知能制御システム学

画像処理の基礎 (1)
— OpenCV による実践 —

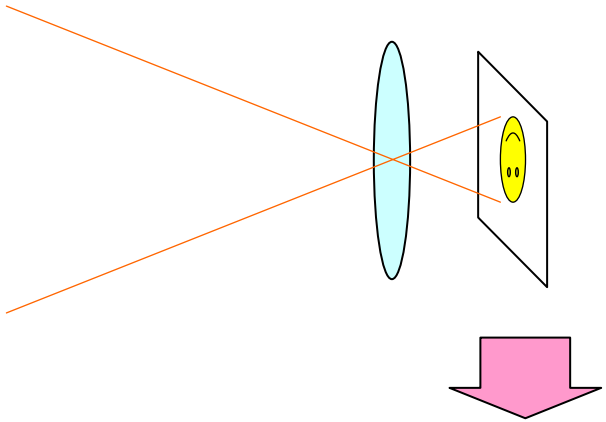
東北大学 大学院情報科学研究科
鏡 慎吾
swk(at)ic.is.tohoku.ac.jp

2007.06.12

目的

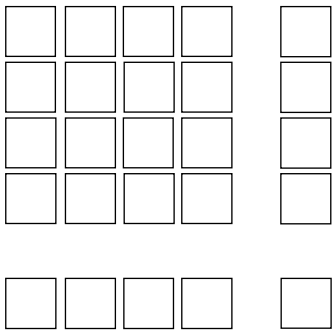
- 機械系および情報科学研究科のカリキュラムでは，画像処理，視覚認識に関する先進的な講義は多数用意されているものの，実践的な「画像処理プログラミングの基礎」に関する講義・演習は必ずしも十分に行われてはいないようである．
- この機会に「実際のプログラミング」を通して，画像処理の基礎の基礎を学ぶ．
- しつこいようだが，本当に基礎的なことしかやらないので，既に画像処理についてよく知っている者，画像処理を専門とする者にとっては退屈な内容になるはずである．（退屈に感じないようでは困る）
- 「実際にプログラミングを行う」レポートを課すので，そのつもりで．

Digital Image



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）



ディジタル画像:

