++)
    {
        B[k] = 0;
        G[k] = 0;
        R[k] = 0;
        H[k] = 0;
        S[k] = 0;
        V[k] = 0;
    }
```

```
for(i = 54; i < j; i++)
 {
   one = fgetc(fp1);

   amari1 = (i - 54) % 3;
   amari2 = one / 16;

   if(amari1 == 0)
    {
      B[amari2]++;
      BB = one;
    }

   if(amari1 == 1)
    {
      G[amari2]++;
      GG = one;
    }

   if(amari1 == 2)
    {
      R[amari2]++;
      RR = one;

      BGR_to_HSV(BB, GG, RR, &h, &s, &v);
      H[h]++;
      S[s]++;
      V[v]++;
    }
 }

//内積計算//////////////////////////////////////////////

   length1 = 0;

   for(int b = 0; b < 16; b++)
```

```
 {
    length1 = length1 + pow(B[b], 2);
    inner[b] = B[b];
 }

for(int g = 0; g < 16; g++)
 {
    length1 = length1 + pow(G[g], 2);
    inner[16+g] = G[g];
 }

for(int r = 0; r < 16; r++)
 {
    length1 = length1 + pow(R[r], 2);
    inner[32+r] = R[r];
 }
for(int h = 0; h < 16; h++)
 {
    length1 = length1 + pow(H[h], 2);
    inner[48+h] = H[h];
 }

for(int s = 0; s < 16; s++)
 {
    length1 = length1 + pow(S[s], 2);
    inner[64+s] = S[s];
 }

for(int v = 0; v < 16; v++)
 {
    length1 = length1 + pow(V[v], 2);
    inner[80+v] = V[v];
 }

length1 = sqrt(length1);
```

```
    fclose(fp1);

////////////////////////////////////////////////////
//２個目ファイルデータの処理
////////////////////////////////////////////////////

    for(i = 0; i < 54; i++)
     {
        two = (unsigned int)fgetc(fp2);

        if(i == 18)
            w1 = two;                              //横 Pixel 数

        if(i == 19)
            w2 = two;

        if(i == 20)
            w3 = two;

        if(i == 21)
         {
            w4 = two;
            width = (pow(255, 3) * w4) + (pow(255, 2) * w3) +
                        (255 * w2) + w1;
         }

        if(i == 22)
            h1 = two;                              //縦 Pixel 数

        if(i == 23)
            h2 = two;

        if(i == 24)
            h3 = two;

        if(i == 25)
```

```
        {
          h4 = two;
          height = (pow(255, 3) * h4) + (pow(255, 2) * h3) +
                   (255 * h2) + h1;
        }

      if(i == 28)
        {
          if(two == 24)
            set = 3;
          else
            {
              cout << endl;
              exit(0);
            }
        }

      if(i == 46)
        {
          if(two)
            set = two;

          j = width * height * set + 54;
        }
    }

//画像データ本体の処理////////////////////////////////////

  for(int k = 0; k < 16; k++)
    {
      B[k] = 0;
      G[k] = 0;
      R[k] = 0;
      H[k] = 0;
      S[k] = 0;
      V[k] = 0;
```

```
    }

    for(i = 54; i < j; i++)
    {
        two = fgetc(fp2);

        amari1 = (i - 54) % 3;
        amari2 = two / 16;

        if(amari1 == 0)
        {
            B[amari2]++;
            BB = two;
        }

        if(amari1 == 1)
        {
            G[amari2]++;
            GG = two;
        }

        if(amari1 == 2)
        {
            R[amari2]++;
            RR = two;

            BGR_to_HSV(BB, GG, RR, &h, &s, &v);
            H[h]++;
            S[s]++;
            V[v]++;
        }
    }

