動きをとらえる
動画処理の応用

動画処理の解説を進めながらというのも変ですが，「動画処理手法とはいかに？」といわれると即答に困ってしまいます．画像処理なら輪郭を出して測ったり，マッチングしたり，あるいは空間周波数やテクスチャ解析と，いくつかの定番が思い浮かびますが，動画の場合は「動きを求めて…えーっと」となってしまいます．動画の場合，多次元の相関やオプティカル・フロー(用語)など理論的にはいくつかの方法がありますが，実際の要求との間にはやはり距離を感じてしまいます．また，動画データは，見方を変えれば時間という奥行きをもった立体データとも言えます．ですから，立体に対する理論や手法を学ぶことも動画データの理解を深める一つの方法だと思います．

動画を使った計測の場合では，動画という3次元のデータのかたまりを3次元のデータとして直接処理することは希で，やはり動画計測，動画処理といえどもフレームごとに切り出した画像データの画像処理を行うのが一般的です．本章では，ごく基本的な処理手法についていくつか紹介します．

7-1　炎の 1/*f* ゆらぎを計測する

身近な対象を使って動画計測を試みます．さまざまな特徴量の取り出し，外部アプリケーションとの連携について紹介します．

● 1/*f* ゆらぎ

1/*f* ゆらぎという言葉がよく知られるようになってずいぶんたちました．1/*f* ゆらぎとは，周波数解析をするとパワーが周波数に逆比例，つまり 1/*f* になっているような現象を指します．癒し系の最古参の一つといってもよいでしょう．実は，自然界の現象の大半は 1/*f* を示します．しかし，いまだにその発生メカニズムや，人に与える「癒し系」の効果については，「少なからず効果は認められるが要因はよくわからない」というのが通説のようです．

こんな見方もできます．逆の状態，つまり 1/*f* 以外の刺激，たとえば，ランダム・ノイズやピークの強い周期性のもの，音ならばランダム・ノイズや純音，画像なら単純な縞模様などは，心地よいものではありません．

この節では，身近なゆらぎ対象としてろうそくの炎を取り上げ，その明るさの強度，動き，色について計測を試みます．計測対象の動画ファイルは，ディジタル・ビデオ・カメラからキャプチャされた DV-AVI フォーマットのファイルを想定します．これら計測システムの概要を**図 7-1** に示します．

● 特徴量の取り出し

適当な長さの動画ファイルを対象に，前述した明るさ，動き，色を取り出すプログラムを組み立てます．このような要求では，動画ファイルから1フレームを切り出し，そのフレームについて各特徴量を計算す

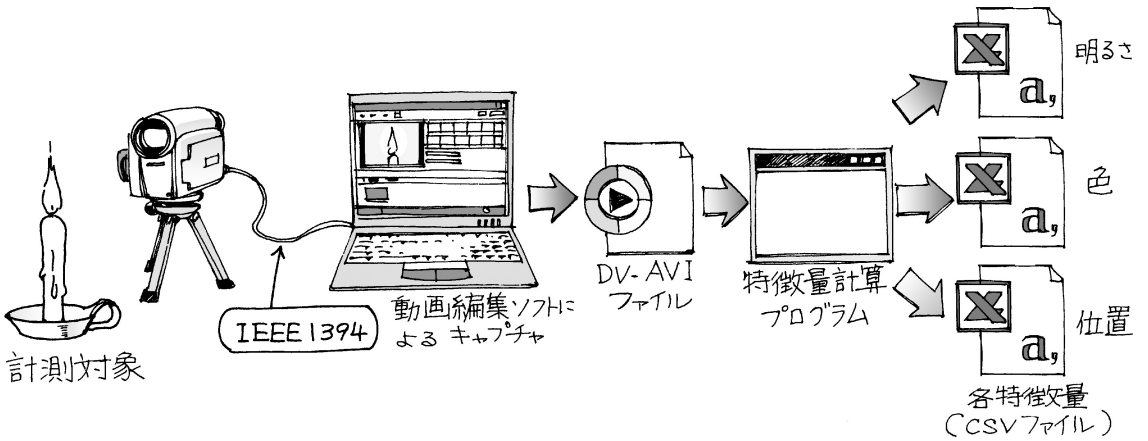

図 7-1　動画処理の応用，炎の 1/*f* ゆらぎを計測する

るという流れが適当でしょう．明るさの計算は，各画素値の総和を求めて明るさとします．動きは，明るさを重みとして重心計算を行います．色は，各画素の XY 色度図の *xy* 値を求め，その炎全体の平均値を計算することにします．

これらを計算するプログラムを**リスト 7-1** に示します．このプログラムでは，第 6 章で紹介した DirectShow を使って動画ファイルから 1 フレームを展開します．はじめに，フィルタ・グラフ・マネージャとサンプル・グラバ・フィルタを生成し，サンプル・グラバをフィルタ・グラフに組み込みます．そのあと，再生する動画ファイルを設定します．この時点で動画ファイルに対応したフィルタが構築されます．そのあとサンプル・グラバを開始モードにして，レンダリングのループに入ります．

ループの中は，Video For Windows と同じように，切り出されたフレーム・データの格納されたバッファに対して必要な計算処理を行います．所定のフレーム数だけ切り出しが済めば，計算されたデータをファイルに書き出し，その後，フィルタ・グラフなどのリリースを行います．ループ内の計算では炎の下側を計算しない，暗い部分の残留ノイズ抑制のためのしきい値処理など，計測対象に特化した処理を追加して