画素の明るさを表す値（画素値）が2次元に並んだもの

Quantization

「電荷量」なり「電圧」なりというアナログの量を，ディ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

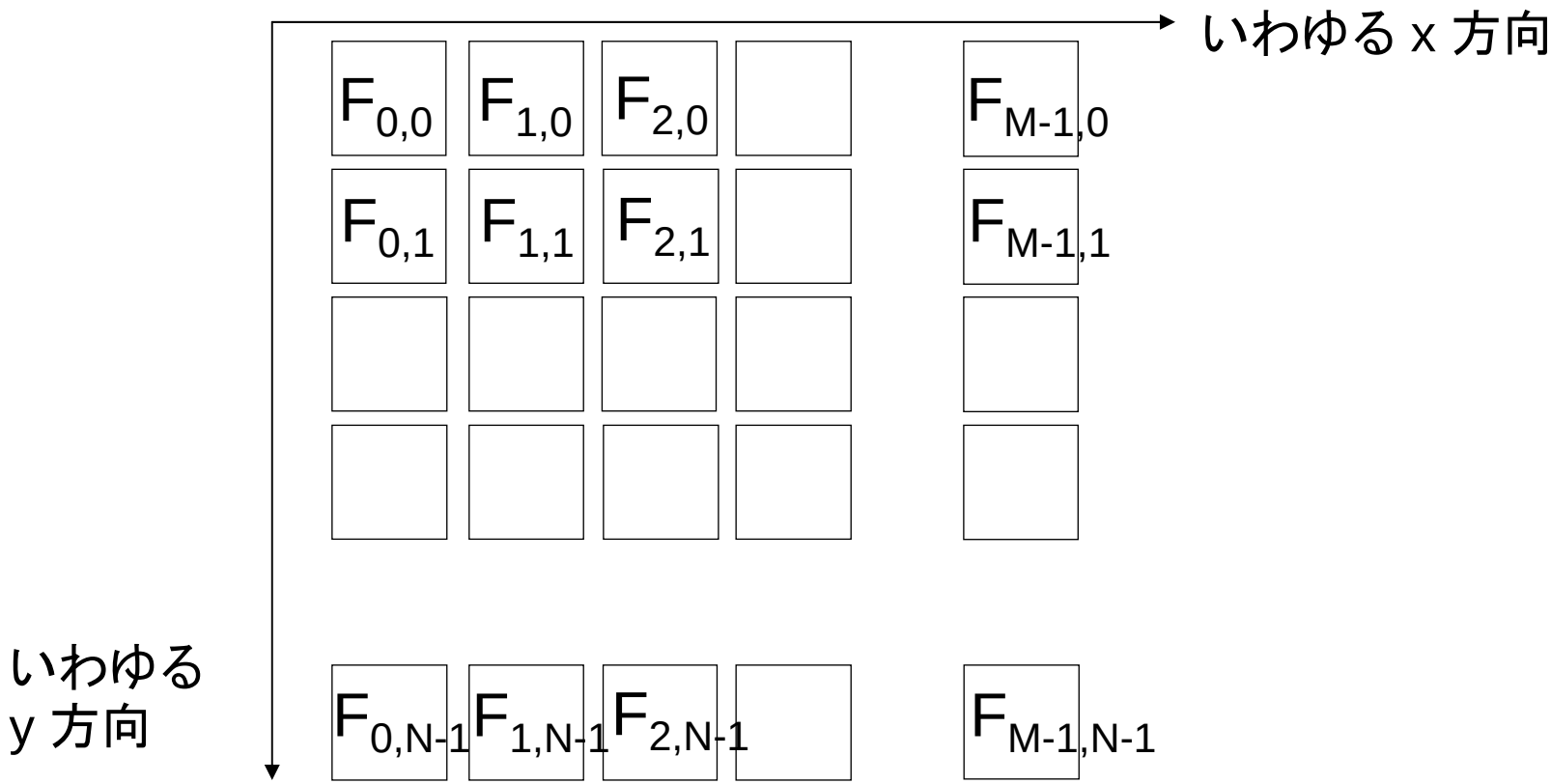
0～255 の 256 階調に一様量子化する例
→ 8ビット多値画像
（8-bit grayscale image）

特に1ビット画像のことを2値画像 (binary image) と呼ぶ

Expression of Digital Images

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

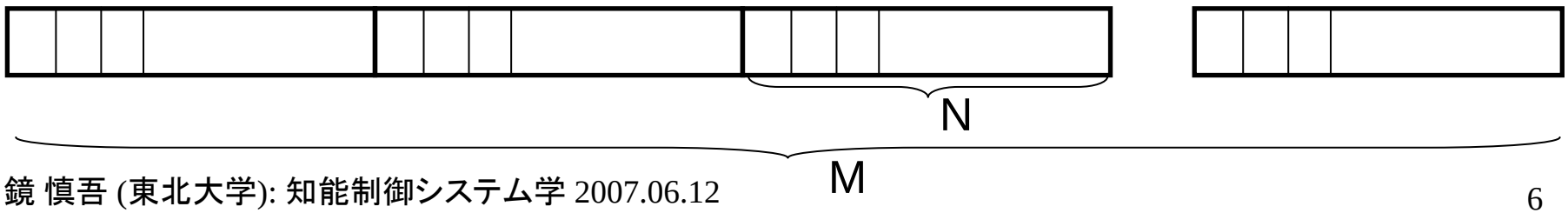
$\{ F_{x,y} \}$, $x = 0, 1, \cdots, M-1$, $y = 0, 1, \cdots, N-1$