//内積計算//////////////////////////////////////////////

    length2 = 0;
```

```
for(int b = 0; b < 16; b++)
 {
   length2 = length2 + pow(B[b], 2);
   inner[b] = inner[b].* B[b];
 }

for(int g = 0; g < 16; g++)
 {
   length2 = length2 + pow(G[g], 2);
   inner[16+g] = inner[16+g] * G[g];
 }

for(int r = 0; r < 16; r++)
 {
   length2 = length2 + pow(R[r], 2);
   inner[32+r] = inner[32+r] * R[r];
 }
for(int h = 0; h < 16; h++)
 {
   length2 = length2 + pow(H[h], 2);
   inner[48+h] = inner[48+h] * H[h];
 }

for(int s = 0; s < 16; s++)
 {
   length2 = length2 + pow(S[s], 2);
   inner[64+s] = inner[64+s] * S[s];
 }

for(int v = 0; v < 16; v++)
 {
    length2 = length2 + pow(V[v], 2);
   inner[80+v] = inner[80+v] * V[v];
 }
```

```
    length2 = sqrt(length2);

    fclose(fp2);

    for(int n = 0; n < 96; n++)
     {
        Inner = Inner + inner[n];
     }

    cout << "cosθ= " << Inner / (length1 * length2) << endl;
                                                    //出力
}
```

● ２枚の画像を読み込んでＢＧＲ要素でコサイン尺度を計算

```
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>

////////////////////////////////////////////////////////////////
//main 関数
////////////////////////////////////////////////////////////////

void main(int argc, char *argv[])
 {
    FILE *fp1, *fp2;
    char add1[100] = "c:¥¥bmp¥¥bmp¥¥";
    char add2[100] = "c:¥¥bmp¥¥bmp¥¥";
    unsigned int one, two;
    int i;
    long j;
    int h1, h2, h3, h4, w1, w2, w3, w4;
    int height, width, set;
    int B[16], G[16], R[16];
    int amari1, amari2;
    double length1, length2, Inner = 0.0;
    float inner[96];

    fp1 = fopen(strcat(add1, argv[1]), "r");
    fp2 = fopen(strcat(add2, argv[2]), "r");

////////////////////////////////////////////////////
//１個目ファイルデータの処理
////////////////////////////////////////////////////

    for(i = 0; i < 54; i++)
     {
        one = (unsigned int)fgetc(fp1);
```

```
if(i == 18)
    w1 = one;                                   //横 Pixel 数

if(i == 19)
    w2 = one;

if(i == 20)
    w3 = one;

if(i == 21)
 {
    w4 = one;
    width = (pow(255, 3) * w4) + (pow(255, 2) * w3) +
                (255 * w2) + w1;
 }

if(i == 22)
    h1 = one;                                   //縦 Pixel 数

if(i == 23)
    h2 = one;

if(i == 24)
    h3 = one;

if(i == 25)
 {
    h4 = one;
    height = (pow(255, 3) * h4) + (pow(255, 2) * h3) +
                (255 * h2) + h1;
 }

if(i == 28)
 {
    if(one == 24)
```

```
            set = 3;
        else
         {
            cout << endl;
            exit(0);
         }
    }

    if(i == 46)
     {
        if(one)
            set = one;

        j = height * width * set + 54;
     }
  }

//画像データ本体の処理////////////////////////////////////

  for(int k = 0; k < 16; k++)
   {
     B[k] = 0;
     G[k] = 0;
     R[k] = 0;
   }

  for(i = 54; i < j; i++)
   {
     one = fgetc(fp1);

     amari1 = (i - 54) % 3;
     amari2 = one / 16;

     if(amari1 == 0)
      {
        B[amari2]++;
```

```
        }

        if(amari1 == 1)
        {
          G[amari2]++;
        }

        if(amari1 == 2)
        {
          R[amari2]++;
        }
      }

//内積計算//////////////////////////////////////////////

      length1 = 0;

      for(int b = 0; b < 16; b++)
      {
        length1 = length1 + pow(B[b], 2);
        inner[b] = B[b];
      }

      for(int g = 0; g < 16; g++)
      {
        length1 = length1 + pow(G[g], 2);
        inner[16+g] = G[g];
      }

      for(int r = 0; r < 16; r++)
      {
        length1 = length1 + pow(R[r], 2);
        inner[32+r] = R[r];
      }

      length1 = sqrt(length1);
```

```
    fclose(fp1);

