画像処理プログラムの説明

説明の内容

- 画像処理に際して、画像ファイルからデータを抽出し利用する手順を以下の項目に分けて説明する
  - PPM形式
  - 画像変換コマンド群netpbm
  - データ抽出(プログラムと説明)
  - 画像処理の例とその結果

PPM形式

- Portable PixMap
- RGBカラー画像を扱う為のフォーマット
- 「ヘッダ」+「画像データ」で構成
- ヘッダ
  - PPM形式を示すマジックナンバー
  - 画像の幅、高さ
  - R,G,B値の最大値
- 画像データ
  - バイナリ又はテキスト形式
  - 画像の上から下へ一行(ピクセル)ずつ羅列

PPM形式の具体例

テキスト形式(P3)

ヘッダ

画像データ

```
P3
128 128
255
101 93 38 92 95 49 102 94 56 92 95 49
90 89 48 90 89 48 102 94 56 104 100 57
104 100 57 104 100 57 102 94 56 104 100 57
92 95 49 102 94 56 90 89 48 92 95 49
90 89 48 92 95 49 104 100 57 92 95 49
90 89 48 90 89 48 84 82 48 90 89 48
108 107 63 104 100 57 92 95 49 108 107 63
⋮
```

バイナリ形式(P6)

ヘッダ

画像データ

```
P6
128 128
255
?lk?lk?lk?tp@lk?lk?lk?lk?lk?lk?hd
tp*tp@tp@tp*tp@tw-tp@tw-tp@~x?tv=
[■¥Z■¥[■SX■TW■SX■TW■¥d■ee■hk■kl&k
■kl&hk■hk■ee■hk■hk■hk■lr%tw-hk■hk
|y/tw-|y/tw-tp*jr■tp*tw-tw-tw-to■
l&kl&lk?tp*d^■d^■he%e]&he%d^■he%e
-~x?t■(tw-■■0tw-tw#lr%hk■kl&tw-lr
Φ=t■(t■(■■0ｆ1■■0■■0t■(■■0■■0■■0
l&¥j■kl&¥j■lr%¥j■¥j■lw!lw!jr■lw!t
(t■(・4ル3t■(ｆ1迫;ル3ｆ1t■(t■(t■
遥0-弄■^■¥;■tZ'継*桿9ッ吋桿9梵A梵A
⋮
```

ヘッダ

マジックナンバー
幅　高さ
R,G,B値の最大値

画像データ

Rの値　Gの値　Bの値　Rの値・・・

・テキストの場合は10進数のデータ
・バイナリの場合は8bitのデータ

netpbm

- 各種画像形式の変換を行うコマンド群
- giftoppm
  - GIFファイルをPPMファイル(P6)に変換する
- djpeg
  - JPEGファイルをPPMファイル(P6)に変換する
- pnmscale –xsize -ysize
  - PNMファイルの大きさを変える
- パイプの使用が可能
  - 例

giftoppm in.gif |pnmscale –xsize 128 –ysize 128 >out.ppm

画像ファイルの操作

- 処理１
  - 画像リストから画像ファイルを読み込む
- 処理２
  - 画像ファイルをPPMファイルに変換する
- 処理３
  - PPMファイルのヘッダから画像の幅、高さを読み込む
  - PPMファイルの画像データR,G,B値をそれぞれ配列に入れる