Expression in C

- 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
- 2次元配列で表すのが自然なように思えるが，Cの場合は，実はあまり便利ではない
  - 添え字の順を慣例どおりに書くと，y方向がメモリ上で連続に並んでしまう.（普通はx方向に走査したい）
  - 2次元配列は動的に確保・開放するのが面倒

```
#define M 640
#define N 480
unsigned char image[M][N];
// M-elements array of N-elements array of unsigned char
```

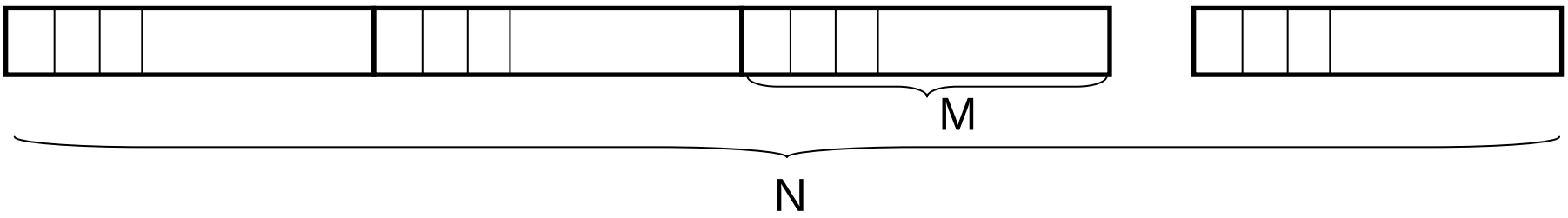


Expression in C

- というわけで，Cの場合は1次元配列として確保し，自前で2次元的にアクセスする方が（個人的には）おススメ

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

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



Typical Image Processing

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;
        }
    }
}
```

Image Processing Libraries

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

※ 以下で示すサンプルプログラムはいずれも説明用に簡略化されたものであり，実用に耐える品質のものではない

Environment for Examples

他の環境の方がわかりやすい者は，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
- Devel. Env.: Microsoft Visual Studio Professional 2005
- Lang: C (C++)
- Image Processing Library: OpenCV
  http://sourceforge.net/projects/opencvlibrary/
  http://opencvlibrary.sourceforge.net/

Installing Visual Studio