////////////////////////////////////////////////////
//２個目ファイルデータの処理
////////////////////////////////////////////////////

    for(i = 0; i < 54; i++)
     {
        two = (unsigned int)fgetc(fp2);

        if(i == 18)
            w1 = two;                          //横 Pixel 数

        if(i == 19)
            w2 = two;

        if(i == 20)
            w3 = two;

        if(i == 21)
         {
            w4 = two;
            width = (pow(255, 3) * w4) + (pow(255, 2) * w3) +
                        (255 * w2) + w1;
         }

        if(i == 22)
            h1 = two;                          //縦 Pixel 数

        if(i == 23)
            h2 = two;

        if(i == 24)
            h3 = two;
```

```
    if(i == 25)
     {
        h4 = two;
        height = (pow(255, 3) * h4) + (pow(255, 2) * h3) +
                 (255 * h2) + h1;
     }

    if(i == 28)
     {
        if(two == 24)
          set = 3;
        else
         {
           cout << endl;
           exit(0);
         }
    }

    if(i == 46)
     {
        if(two)
          set = two;

        j = height * width * set + 54;
     }
  }

//画像データ本体の処理////////////////////////////////////

  for(int k = 0; k < 16; k++)
   {
     B[k] = 0;
     G[k] = 0;
     R[k] = 0;
   }
```

```
for(i = 54; i < j; i++)
 {
   two = fgetc(fp2);

   amari1 = (i - 54) % 3;
   amari2 = two / 16;

   if(amari1 == 0)
    {
      B[amari2]++;
    }

   if(amari1 == 1)
    {
      G[amari2]++;
    }

   if(amari1 == 2)
    {
      R[amari2]++;
    }
 }

//内積計算////////////////////////////////////////////////

   length2 = 0;

   for(int b = 0; b < 16; b++)
    {
      length2 = length2 + pow(B[b], 2);
      inner[b] = inner[b] * B[b];
    }

   for(int g = 0; g < 16; g++)
    {
      length2 = length2 + pow(G[g], 2);
```

```
        inner[16+g] = inner[16+g] * G[g];
    }

    for(int r = 0; r < 16; r++)
    {
        length2 = length2 + pow(R[r], 2);
        inner[32+r] = inner[32+r] * R[r];
    }

    length2 = sqrt(length2);

    fclose(fp2);

    for(int n = 0; n < 48; n++)
    {
        Inner = Inner + inner[n];
    }

    cout << "cosθ= " << Inner / (length1 * length2) << endl;
}
```

● ２枚の画像を読み込んでＨＳＶ要素でコサイン尺度を計算

```
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>

////////////////////////////////////////////////////////////////
//HSV 変換関数
////////////////////////////////////////////////////////////////

void BGR_to_HSV(int B, int G, int R, int *h, int *s, int *v)
 {
   float H, S, V, X;
   float r = 0, g = 0, b = 0;
   int pi = 180;

   if(B > G)
    {
      if(B > R)
        V = B;
      else
        V = R;
    }
   else
    {
      if(G > R)
        V = G;
      else
        V = R;
    }

   if(B < G)
    {
      if(B < R)
        X = B;
```

```
    else
      X = R;
  }
else
  {
    if(G < R)
      X = G;
    else
      X = R;
  }

if(V != X)
  {
    r = (V - R) / (V - X);
    g = (V - G) / (V - X);
    b = (V - B) / (V - X);
  }

if(R == V)
  H = (pi / 3) * (b - g);
else if(G == V)
  H = (pi / 3) * (2 + r - b);
else if(B == V)
  H = (pi / 3) * (4 + g - r);

if(H < 0)
  H = H + 360;

