画像処理入門

静止画の処理方法をマスタする

ディジタル・ビデオからキャプチャしたビデオ・ファイルやWeb上の動画クリップなど大部分の動画データは，フレーム(静止画)の集合として構成されます．ですから，動画を扱う場合でも，そのフレームを構成する静止画の知識についても基礎的な事項を理解しておくことは不可欠になります．本章では，静止画の成り立ちと基本的な処理方法について紹介します．

3-1　ディジタル画像

静止画に対する処理方法であるディジタル画像処理は，ディジタル形式の画像情報に対する操作ということになります．ここでいう「ディジタル形式」とは，どのような形式なのでしょうか．ディジタル形式の画像情報を作り出す画像におけるディジタル化は，ほかのディジタル化と同様にサンプリングという操作を行います．たとえば，人が目にする連続階調の画像に対して独立した点(位置)の情報に分解するという作業になります．この一つの点を画素(picture element = pixel：ピクセル)と呼びます．この画素はその分解する精度の種類が二つあり，分割する点(位置)の細かさを表す空間分解能と，分割された各点の明るさの分解能となります．動画の場合は，さらにこれに対して時間の分解能が加わることになります(**図3-1**)．

また，ディジタル化された情報の格納方法として，画素ごとの情報を右から左を順に走査し，そのあと次の行の走査を行って1次元のデータの並びとして格納する「ラスタ形式」と，たとえば線分や円など画像の要素ごとにその要素のコードと位置を格納する「ベクタ形式」があります．本書で取り上げる処理は，ラスタ形式のディジタル画像です．

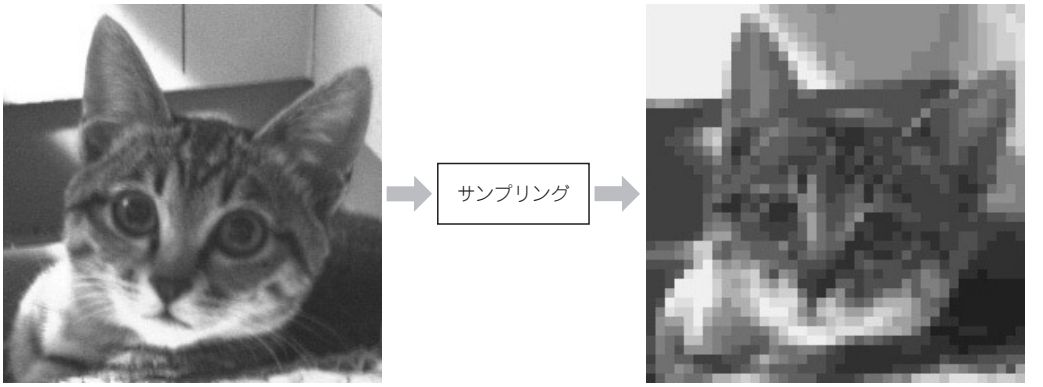
3-2　画像データのファイル形式

スキャナ，ディジタル・カメラ，ビデオ・カメラなどでは，さまざまな方法でサンプリングされた画像データを保存するファイルの形式には多くの種類があります．この節では，Windowsの標準的な形式のBMPとディジタル・カメラなどで標準的に使われるJPEGについて読み出し方法を紹介します．

● BMP

BMP(Bit MaP)は，Windowsで標準的に使われるファイル形式で，同じくWindowsで使われる画像データの形式DIB(非デバイス依存型ビットマップ)にファイルの情報を追加したものです．BMPでは白黒(2値画像)から24ビット・カラー(用語)まで多くのフォーマットの画像データを扱うことができます．また，カラー・パレットの利用も可能です．