- I use Microsoft Visual Studio Professional 2005 in this lecture. The installation process is *not lectured*.

- Note that the free version (Visual Studio 2005 Express) seems to have restrictions in optimizing (So it might be difficult to use it for performance evaluation of your codes)

Installing OpenCV

- Download and install opencv-win
- In Visual Studio, open Tool – Option,
  Choose Project and Solution – VC++ directories, and
  Set for include directories:
  ```
  C:¥Program Files¥OpenCV¥cv¥include
  C:¥Program Files¥OpenCV¥cvaux¥include
  C:¥Program Files¥OpenCV¥cxcore¥include
  C:¥Program Files¥OpenCV¥otherlibs¥highgui
  ```
  Set for library directories:
  ```
  C:¥Program Files¥OpenCV¥lib
  ```
- Append
  ```
  ;C:¥Program Files¥OpenCV¥bin
  ```
  to the environment variable PATH
- See e.g.
  - http://chihara.naist.jp/people/2004/kenta-t/OpenCV/pukiwiki/index.php?%A5%A4%A5%F3%A5%B9%A5%C8%A1%BC%A5%EB
  - http://opencvlibrary.sourceforge.net/VisualC%2B%2B

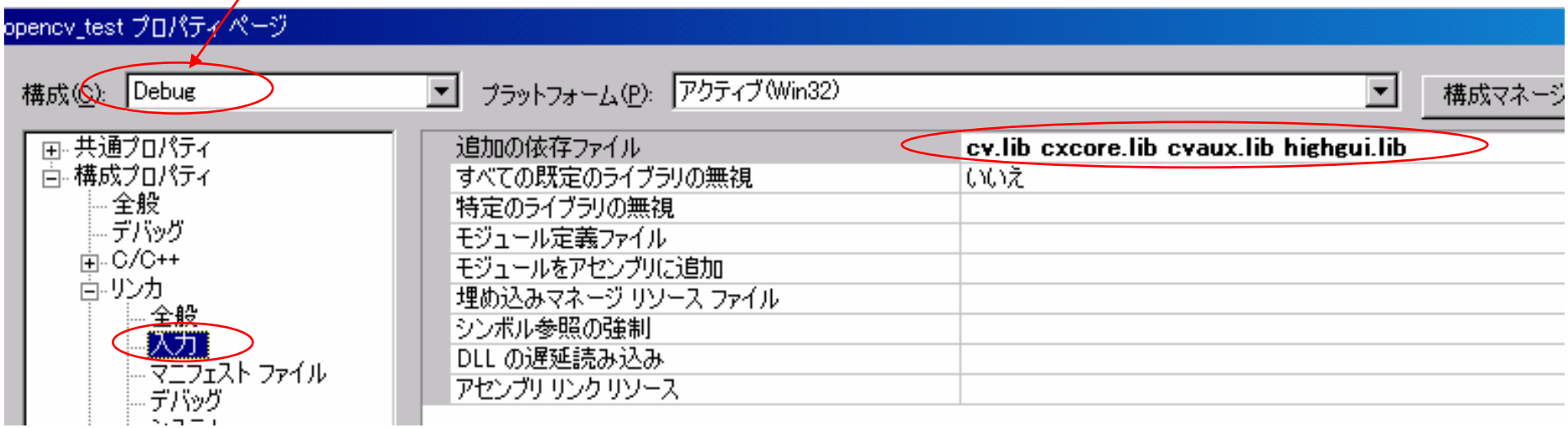
Creating a new project

- Choose Win32 Console Application
- Set the property of the project (see the fig. below):

