

---

知能制御システム学

画像処理の基礎 (1)  
— 基礎概念と OpenCV の導入 —

東北大学 大学院情報科学研究科

鏡 慎吾

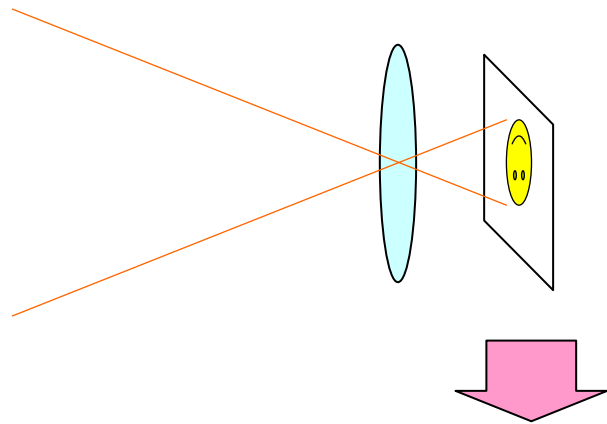
swk(at)ic.is.tohoku.ac.jp

2008.06.24

# 目的

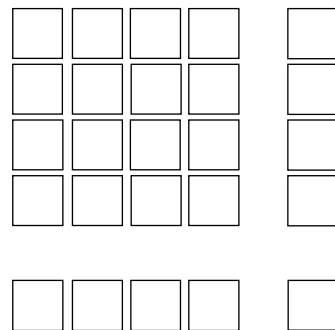
- 機械系および情報科学研究科のカリキュラムでは、画像処理、視覚認識に関する先進的な講義は多数用意されているものの、実践的な「画像処理プログラミングの基礎」に関する講義・演習は必ずしも十分に行われてはいないようである。
- この機会に「実際のプログラミング」を通して、画像処理の基礎の基礎を学ぶ。
- 「実際にプログラミングを行う」レポートを課すので、そのつもりで。

# デジタル画像



撮像面における入射光  
強度のアナログ分布

2次元離散化（「画素」への分割）  
量子化（A/D変換）

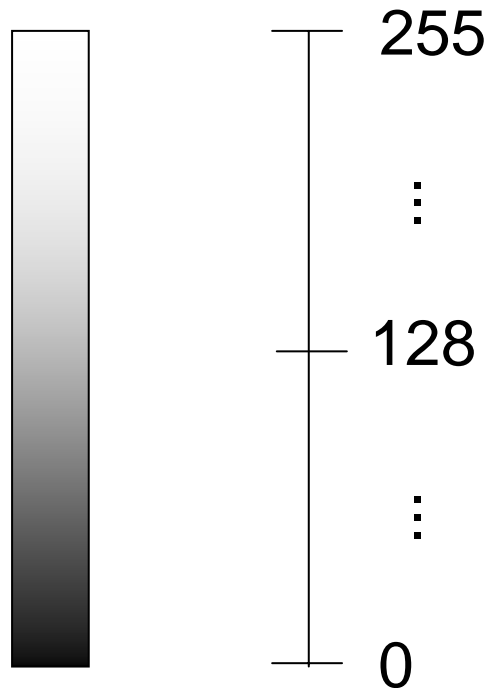


デジタル画像:

画素の明るさを表す値（画素  
値）が2次元に並んだもの

# 画素値

「電荷量」なり「電圧」なりというアナログの量を，デジタルな数字に量子化する．このようにして得られた値を濃度値，濃淡値，グレイレベル，あるいは単に画素値などと呼ぶ



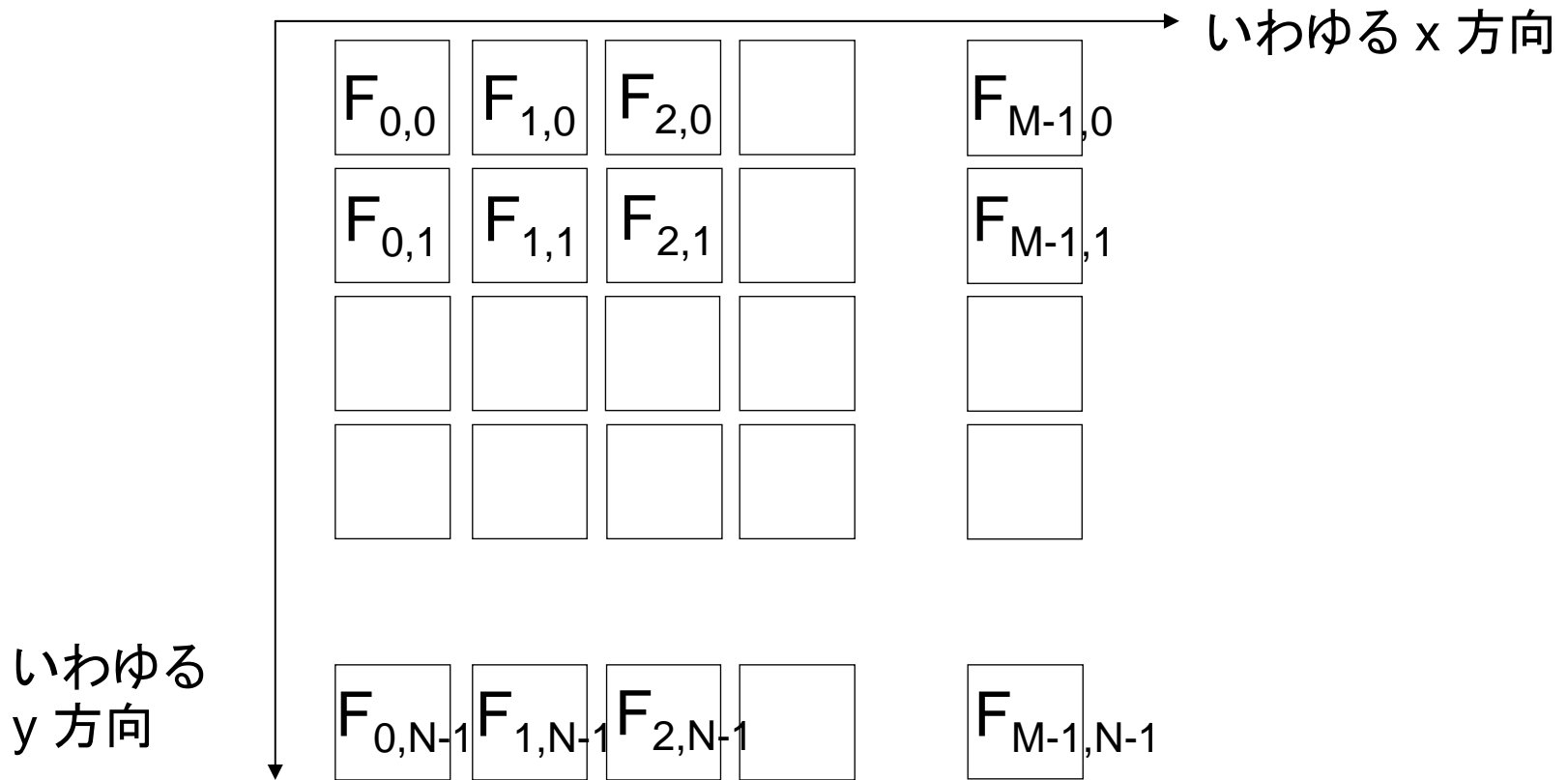
0~255 の 256 階調に一樣量子化する例  
→ 8ビット多値画像  
(8-bit grayscale image)

特に1ビット画像のことを2値画像  
(binary image) と呼ぶ

# デジタル画像の表現

$M \times N$  画素のデジタル画像: 画像上の位置  $(x,y)$  における画素値を  $F_{x,y}$  で表す

$\{ F_{x,y} \}, x = 0, 1, \dots, M-1, y = 0, 1, \dots, N-1$



# Cでの表現

- 8ビットグレイスケール画像であれば, unsigned char 型の配列で表せばよい
- 通常はx方向を優先して走査したいので, x方向に画素が連続して配置されるような構造が望ましい

```
#define M 640  
#define N 480
```

```
unsigned char image1[N][M];  
// (unsigned char の N 要素の配列) の N 要素の配列
```

```
unsigned char *image2[N];  
// (unsigned char の配列へのポインタ) の N 要素の配列
```

```
unsigned char image3[M * N];  
// unsigned char の (M * N) 要素の配列
```