処理1

- プログラムの実行は次のようにする
  - % ./fv imagelist data_no

imagelist(例)

```
/jasmine/jasmine121/jasmine/imagedb/images/usedcar/1/B0000001.gif
/jasmine/jasmine121/jasmine/imagedb/images/usedcar/1/B0000002.gif
/jasmine/jasmine121/jasmine/imagedb/images/usedcar/1/B0000003.gif
/jasmine/jasmine121/jasmine/imagedb/images/usedcar/1/B0000004.gif
/jasmine/jasmine121/jasmine/imagedb/images/usedcar/1/B0000005.gif
/jasmine/jasmine121/jasmine/imagedb/images/usedcar/1/B0000006.gif
/jasmine/jasmine121/jasmine/imagedb/images/usedcar/1/B0000007.gif
/jasmine/jasmine121/jasmine/imagedb/images/usedcar/1/B0000008.gif
```

data_no・・・imagelistに含まれる画像の枚数

imagelistを読み込んで、data_noの数だけ処理を行う

プログラム

```
 1 main(int argc, char **argv)
 2 {
 3         FV fv;
 4         FILE *fp;
 5         int no;
 6         char *ImageName;
 7         char **NameVectors;
 8         int DATA_NO = atoi (argv[2]);
 9
10         ImageName = new char [256];
11         NameVectors = new char *[DATA_NO];
12         fp = fopen (argv[1], "r");
13         no = 0;
14         while (fscanf (fp, "%s", ImageName) !=EOF)
15         {
16                 NameVectors[no] = new char[strlen (ImageName) + 1];
17                 strcpy (NameVectors[no], ImageName);
18                 no++;
19         }
20         fclose (fp);
21
22         for (no = 0; no < DATA_NO; no++)
23         {
24                 fv.GetFeature(NameVectors[no]);
25         }
26
27         for (int j = 0; j < DATA_NO; j++)
28         {
29                 delete[]NameVectors[j];
30         }
31 }
```

クラス

- 新しく定義された型で、構成する要素として関数を含む

```
 1 class FV{
 2   private:
 3     FILE *ppm_file;
 4     struct PPMHead{
 5         int Width;
 6         int Height;
 7     }ppmhead;
 8     int *colR,*colG,*colB;
 9     double *colY,*colCb,*colCr;
10
11     void ppmopen(const char *);
12     void ppmclose(void);
13     unsigned int PPMGetVal(void);
14     void ppmheader(void);
15     void readPPM(const char* file_name);
16
17   public:
18     void GetFeature(char *);
19 };
```

クラス定義の例:FV

private:ここで指定されたメンバは
メンバ関数の中だけから参照される

```
void FV::
ppmopen(const char *file_name)
{          …
}
```

public:ここで指定されたメンバは
どの関数の中からも参照される

```
FV fv;
fv.GetFeature(*ImageName);
```

プログラム説明

- (1行目)int argc, char **argv
  - コマンドラインのパラメータをプログラムに渡す
  - argcはパラメータの数、argvはパラメータの文字列（この場合、argcは3、argv[0]は’fv’、argv[1]は’imagelist’、argv[2]は’data_no’となる）
- (4行目)FILE *
  - ファイルを参照するオブジェクトを宣言する型
- (10,11行目)new
  - 新たに割り当てられたオブジェクトを参照する、指定された型のポインタを返す演算子
  - 配列の大きさを指定するとそれに応じた記憶領域が割り当てられる

プログラム説明(続き)

- (14行目～19行目)
  - imagelistから一行画像ファイル名を読み込む
  - 画像ファイル名の長さを計算し、その長さの文字列を格納する配列(NameVectors[])の記憶領域をnew演算子を用いて割り当てる
  - 配列に画像ファイル名を格納する
  - 次の行の画像ファイル名について以上の処理を繰り返す
- (22行目～25行目)
  - 配列NameVectorsに格納されている画像ファイルについてGetFeature()関数による処理を行う
- (27行目～30行目)delete
  - new演算子によって割り当てられたオブジェクトを削除する

処理2

- GIF画像をPPM形式に変換する
- PPMファイルはサイズが大きいので一時的なファイルとして扱う(tmpnam)
- readPPM()はヘッダと画像データの抽出を行う関数である