```
cv.lib cxcore.lib cvaux.lib highgui.lib
```

- Don't forget to set it in "Release" configuration

set for both "Debug" and "Release"



Typical Image Processing Flow (Single Image)

```
#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:

- imageproc_singleframe.cpp
- imageproc_singleframe_loadimage.cpp

IplImage Structure

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

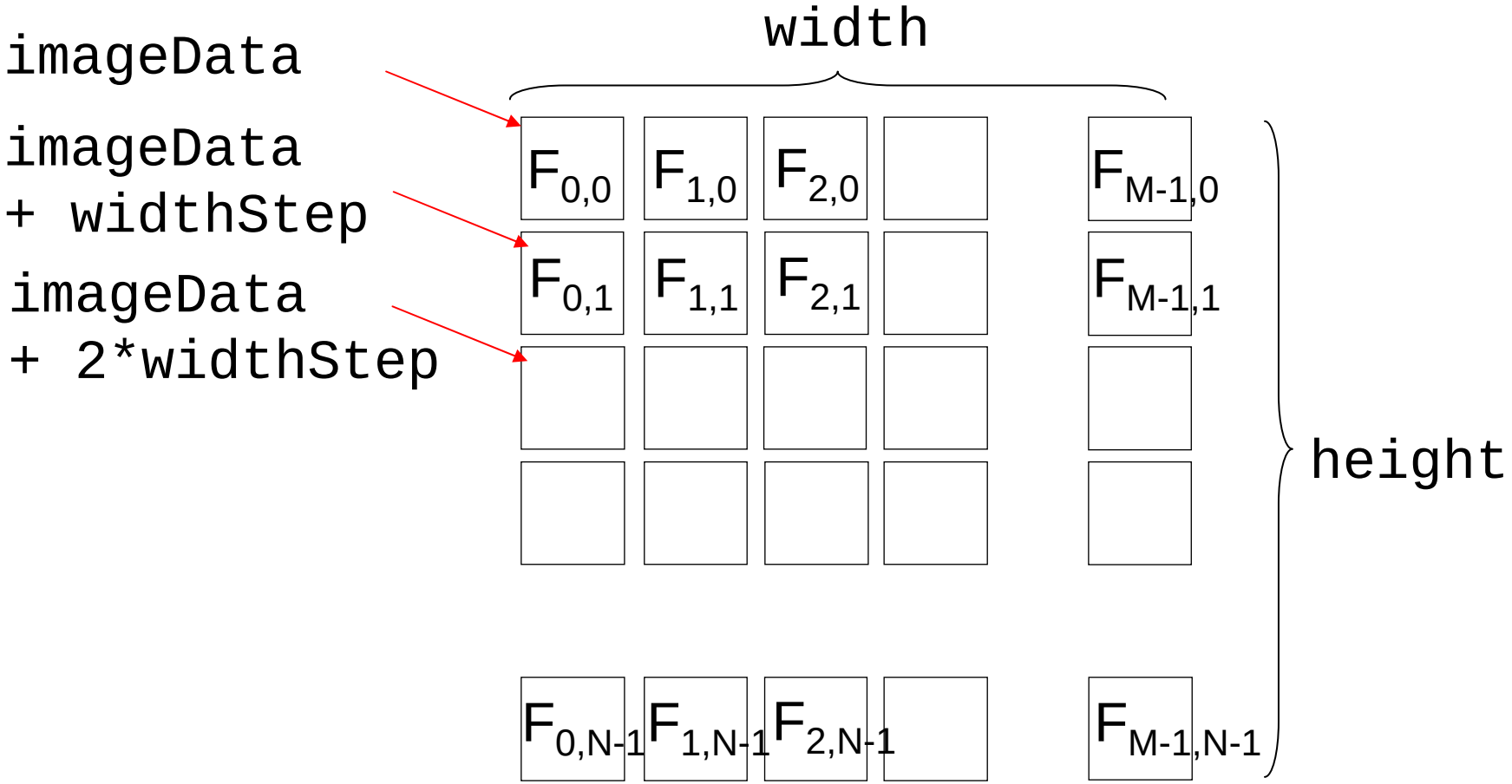
IplImage Structure

```
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 for Grayscale (Monochrome) Image



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

imageData for Grayscale (Monochrome) Image

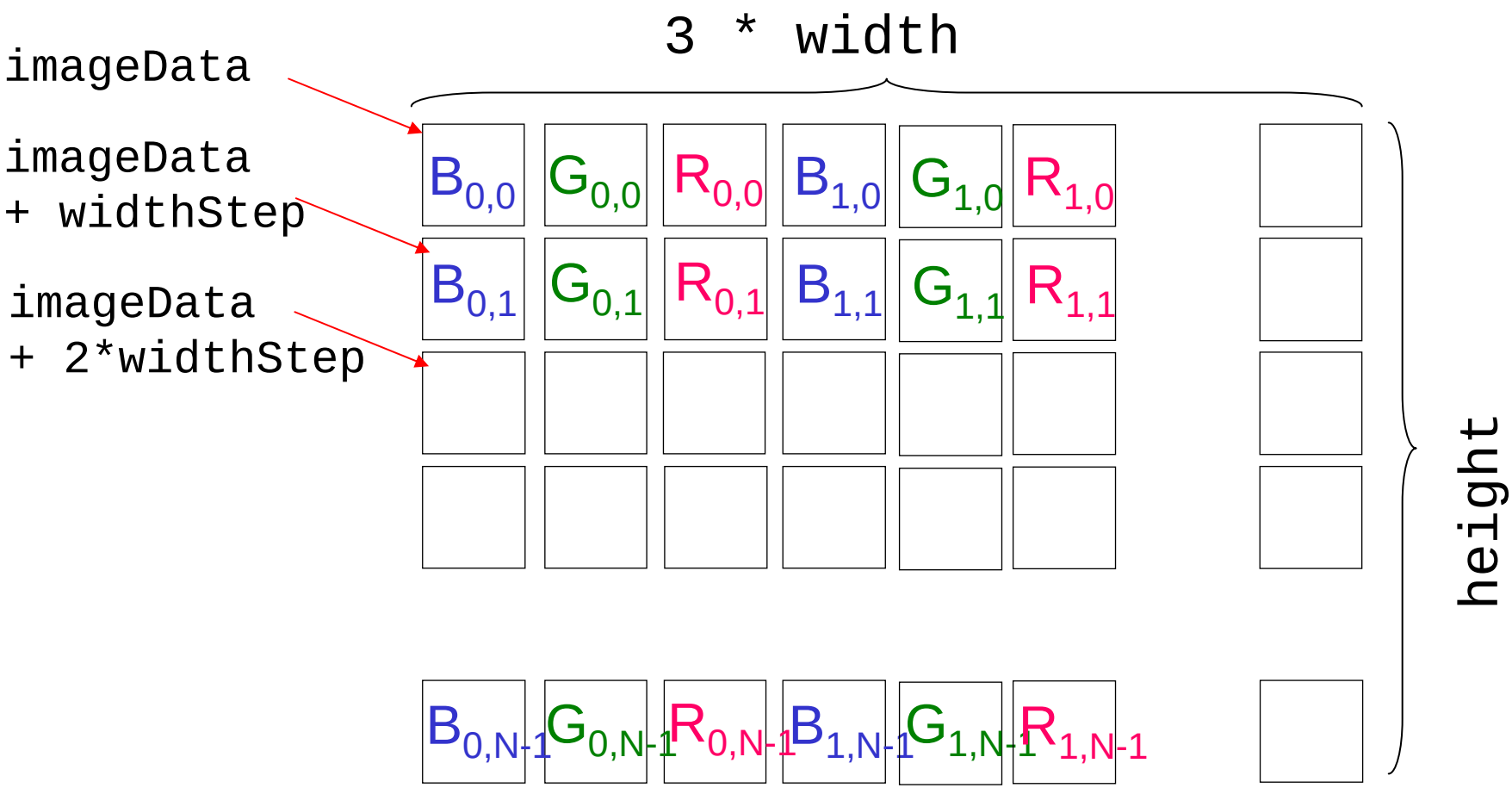
for depth = IPL_DEPTH_8U and nChannel = 1,

$F_{x,y}$ ~ (*(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);
```