V = (V / 256) * 100;
X = (X / 256) * 100;

if(V)
  S = ((V - X) / V) * 100;
else
  S = 0;
```

```
    *h = H / 22.56;
    *s = S / 6.3;
    *v = V / 6.3;
}

////////////////////////////////////////////////////////////////
//main 関数
////////////////////////////////////////////////////////////////

void main(int argc, char *argv[])
{
    FILE *fp1, *fp2;
    char add1[100] = "c:¥¥bmp¥¥bmp¥¥";
    char add2[100] = "c:¥¥bmp¥¥bmp¥¥";
    unsigned int one, two;
    int i;
    long j;
    int h1, h2, h3, h4, w1, w2, w3, w4;
    int height, width, set;
    int H[16], S[16], V[16];
    int BB, GG, RR, h, s, v;
    int amari1;
    double length1, length2, Inner = 0.0;
    double inner[96];

    fp1 = fopen(strcat(add1, argv[1]), "r");
    fp2 = fopen(strcat(add2, argv[2]), "r");

////////////////////////////////////////////////////
//１個目ファイルデータの処理
////////////////////////////////////////////////////

    for(i = 0; i < 54; i++)
    {
        one = (unsigned int)fgetc(fp1);
```

```
if(i == 18)
    w1 = one;                                   //横 Pixel 数

if(i == 19)
    w2 = one;

if(i == 20)
    w3 = one;

if(i == 21)
 {
    w4 = one;
    width = (pow(255, 3) * w4) + (pow(255, 2) * w3) +
                (255 * w2) + w1;
 }

if(i == 22)
    h1 = one;                                   //縦 Pixel 数

if(i == 23)
    h2 = one;

if(i == 24)
    h3 = one;

if(i == 25)
 {
    h4 = one;
    height = (pow(255, 3) * h4) + (pow(255, 2) * h3) +
              (255 * h2) + h1;
 }

if(i == 28)
 {
    if(one == 24)
      set = 3;
```

```
        else
         {
            cout << endl;
            exit(0);
         }
    }

    if(i == 46)
     {
        if(one)
          set = one;

        j = height * width * set + 54;
     }
  }

//画像データ本体の処理////////////////////////////////////

  for(int k = 0; k < 16; k++)
   {
     H[k] = 0;
     S[k] = 0;
     V[k] = 0;
   }

  for(i = 54; i < j; i++)
   {
     one = fgetc(fp1);

     amari1 = (i - 54) % 3;

     if(amari1 == 0)
      {
        BB = one;
      }
```

```
        if(amari1 == 1)
         {
            GG = one;
         }

        if(amari1 == 2)
         {
            RR = one;

            BGR_to_HSV(BB, GG, RR, &h, &s, &v);
            H[h]++;
            S[s]++;
            V[v]++;
         }
      }

//内積計算//////////////////////////////////////////////

      length1 = 0;

      for(int h = 0; h < 16; h++)
       {
         length1 = length1 + pow(H[h], 2);
         inner[h] = H[h];
       }

      for(int s = 0; s < 16; s++)
       {
         length1 = length1 + pow(S[s], 2);
         inner[16+s] = S[s];
       }

      for(int v = 0; v < 16; v++)
       {
         length1 = length1 + pow(V[v], 2);
         inner[32+v] = V[v];
```

```
    }

    length1 = sqrt(length1);

    fclose(fp1);

////////////////////////////////////////////////
//2個目ファイルデータの処理
////////////////////////////////////////////////

    for(i = 0; i < 54; i++)
     {
        two = (unsigned int)fgetc(fp2);

        if(i == 18)
            h1 = two;                          //横 Pixel 数

        if(i == 19)
            h2 = two;

        if(i == 20)
            h3 = two;

        if(i == 21)
         {
            h4 = two;
            height = (pow(255, 3) * h4) + (pow(255, 2) * h3) +
                     (255 * h2) + h1;
         }