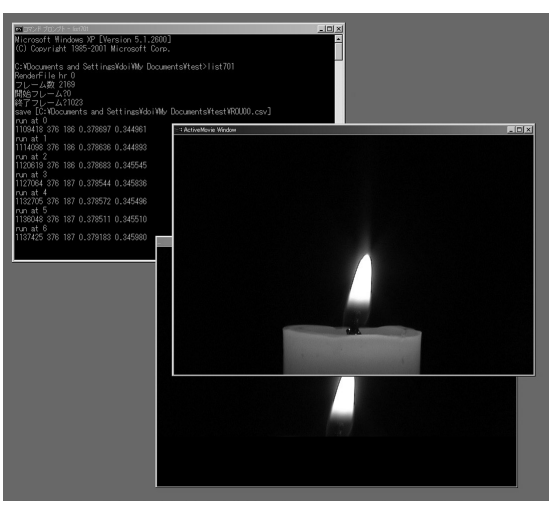

図 7-2　リスト 7-1 の実行画面

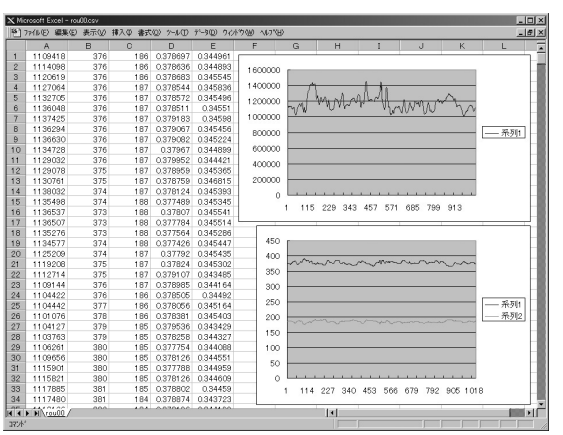

図 7-3　抽出された明るさ，動きのグラフ

リスト 7-1　動画処理の応用～特徴量の取り出し

```
//
// list701.cpp DirectShowを使ったフレームの切り出しの応用
//  ▶list502.h、list502b.cppを合わせて，
//    コンソール・アプリケーションで作成すること
//  ▶[プロジェクト] [リンク]にstrmiids.lib を追加
//  ▶COMの操作はすべてHRESULTを返すが，プログラムを簡単にするため
//    最低限の吟味しか行っていない．

#include <windows.h>
#include <string.h>
#include <dshow.h>          DirectShowのヘッダ・ファイル
#include <qedit.h>          SampleGrabber用
#include <conio.h>          getch()用
#include <stdio.h>
#include "list502.h"

void main ( void )
{
        インターフェース用のポインタ．
        フィルタ・グラフ用
        IGraphBuilder *pigb  = NULL;
        IMediaControl *pimc  = NULL;
        IMediaSeeking *pims  = NULL;
        サンプル・グラバ用
        IBaseFilter    *pF   = NULL;
        ISampleGrabber *pGrab = NULL;    これらは後で解放すること

        IMG0 img00;                      表示ウィンドウ用の構造体
        BYTE  *buffer;          外部バッファ
        AM_MEDIA_TYPE   amt;
        WCHAR filename [ MAX_PATH ] ;
        HRESULT hr;

        img00.hi =  (HINSTANCE) GetWindowLong ( HWND_DESKTOP, GWL_HINSTANCE ) ;
        img00.x = 100; img00.y = 100;
        gr_reg () ;          表示用ウィンドウの登録

        CoInitialize (NULL) ;          COMの準備
        FilterGraphの初期化
        CoCreateInstance ( CLSID_FilterGraph, NULL, CLSCTX_INPROC_SERVER,
                           IID_IGraphBuilder,  (void **) &pigb) ;
        フィルタ・グラフのインターフェースを得る
        pigb -> QueryInterface ( IID_IMediaControl,  (void **) &pimc ) ;
        pigb -> QueryInterface ( IID_IMediaSeeking,  (void **) &pims ) ;
        グラバ・フィルタを作りフィルタ・グラフに追加
        CoCreateInstance ( CLSID_SampleGrabber, NULL, CLSCTX_INPROC_SERVER,
                           IID_IBaseFilter,  (LPVOID *) &pF) ;
        pF -> QueryInterface ( IID_ISampleGrabber,  (void **) &pGrab ) ;
        pigb -> AddFilter ( pF, L"SamGra" ) ;
        グラバ・フィルタの挿入場所の特定のための設定
        ZeroMemory ( &amt, sizeof (AM_MEDIA_TYPE)  ) ;
        amt.majortype  = MEDIATYPE_Video;
        amt.subtype    = MEDIASUBTYPE_RGB24;
        amt.formattype = FORMAT_VideoInfo;
        pGrab -> SetMediaType (&amt) ;

        OPENFILENAME fname;
        static char fn [256] ;
        memset (&fname, 0, sizeof (OPENFILENAME) ) ;
        fname.lStructSize     = sizeof (OPENFILENAME) ;
        fname.lpstrFile       = fn;     パス付きファイル名が格納されるアドレス
```