imageData for RGB color Image



Typical Image Processing Flow (multiple frames)

```
int
main()
{
  // Initiallize

  while (1) {

    // capture image from camera

    // Process the Image

    // (Display or output the result)

  }

  // (Finalize)

  return 0;
}
```

Obtaining images from a USB camera

```
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):

- imageproc_interframesub.cpp

Simple Method of Measuring Time

```
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:

- imageproc_interframesub_time.cpp

The measurement result for the last frame is displayed in the console.

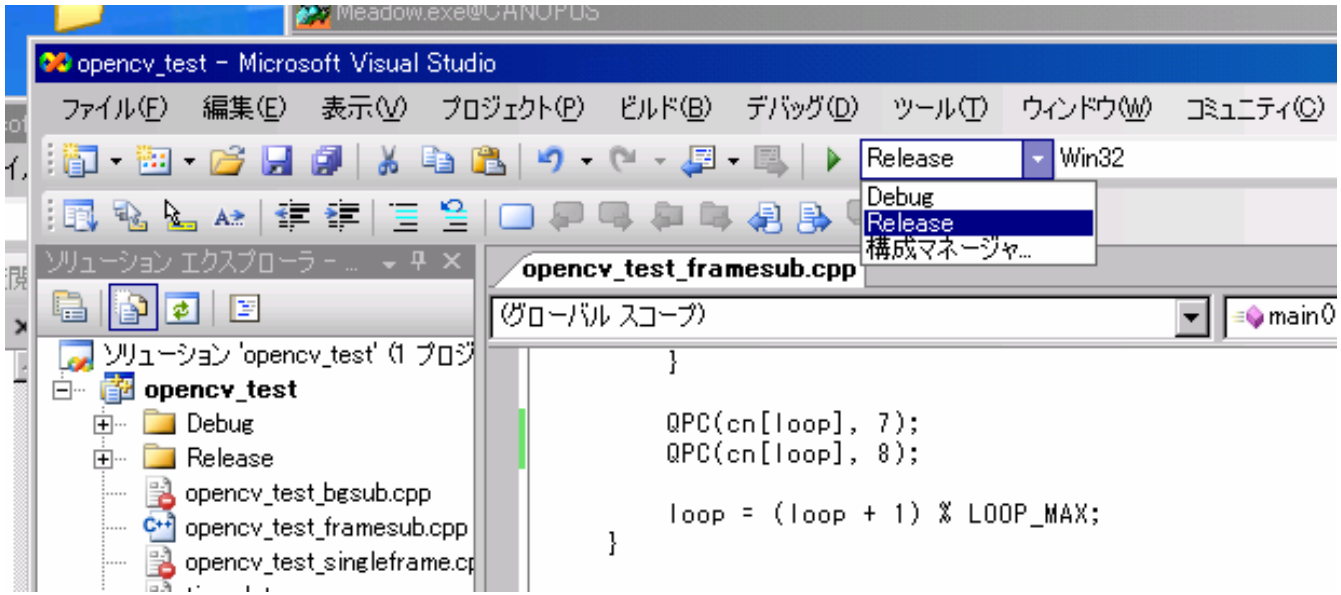
The results for the last 100 frames are written to `time.dat` file. You can visualize it with some plotting software, say, gnuplot.

```
gnuplot> plot “time.dat”
```

Debug and Release

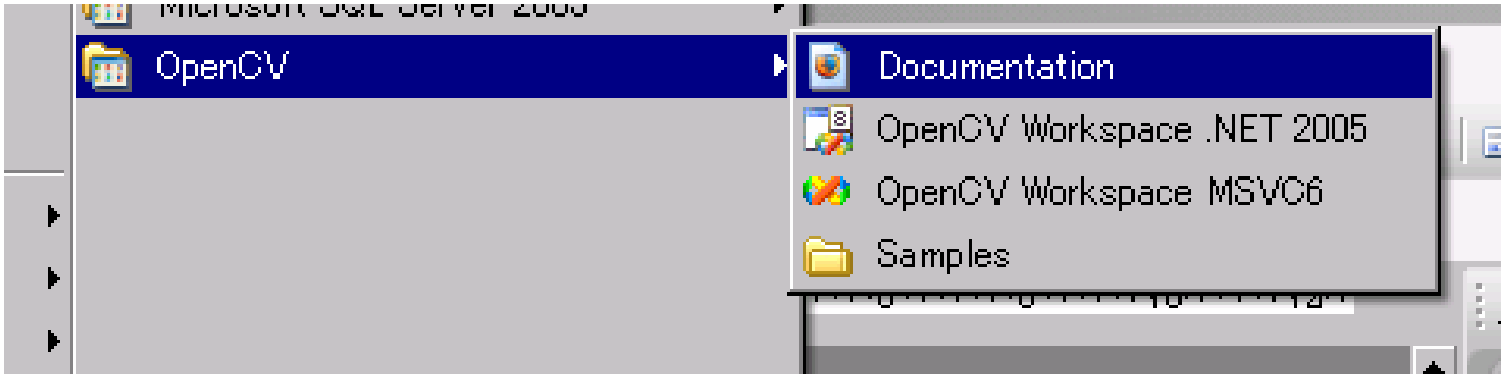
Object files generated in Debug configuration are not strongly optimized!

Don’t forget to choose Release configuration when you measure time. (Don’t forget to add linker inputs to Release configuration)



OpenCV Documents

You can open HTML documents and samples from the Start menu (or see the official web site)



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/