- 2次元配列(またはポインタの配列)をそのまま使うと, 添え字の順番が慣例と逆になる
- 「配列へのポインタ」の配列にする方が, ランダムアクセスの際のアドレス計算の回数は減らせる(その分, 動的確保に手間はかかり, 余分な記憶領域が必要)
- むしろ最初から1次元配列にしてしまうことも多い

```
#define M 640
#define N 480
unsigned char image3[M * N];
```

```
image3[M * y + x] = 30;
// F(x, y) := 30
```

image1, image3

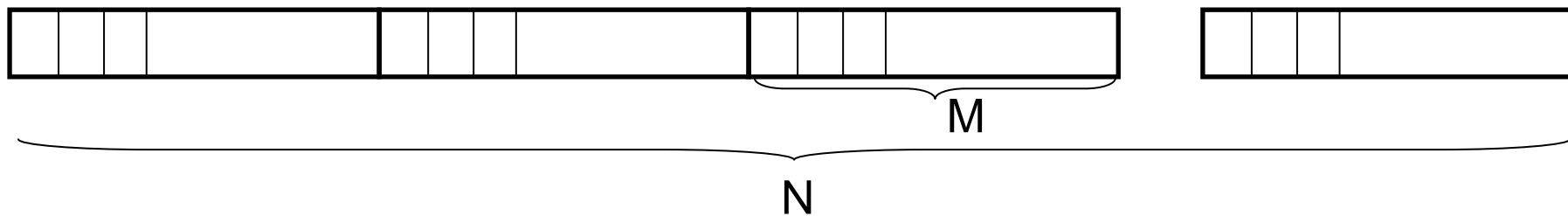
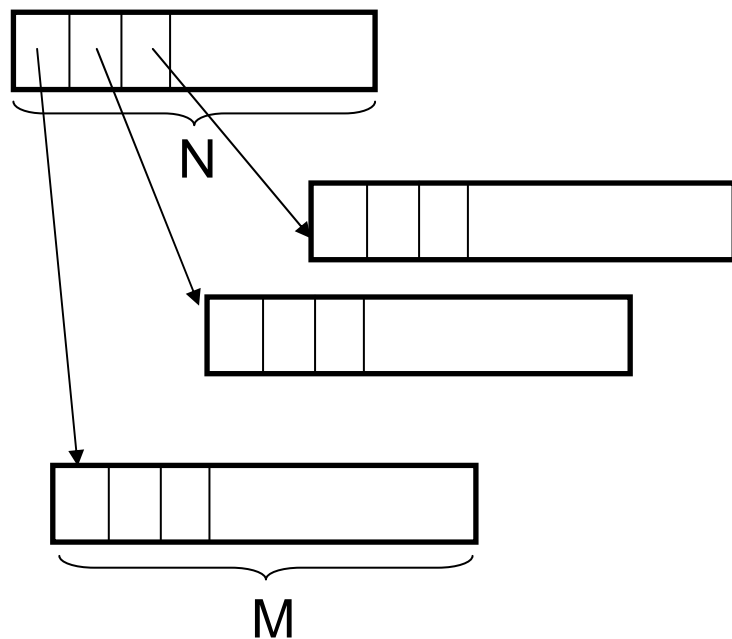


image2





# 典型的な画像処理

## 2値化 (binarization)

```
#define M 640
#define N 480
unsigned char image[M * N];
int i, j;

for (j = 0; j < N; j++) {
    for (i = 0; i < M; i++) {
        if (image[M * j + i] >= 128) {
            image[M * j + i] = 255;
        } else {
            image[M * j + i] = 0;
        }
    }
}
```

# 画像処理ライブラリ

このように自前で画像処理を書くことはもちろん可能だが、標準的なライブラリを利用すると(あるいは少なくともデータ構造だけでも標準的なライブラリに合わせておくと)、いろいろと便利

ここでは OpenCV を紹介する

- 以下で示すプログラム例は、説明用に簡略化されたものであり、そのまま実用に耐えるものではない点に注意すること
- OpenCVに標準関数として用意されているものを敢えて再実装している場合もある

# この講義での実行環境

他の環境を使いたい人は、適当に脳内変換して下さい  
(私自身も決してこれらの環境での作業が得意なわけでは  
ないです。というかどちらかというと苦手です)

- OS: Microsoft Windows
- 開発環境.: Microsoft Visual Studio Professional 2005
- 言語: C (C++)
- ライブラリ: OpenCV