ファイルは**図3-2**に示すように，ファイル情報のヘッダ，ビットマップ情報のヘッダ，色パレット，画



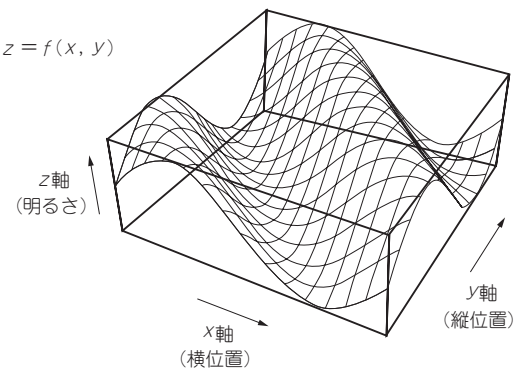


図 3-1　画像のディジタル化
連続階調の画像情報を独立した点(位置)の情報に分解する．ディジタル画像データは縦横位置が決まればその点の明るさが決まる．

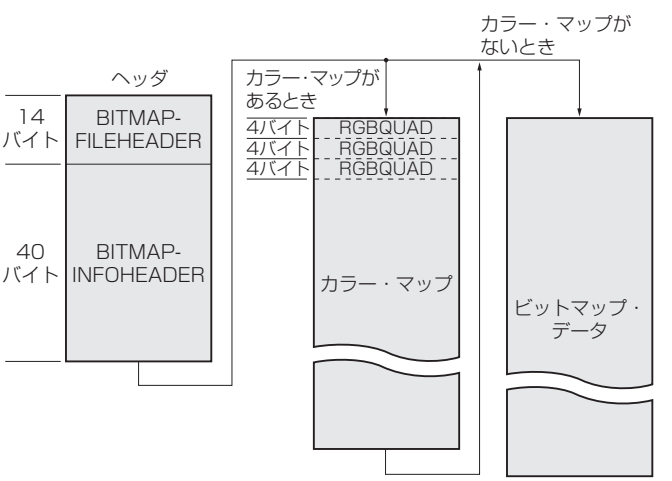


図 3-2　BMP ファイルの構造

素データと続きます．画素データの格納方法は，画素あたりに使用できる色数によって異なります．1 ライン分の画素データは，long 境界(4 バイト)をまたぐことがないようパッディング(用語)されます．

たとえば，24 ビット・カラーの場合は，色パレットが省略され，画素データは画面右下を起点として，

表 3-1　BMP ファイルを構成する構造体

```
typedef struct tagBITMAPFILEHEADER {     /* bmfh */
    UINT    bfType;       // ファイルのタイプを示す．'BM' の値をとる
    DWORD   bfSize;       // ファイルのサイズをバイト単位で示す
    UINT    bfReserved1; // 予約されている．0 に設定する
    UINT    bfReserved2; // 予約されている．0 に設定する
    DWORD   bfOffBits;    // BITMAPFILEHEADER から実際のビットマップ・データまでのオフセットをバイト単
                          // 位で示す
  } BITMAPFILEHEADER;
```

BITMAPFILEHEADER 構造体は，デバイス独立型ビットマップ（DIB）ファイルのタイプ，サイズ，データの配置に関する情報を格納する

（**a**）BITMAPFILEHEADER 構造体

```
typedef struct tagBITMAPINFOHEADER {     /* bmih */
    DWORD   biSize;           //BITMAPINFOHEADER 構造体のバイト数を示す
    LONG    biWidth;          // ビットマップの幅をピクセル単位で示す
    LONG    biHeight;         // ビットマップの高さをピクセル単位で示す
    WORD    biPlanes;         // ターゲット・デバイスのプレーン数を示す
    WORD    biBitCount;       // ピクセル当たりのビット数を示す
    DWORD   biCompression;    // ビットマップの圧縮タイプを示す
    DWORD   biSizeImage;      // イメージのサイズをバイト単位で示す
    LONG    biXPelsPerMeter; // ターゲット・デバイスの水平解像度をメートル当たりのピクセル数で示す
    LONG    biYPelsPerMeter; // ターゲット・デバイスの垂直解像度をメートル当たりのピクセル数で示す
    DWORD   biClrUsed;        // ビットマップが実際に使用するカラー・テーブル内のカラー・インデックスの
                              // 個数を示す
    DWORD   biClrImportant;  // ビットマップの表示において重要と思われるカラー・インデックスの個数を示す
   } BITMAPINFOHEADER;
```

BITMAPINFOHEADER 構造体は，デバイス独立型ビットマップ（DIB）の寸法とカラー形式に関する情報を格納する

（**b**）BITMAPINFOHEADER 構造体

```
typedef struct tagRGBQUAD {      /* rgbq */
    BYTE rgbBlue;         // 青の強度を示す
    BYTE rgbGreen;        // 緑の強度を示す
    BYTE rgbRed;          // 赤の強度を示す
    BYTE rgbReserved;     // 使われない．0 に設定する
   } RGBQUAD;
```

RGBQUAD 構造体は，赤，緑，青の強度で構成される色を記述する

（**c**）RGBQUAD 構造体

```
typedef struct tagBITMAPINFO {  // bmi
    BITMAPINFOHEADER bmiHeader;
    RGBQUAD          bmiColors[1];
   } BITMAPINFO;
```

BITMAPINFO 構造体は BITMAPINFOHEADER と RGBQUAD（1 色分）で構成される

（**d**）BITMAPINFO 構造体

その画素の色の強度が青，緑，赤の順に各色 1 バイトで並びます．また，横の画素数が 101 の場合，101 × 3 = 303 バイトとなり，パッディングのための 1 バイトが追加されます．メモリの量や CPU の性能が高くなった現在のパソコンでは，24 ビット・カラーが多く使われます．

BMP の読み込みと表示を行う例を**リスト 3-1** に示します．コマンドラインの引数から読み出すファイルの名前を得て，そのファイルをオープンします．BMP ファイルはバイナリ・ファイルなので，読み込みには `fread()`を使います．

ファイル情報のヘッダ，ビットマップ情報のヘッダは，それぞれ対応する構造体が用意されているので，