        if(i == 22)
            w1 = two;                          //縦 Pixel 数

        if(i == 23)
            w2 = two;
```

```
        if(i == 24)
            w3 = two;

        if(i == 25)
         {
            w4 = two;
            width = (pow(255, 3) * w4) + (pow(255, 2) * w3) +
                    (255 * w2) + w1;
         }

        if(i == 28)
         {
            if(two == 24)
              set = 3;
            else
             {
               cout << endl;
               exit(0);
             }
        }

        if(i == 46)
         {
            if(two)
              set = two;

            j = height * width * set + 54;
         }
    }

//画像データ本体の処理////////////////////////////////////

    for(int k = 0; k < 16; k++)
     {
        H[k] = 0;
        S[k] = 0;
```

```
        V[k] = 0;
    }

    for(i = 54; i < j; i++)
    {
        two = fgetc(fp2);

        amari1 = (i - 54) % 3;

        if(amari1 == 0)
        {
            BB = two;
        }

        if(amari1 == 1)
        {
            GG = two;
        }

        if(amari1 == 2)
        {
            RR = two;

            BGR_to_HSV(BB, GG, RR, &h, &s, &v);
            H[h]++;
            S[s]++;
            V[v]++;
        }
    }

//内積計算////////////////////////////////////////////////

    length2 = 0;

    for(int h = 0; h < 16; h++)
    {
```

```
        length2 = length2 + pow(H[h], 2);
        inner[h] = inner[h] * H[h];
      }

    for(int s = 0; s < 16; s++)
      {
        length2 = length2 + pow(S[s], 2);
        inner[16+s] = inner[16+s] * S[s];
      }

    for(int v = 0; v < 16; v++)
      {
         length2 = length2 + pow(V[v], 2);
        inner[32+v] = inner[32+v] * V[v];
      }

    length2 = sqrt(length2);

    fclose(fp2);

    for(int n = 0; n < 48; n++)
      {
        Inner = Inner + inner[n];
      }

    cout << "cosθ= " << Inner / (length1 * length2) << endl;
}
```

● ２枚の画像を読み込んで中央部分だけを対象にＢＧＲＨＳＶ要素でコサイン尺度を計算

```
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>

////////////////////////////////////////////////////////////////
//HSV 変換関数
////////////////////////////////////////////////////////////////

void BGR_to_HSV(int B, int G, int R, int *h, int *s, int *v)
 {
   float H, S, V, X;
   float r = 0, g = 0, b = 0;
   int pi = 180;

   if(B > G)
    {
      if(B > R)
        V = B;
      else
        V = R;
    }
   else
    {
      if(G > R)
        V = G;
      else
        V = R;
    }

   if(B < G)
    {
      if(B < R)
```

```
      X = B;
    else
      X = R;
  }
else
  {
    if(G < R)
      X = G;
    else
      X = R;
  }

if(V != X)
  {
    r = (V - R) / (V - X);
    g = (V - G) / (V - X);
    b = (V - B) / (V - X);
  }

if(R == V)
  H = (pi / 3) * (b - g);
else if(G == V)
  H = (pi / 3) * (2 + r - b);
else if(B == V)
  H = (pi / 3) * (4 + g - r);

if(H < 0)
  H = H + 360;

V = (V / 256) * 100;
X = (X / 256) * 100;

if(V)
  S = ((V - X) / V) * 100;
else
  S = 0;
```

```
    *h = H / 22.56;
    *s = S / 6.3;
    *v = V / 6.3;
}

////////////////////////////////////////////////////////////////
//main 関数
////////////////////////////////////////////////////////////////

void main(int argc, char *argv[])
 {
    FILE *fp1, *fp2;
    char add1[100] = "c:¥¥bmp¥¥bmp¥¥";
    char add2[100] = "c:¥¥bmp¥¥bmp¥¥";
    unsigned int one, two;
    int i;
    long j;
    int h1, h2, h3, h4, w1, w2, w3, w4;
    int height, width, set;
    int h_three, w_three;
    int B[16], G[16], R[16], H[16], S[16], V[16];
    int BB, GG, RR, h, s, v;
    int amari1, amari2;
    double length1, length2, Inner = 0.0;
    double inner[96];

    fp1 = fopen(strcat(add1, argv[1]), "r");
    fp2 = fopen(strcat(add2, argv[2]), "r");