<http://sourceforge.net/projects/opencvlibrary/>  
<http://opencvlibrary.sourceforge.net/>

# Visual Studio

- 講義では Microsoft Visual Studio Professional 2005 を使用. インストール法などの説明は省略
- 注意: フリーのバージョンの Visual Studio (Visual Studio 2005 Express など) には, コードの最適化に制限があるようである(従って, 性能の評価がしにくいかも知れない)

# OpenCV のインストール

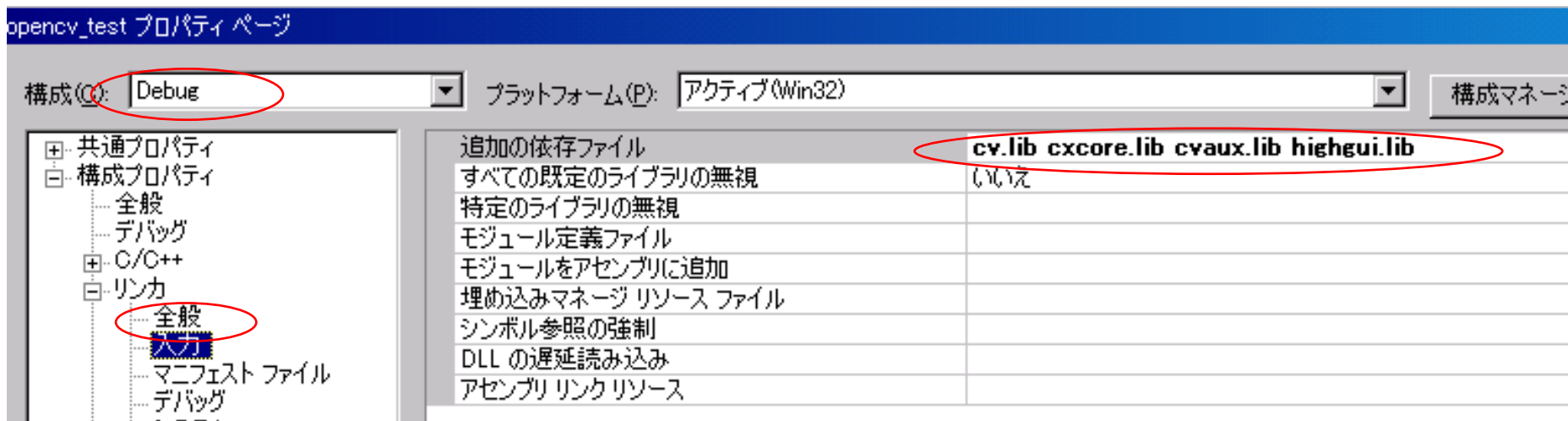
- opencv-win をダウンロードしてインストール
- Visual Studio の ツール-オプション にて,  
プロジェクトおよびソリューション – VC++ ディレクトリを選び,  
「インクルードファイル」に以下を追加
  - C:¥Program Files¥OpenCV¥cv¥include
  - C:¥Program Files¥OpenCV¥cxcore¥include
  - C:¥Program Files¥OpenCV¥otherlibs¥highgui
  - C:¥Program Files¥OpenCV¥cvaux¥include
  - C:¥Program Files¥OpenCV¥otherlibs¥cvcam¥include
- 「ライブラリファイル」に以下を追加
  - C:¥Program Files¥OpenCV¥lib
- 「ソースファイル」にも対応するディレクトリを追加しておくとも便利かもしれない
- 環境変数 PATH に以下を追加
  - ;C:¥Program Files¥OpenCV¥bin
- See e.g.  
<http://opencvlibrary.sourceforge.net/VisualC%2B%2B>

# 新規プロジェクトの作成

- プロジェクトのプロパティで構成プロパティ-リンカ-入力の「追加の依存ファイル」に以下を追加

cv.lib cxcore.lib highgui.lib  
(cvaux.lib cvcam.lib ← 必要に応じて)

- Release構成でも設定しておくのを忘れないこと



# 典型的な流れ(単一の画像)

```
#include <stdio.h>
#include "cv.h"
#include "cxcore.h"
#include "highgui.h"

int
main()
{
    // Initialize

    // Load Image

    // Process the Image

    // (Display or output the result)

    // (Finalize)

    return 0;
}
```