プログラム

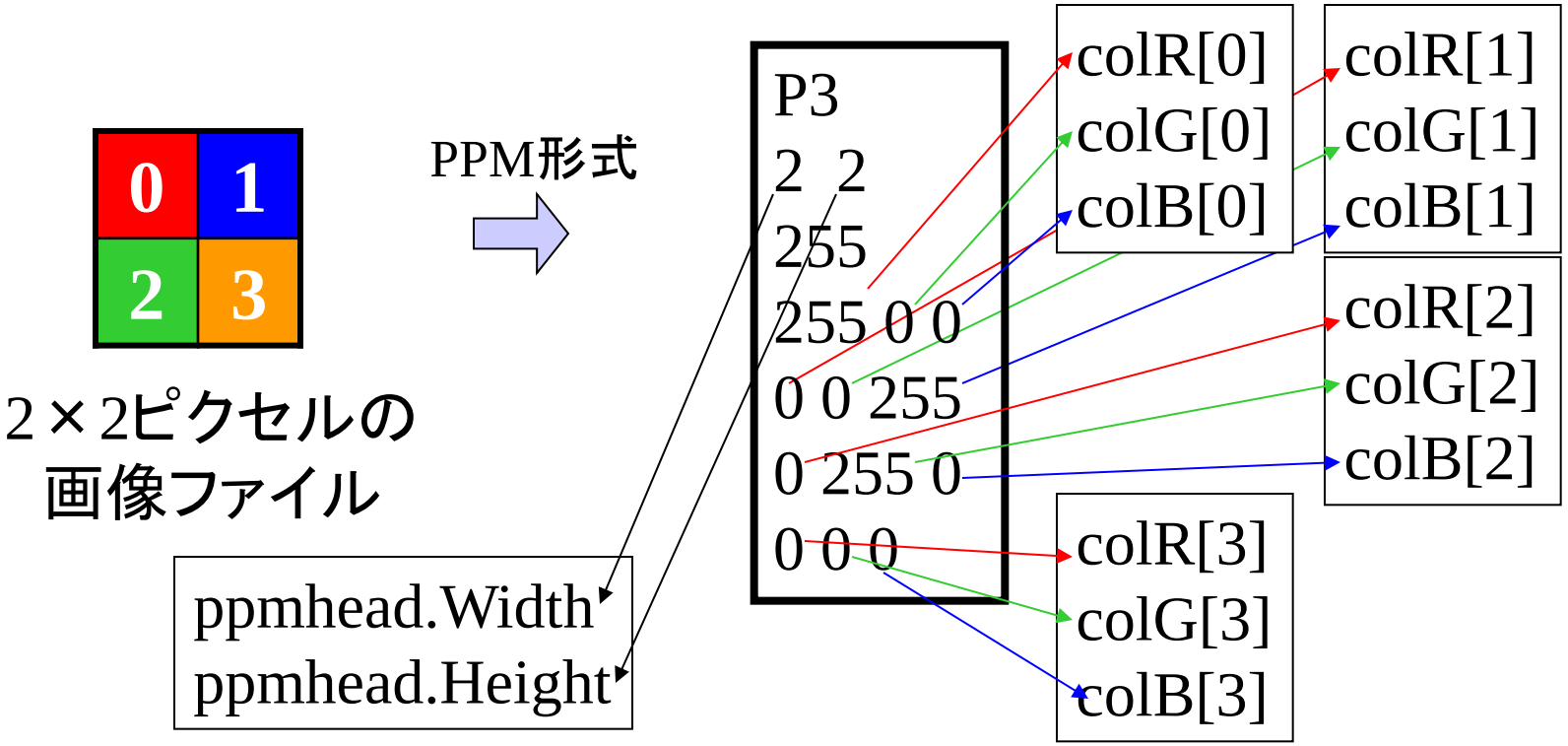
```
1  void FV::
2  GetFeature (char *GIFImageName)
3  {
4          char com[256];
5          char PPMImageName[L_tmpnam];
6
7          sprintf (com, "giftoppm %s > %s", GIFImageName, tmpnam (PPMImageName));
8          system (com);
9
10         readPPM (PPMImageName);
11         unlink (PPMImageName);
```