////////////////////////////////////////////////////
//１個目ファイルデータの処理
////////////////////////////////////////////////////

    for(i = 0; i < 54; i++)
     {
```

```
one = (unsigned int)fgetc(fp1);

if(i == 18)
    w1 = one;                                   //横 Pixel 数

if(i == 19)
    w2 = one;

if(i == 20)
    w3 = one;

if(i == 21)
 {
    w4 = one;
    width = (pow(255, 3) * w4) + (pow(255, 2) * w3) +
                (255 * w2) + w1;
    w_three = (width / 3) * 3;
 }

if(i == 22)
    h1 = one;                                   //縦 Pixel 数

if(i == 23)
    h2 = one;

if(i == 24)
    h3 = one;

if(i == 25)
 {
    h4 = one;
    height = (pow(255, 3) * h4) + (pow(255, 2) * h3) +
                (255 * h2) + h1;
    h_three = height / 3;
 }
```

```
    if(i == 28)
     {
        if(one == 24)
          set = 3;
        else
         {
           cout << endl;
           exit(0);
         }
     }

    if(i == 46)
     {
        if(one)
          set = one;

        j = height * width * set + 54;
     }
  }

//画像データ本体の処理////////////////////////////////

  for(int k = 0; k < 16; k++)
   {
     B[k] = 0;
     G[k] = 0;
     R[k] = 0;
     H[k] = 0;
     S[k] = 0;
     V[k] = 0;
   }

  for(i = 54; i < j; i++)
   {
     one = fgetc(fp1);
```

```
        if(((i / (width * 3)) > h_three) && ((i / (width * 3)) <= (h_three * 2)))
         {
           if(((i % (width * 3)) >= w_three) && ((i % (width * 3)) < (w_three *
2)))
            {
              amari1 = (i - 54) % 3;
              amari2 = one / 16;

              if(amari1 == 0)
               {
                 B[amari2]++;
                 BB = one;
               }

              if(amari1 == 1)
               {
                 G[amari2]++;
                 GG = one;
               }

              if(amari1 == 2)
               {
                 R[amari2]++;
                 RR = one;

                 BGR_to_HSV(BB, GG, RR, &h, &s, &v);
                 H[h]++;
                 S[s]++;
                 V[v]++;
               }
            }
         }
    }

//内積計算//////////////////////////////////////////////
```

```
length1 = 0;

for(int b = 0; b < 16; b++)
 {
   length1 = length1 + pow(B[b], 2);
   inner[b] = B[b];
 }

for(int g = 0; g < 16; g++)
 {
   length1 = length1 + pow(G[g], 2);
   inner[16+g] = G[g];
 }

for(int r = 0; r < 16; r++)
 {
   length1 = length1 + pow(R[r], 2);
   inner[32+r] = R[r];
 }
for(int h = 0; h < 16; h++)
 {
   length1 = length1 + pow(H[h], 2);
   inner[48+h] = H[h];
 }

for(int s = 0; s < 16; s++)
 {
   length1 = length1 + pow(S[s], 2);
   inner[64+s] = S[s];
 }

for(int v = 0; v < 16; v++)
 {
   length1 = length1 + pow(V[v], 2);
   inner[80+v] = V[v];
 }
```

```
    length1 = sqrt(length1);

    fclose(fp1);