# Sample Codes

---

sample programs:

- single\_image.cpp
- single\_image\_load.cpp



# IplImage Structure

```
typedef struct _IplImage {  
    ...  
    int nChannel;  
    int depth;  
    int origin;  
    int width;  
    int height;  
    char *imageData;  
    int widthStep;  
    ...  
}
```

# IplImage Structure

nChannel

- 1: grayscale images
- 3: RGB color images

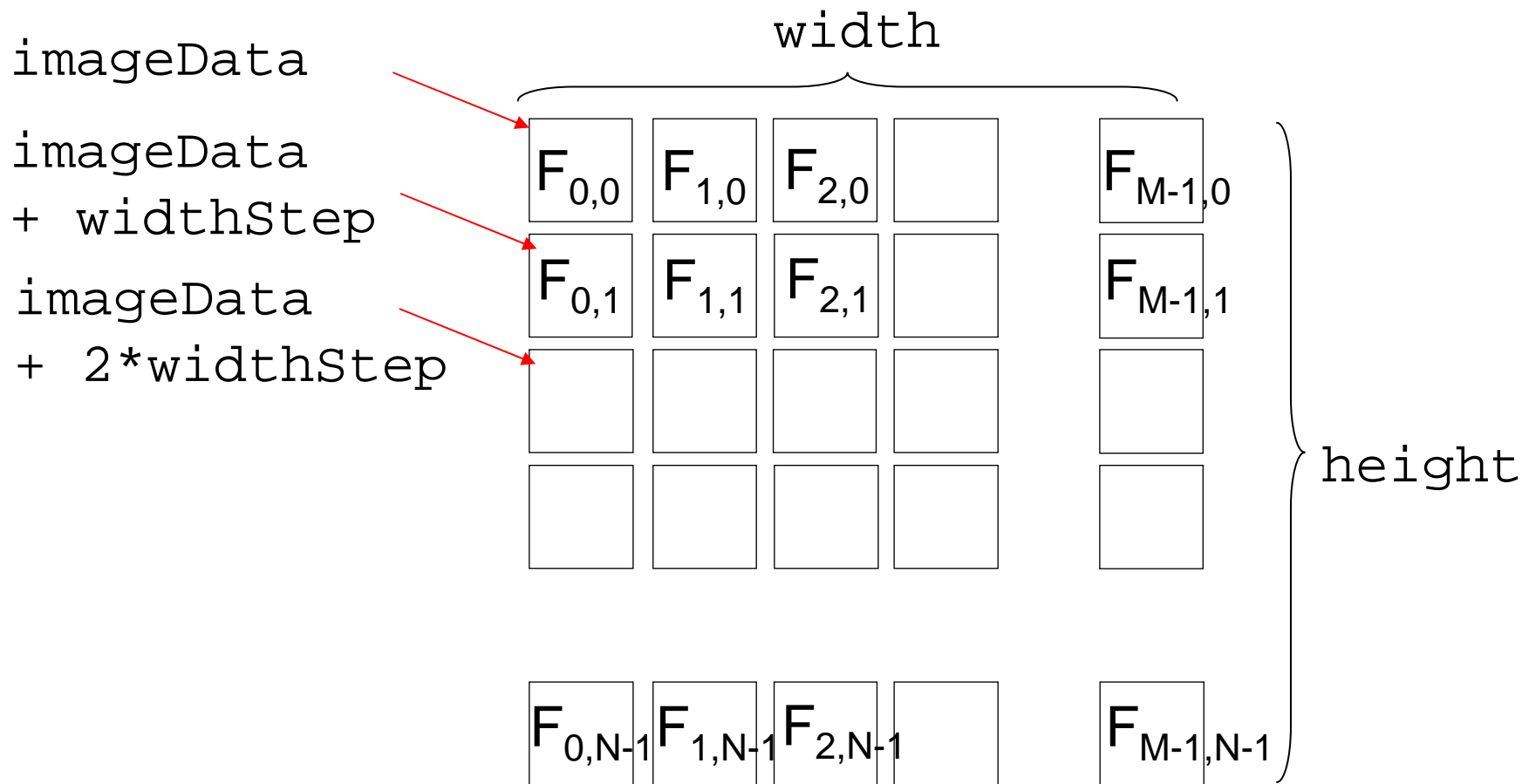
depth

- IPL\_DEPTH\_8U: 8-bit natural number (i.e. uchar)
- IPL\_DEPTH\_64F: 64-bit real number (i.e. double)

origin

- 0: origin at top-left
- 1: origin at bottom-left (e.g. Windows BMP)