プログラム説明

- (7行目)int sprintf(char *s, const char *arg1 …);
  - 引数sで指定される文字列に出力を書き込む
  - この場合comには”giftoppm A > B”という文字列が格納される。ただし、Aはimagelist中の画像ファイル名、Bは一意的なファイル名を表す文字列である
- (7行目)char *tmpnam(char *fname);
  - fnameで参照される配列に一意的なファイル名を格納する
  - 配列の大きさはL_tmpnam
- (8行目)int system(char *cmd);
  - 文字列cmdで指定されるコマンドをOSで実行する
  - この場合、giftoppmコマンドを実行し画像の変換が行われる
- (11行目)int unlink(const char *pathname);
  - 名前の削除、又はそれが参照しているファイルを削除する

処理3

- ヘッダから画像の大きさ、画像データからR,G,B値を読み込む



プログラム(1)

```
 1 void FV::
 2 ppmopen (const char *file_name)
 3 {
 4         if ((ppm_file = fopen (file_name, "r")) == NULL)
 5         {
 6                 fprintf (stderr, "cannot open %s¥n", file_name);
 7                 exit (1);
 8         }
 9 }
10
11 void FV::
12 ppmclose (void)
13 {
14         fclose (ppm_file);
15 }
16
17 unsigned int FV::
18 PPMGetVal (void)
19 {
20         unsigned int tmp;
21         unsigned char ch;
22
23         do
24         {
25                 ch = fgetc (ppm_file);
26         }
27         while (ch <= ' ');
28         if (fscanf (ppm_file, "%u", &tmp) != 1)
29         {
30                 fprintf (stderr, "%s¥n", "Error parsing file!");
31                 exit (1);
32         }
33         return (tmp);
34 }
```

プログラム(2)

```
35
36 void FV::
37 ppmheader (void)
38 {
39         char is_ppm, is_raw_format;
40
41         if ((is_ppm = fgetc (ppm_file)) != 'P')
42         {
43                 fprintf (stderr, "¥n This is not a ppm file. ¥n");
44                 exit (1);
45         }
46         if ((is_raw_format = fgetc (ppm_file)) != '6')
47         {
48                 fprintf (stderr, "¥n This is not a binary-format ppm file. ¥n");
49                 exit (1);
50         }
51
52         while ((is_ppm = fgetc (ppm_file)) != '¥n')
53         {
54         }
55
56         ppmhead.Width = PPMGetVal ();
57         ppmhead.Height = PPMGetVal ();
58 }
```

プログラム(3)

```
59
60 void FV::
61 readPPM (const char *file_name)
62 {
63         ppmopen (file_name);
64         ppmheader ();
65
66         int w = ppmhead.Width;
67         int h = ppmhead.Height;
68
69         colR = new int[w * h];
70         colG = new int[w * h];
71         colB = new int[w * h];
72
73         unsigned char *buffer;
74         buffer = new unsigned char[h * w * 3];
75         long _offset = -1 * h * w * 3;
76         fseek (ppm_file, _offset, SEEK_END);
77         fread (buffer, sizeof (unsigned char), h * w * 3, ppm_file);
78
79         int mn = 0;
80         for (int i = 0; i < h; i++)
81         {
82                 for (int j = 0; j < w; j++)
83                 {
84                         colR[i * w + j] = buffer[mn++];
85                         colG[i * w + j] = buffer[mn++];
86                         colB[i * w + j] = buffer[mn++];
87                 }
88         }
89         ppmclose ();
90         delete[]buffer;
91 }
```