////////////////////////////////////////////////////////////////
//2個目ファイルデータの処理
////////////////////////////////////////////////////////////////

    for(i = 0; i < 54; i++)
     {
        two = (unsigned int)fgetc(fp2);

        if(i == 18)
            w1 = two;                          //横 Pixel 数

        if(i == 19)
            w2 = two;

        if(i == 20)
            w3 = two;

        if(i == 21)
         {
            w4 = two;
            width = (pow(255, 3) * w4) + (pow(255, 2) * w3) +
                        (255 * w2) + w1;
            w_three = (width / 3) * 3;
         }

        if(i == 22)
            h1 = two;                          //縦 Pixel 数

        if(i == 23)
            h2 = two;
```

```
        if(i == 24)
            h3 = two;

        if(i == 25)
         {
            h4 = two;
            height = (pow(255, 3) * h4) + (pow(255, 2) * h3) +
                     (255 * h2) + h1;
            h_three = height / 3;
         }

        if(i == 28)
         {
            if(two == 24)
              set = 3;
            else
             {
               cout << endl;
               exit(0);
             }
        }

        if(i == 46)
         {
            if(two)
              set = two;

            j = height * width * set + 54;
         }
    }

//画像データ本体の処理////////////////////////////////////

    for(int k = 0; k < 16; k++)
     {
       B[k] = 0;
```

```
            G[k] = 0;
            R[k] = 0;
            H[k] = 0;
            S[k] = 0;
            V[k] = 0;
        }

    for(i = 54; i < j; i++)
     {
        two = fgetc(fp2);

        if(((i / (width * 3)) > h_three) && ((i / (width * 3)) <= (h_three * 2)))
         {
           if(((i % (width * 3)) >= w_three) && ((i % (width * 3)) < (w_three *
2)))
            {
              amari1 = (i - 54) % 3;
              amari2 = two / 16;

              if(amari1 == 0)
               {
                 B[amari2]++;
                 BB = two;
               }

              if(amari1 == 1)
               {
                 G[amari2]++;
                 GG = two;
               }

              if(amari1 == 2)
               {
                 R[amari2]++;
                 RR = two;
```

```
                BGR_to_HSV(BB, GG, RR, &h, &s, &v);
                H[h]++;
                S[s]++;
                V[v]++;
            }
        }
    }
}

//内積計算////////////////////////////////////////////

length2 = 0;

for(int b = 0; b < 16; b++)
 {
    length2 = length2 + pow(B[b], 2);
    inner[b] = inner[b] * B[b];
 }

for(int g = 0; g < 16; g++)
 {
    length2 = length2 + pow(G[g], 2);
    inner[16+g] = inner[16+g] * G[g];
 }

for(int r = 0; r < 16; r++)
 {
    length2 = length2 + pow(R[r], 2);
    inner[32+r] = inner[32+r] * R[r];
 }
for(int h = 0; h < 16; h++)
 {
    length2 = length2 + pow(H[h], 2);
    inner[48+h] = inner[48+h] * H[h];
 }
```

```
    for(int s = 0; s < 16; s++)
     {
        length2 = length2 + pow(S[s], 2);
        inner[64+s] = inner[64+s] * S[s];
     }

    for(int v = 0; v < 16; v++)
     {
         length2 = length2 + pow(V[v], 2);
        inner[80+v] = inner[80+v] * V[v];
     }

    length2 = sqrt(length2);

    fclose(fp2);

    for(int n = 0; n < 96; n++)
     {
        Inner = Inner + inner[n];
     }

    cout << "cosθ= " << Inner / (length1 * length2) << endl;
}
```

● ディレクトリ内のｂｍｐファイル全ての組み合わせの２枚をｅｘｅファイルに渡して起動

```
export all="$@"

declare -i i=1;
declare -i j=1;

for f in $all; do
    let j="1"
    for g in $all; do
        if [ $i -lt $j ] ; then
            echo -n "$f $g "
            /bmp/bmp22.exe $f $g
        fi
        let j="$j+1"
    done
    let i="$i+1"
done
```

● ディレクトリ内のｂｍｐファイル全てを１枚ずつｅｘｅファイルに渡して起動

```
for f in *.bmp
do
    echo -n "$1 $f "
    /bmp/bmp22.exe $1 $f
done
```