# imageData (モノクロ画像用)



1-dimensional array of char **continuous in the x (width) direction**

```
for depth = IPL_DEPTH_8U and nChannel = 1,
```

```
    Fx,y ~ ( *(uchar *) (img->imageData  
                    + y * img->widthStep  
                    + x) )
```

```
#define PIXVAL(iplimagep, x, y) ¥  
    ( *(uchar *) ((iplimagep)->imageData ¥  
      + (y) * (iplimagep)->widthStep + (x)) )
```

```
PIXVAL(img, 10, 20) = 30;  
x = PIXVAL(img, 128, 150);
```



# 典型的な流れ(複数フレーム)

```
int
main()
{
    // Initiallize

    while (1) {

        // capture image from camera

        // Process the Image

        // (Display or output the result)

    }

    // (Finalize)

    return 0;
}
```

# USBカメラから画像を得る

```
CvCapture *capture;  
IplImage *frame;  
  
capture = cvCaptureFromCAM(0); // 0: the first camera  
/* returns NULL if no camera is found */  
  
frame = cvQueryFrame(capture);  
/* Note that this image must not be modified, so copy it  
   before you modify */
```

# Sample Code

---

sample program (for USB camera):

- interframesub.cpp



# 実行時間を手軽に測る

```
#include <windows.h>

LARGE_INTEGER freq;
LARGE_INTEGER cn1, cn2;
double elapsed_time_in_microseconds;

if (QueryPerformanceFrequency(&freq) == 0) {
    fprintf(stderr, "cannot use performance counter¥n");
    return 1;
}

QueryPerformanceCounter(&cn1);

// do something

QueryPerformanceCounter(&cn2);

elapsed_time_in_microseconds =
    (1000000.0 * (cn1.QuadPart - cn2.QuadPart)) / freq.QuadPart);
```

# Sample Code

sample program:

- interframesub\_time.cpp

PerformanceCounter を使ったごく簡単 (= 手抜き) な時間測定の例.

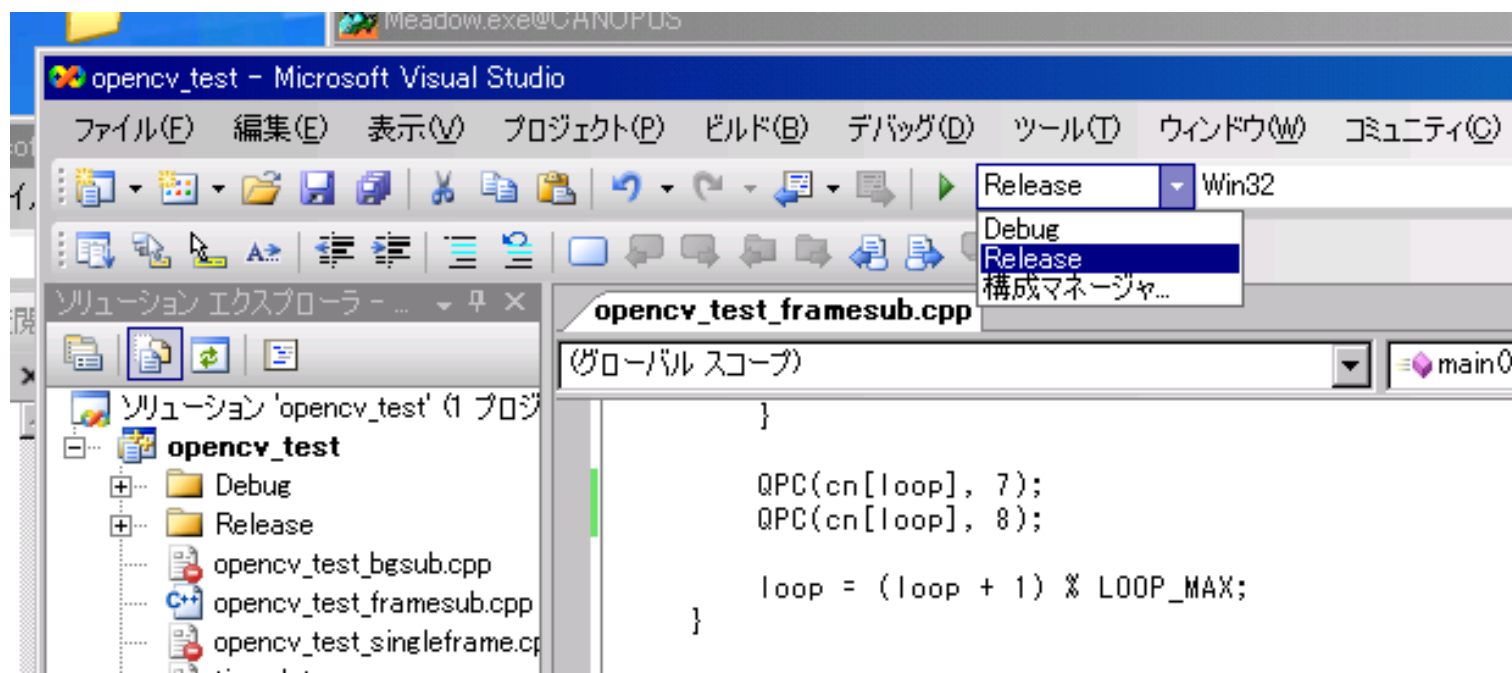
プログラム終了後, 最終フレームの測定結果がコンソールに表示される.