プログラム(1)説明

- (1行目～9行目)
  - PPM形式に変換された画像ファイルを開く
- (11行目～15行目)
  - PPM形式に変換された画像ファイルを閉じる
- (17,20行目)unsigned int
  - 2進整数値を表現する符号無しの整数型
- (21行目)unsigned char
  - 符号無しの文字型
- (23行目～27行目)
  - ヘッダ部分のスペースなどの不必要な情報があればスキップする
- (28行目～32行目)
  - ヘッダから情報（画像の幅、高さ）を符号無し整数として読み取る

プログラム(2)説明

- (41行目～50行目)
  - PPM形式がP6であることを確かめる
- (52行目～54行目)
  - ファイル位置指定子を1行進める(次の行に、画像の幅、高さが記述してある)
- (56行目)
  - 画像の幅を読み取る
- (57行目)
  - 画像の高さを読み取る

プログラム(3)説明

- (69行目～71行目)
  - R,G,B値を格納する配列をそれぞれnew演算子を用いて割り当てる
- (73行目～74行目)
  - 画像のピクセルの数×3の大きさの配列をnew演算子を用いて割り当てる（1つのピクセルはR,G,Bの3種類の値から構成される）
- (75,76行目)int fseek(FILE *fp, long offset, SEEK_END);
  - 画像ファイルの最後からoffsetバイトの位置にファイル位置指定子を移動させる
  - offsetがプラスのときは前方に、マイナスの時は後方に移動する
- (77行目)size_t fread(void *buffer, size_t size, size_t n, FILE *fp);
  - bufferの領域にfpが指すファイルからsize×nバイトのデータを読み込む
  - sizeof(unsigned char)は8ビット
- (79行目～88行目)
  - bufferからR,G,B,R,G,B,...の順で値を取り出す

画像処理

- PPM画像ファイルから得られたR,G,Bの値を用いて画像処理が行われる
- 例:RGB表色系からYCC表色系への変換
  - YCC表色系とは色を輝度[Y]、色差[Cb][Cr]の3つで表す
  - 画像を輝度[Y]のみで表すとモノクロ画像となる
  - Y,Cb,Crの値はR,G,Bの値から以下の式で求めることができる

$$
\begin{aligned}
Y &= 0.299R + 0.587G + 0.114B \\
Cb &= -0.172R - 0.339G + 0.511B \\
Cr &= 0.511R - 0.428G - 0.083B
\end{aligned}
$$

  - ただしR,G,Bが0~255の値のとき、Yは0~255、CbとCrは-128~127の値をとる

プログラム

```
 1 void FV::
 2 rgb2ycc()
 3 {
 4         int w = ppmhead.Width;
 5         int h = ppmhead.Height;
 6
 7         colY  = new double[w * h];
 8         colCb = new double[w * h];
 9         colCr = new double[w * h];
10
11         for (int i = 0; i < w * h; i++)
12         {
13                 colY[i]  =  0.299*colR[i] + 0.587*colG[i] + 0.114*colB[i];
14                 colCb[i] = -0.172*colR[i] - 0.339*colG[i] + 0.511*colB[i] + CENTER;
15                 colCr[i] =  0.511*colR[i] - 0.428*colG[i] - 0.083*colB[i] + CENTER;
16         }
17 }
```

- (7行目～9行目)
  - Y,Cb,Crの値を格納するための配列をnew演算子を用いて割り当てる
  - 配列の大きさはR,G,Bの値を格納している配列と同じになる
- (11行目～16行目)
  - Y,Cb,Crの値をR,G,Bの値から計算してそれぞれ配列に格納する
  - Cb,Crの値の範囲を0~255に揃えるためにCb,Crの値にCENTER(128)を足す

RGB→YCCの結果

- サンプルの画像ファイルをYCC表色系に変換した
- 変換処理の結果はY,Cb,Crの値をPPM形式(P3)でテキストに出力した。よって、この画像データはY,Cb,Cr,Y,Cb,Cr,…となっている
- PPM形式で出力する際に、Cb,Crの値にYを出力しモノクロ画像を作成した

sample

YCC表色系

モノクロ画像