最後の100フレーム分が `time.dat` ファイルに書き出される. 適当なグラフ描画ツールで表示してもよい.

```
gnuplot> plot "time.dat"
```

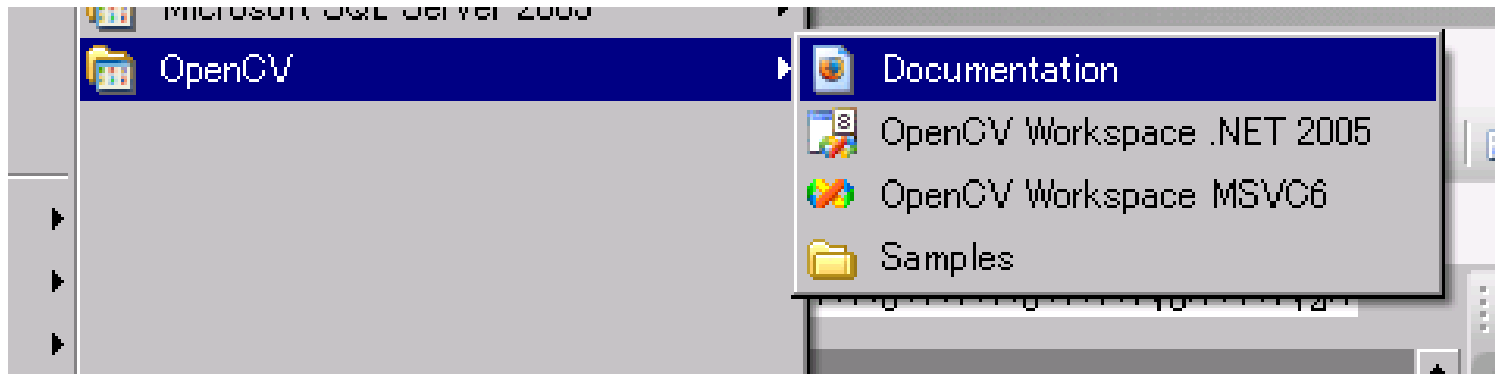
# Debug 構成と Release 構成

Debug 構成だと、あまり最適化が効いていない(かもしれない)ので注意. 性能を測定するときは Release 構成でビルド・実行すること (設定を忘れないように)



# OpenCV ドキュメント

スタートメニューから HTML のリファレンスマニュアルを開ける



最近はウェブ上で、あるいは書籍で日本語の情報も多い

# References

- [1] 田村: コンピュータ画像処理, オーム社, 2002.
- [2] <http://sourceforge.net/projects/opencvlibrary/>
- [3] <http://opencvlibrary.sourceforge.net/>
- [4] 奈良先端科学技術大学院大学 OpenCVプログラミングブック制作チーム: OpenCVプログラミングブック, 毎日コミュニケーションズ, 2007.