産業用デジタルカメラを利用した動画像処理プログラミング環境の構築

田　中　卓　史（情報工学科）
吉　田　位　史（情報工学専攻）

Constructing Programming Environment for Time-Varying Image Processing Using Industrial Digital Cameras

Takushi TANAKA (Department of Computer Science and Engineering)
Narifumi YOSHIDA (Graduate School of Communication and Computer Engineering)

Abstract

We have developed a programming environment of image processing for industrial digital cameras. These cameras directly output digital data for image processing, while NTSC-video cameras needed a converting process from analog data to digital data. As those cameras allow various settings such as resolutions and frame rates, they need various control signals to set up. Our programming environment generates these control signals through GUI. Using the environment, programmers can easily write programs for image processing without detailed knowledge on IIDC standard.

Keywords: *DCAM, IIDC, linux, image processing, IEEE-1394*

1．はじめに

近年，動画像処理においてNTSCビデオカメラに代わりIIDC（Instrumentation and Industrial Digital Camera）規格[2]に沿ったDCAMと呼ばれる産業用カメラが利用されるようになっている。NTSCカメラは解像度やフレームレートが固定であったのに対し，DCAMはカメラのモデルごとに異なるビデオモードやフレームレートを複数持っており，適宜モードを選択することができる。これらのビデオモード，フレームレート，バスの転送速度は相互に依存しており，選択可能な項目は他項目の選択状況に依って変化する。DCAMはその柔軟性から制御に必要な設定量が多くなる。そのため，画像処理に注力したい時間の多くをDCAMの制御に費やさなければならない。また，画像処理を行うたびに制御プログラムを書いていたのでは画像処理プログラムを資産として残し難いという問題もでてくる。Linux上にはCoriander[4]と呼ばれるDCAMによりキャプチャした動画を閲覧するソフトが存在する。一方，画像処理を目的としたDCAM制御ソフトウェアはまだ存在していない。そこで制御に必要な部分をモジュール化してGUI制御可能にし，画像処理部をプラグイン化することでDCAMの詳細に煩わされずに画像処理に専念できるプログラミング環境をLinux上に構築した。

平成20年10月31日受付

2．DCAM

2.1　概要

本研究で用いたDCAMはPoint Grey社のFlea2[6]と呼ばれる製品で，IEEE-1394インタフェースを使うカメラとして広く知られている。この製品は撮像素子から得られた画像データをそのまま出力するか，もしくは以下のカラーコーディングを行い，8通りの形で出力することができる。

- RGB（24bit）
- YUV411（6 bit），YUV422（8 bit），YUV444（12 bit）
- MONO8（8 bit），MONO16（16bit）
- RAW8（8 bit），RAW16（16bit）

RGBは赤，緑，青成分の情報をそれぞれ1バイトずつ出力する。YUVはYが輝度信号，Uが輝度信号と青信号の差，Vが輝度信号と赤信号の差を表しており，YUVに続く3桁の数字でそれぞれの成分のビット数を表している。

RAWはカメラの撮像素子から得られるデータを無変換で出力する。カメラの撮像素子はベイヤーパターンと呼ばれるR.G.Bの画素の並びから構成されている（図1）。特定の位置の色情報は周辺の画素の値から計算される（デモザ

イキング処理)。

MONO は各画素の位置における輝度情報を表すもので周辺の R.G.B の値から計算して出力される。MONO データの量は RAW と同じ量が送られてくる。RAW モードで送られてきた画像データから元のピクセル数の R.G.B 画像データを復元するとデータ量は 3 倍になる。

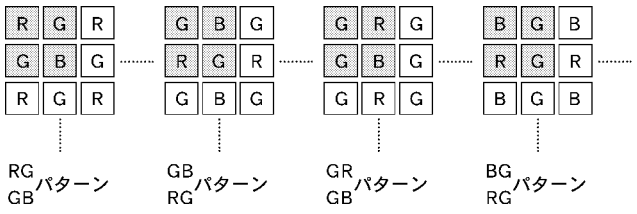

図 1　ベイヤーパターン
Fig. 1　Bayer pattern

2.2　ビデオモードと format

IIDC 規格には画像サイズとカラーコーディングの組み合わせにより次のモード（format_0 ～ 7）が定義されている。

- format_0 ：VGA non-compressed（最大640×480）
- format_1 ：SVGA　non-compressed（最 大1024×768）
- format_2 ：SVGA　non-compressed（最 大1600×1200）
- format_3 ～ 5 ：予約
- format_6 ：Exif（静止画）
- format_7 ：Partial Image Size Format

ここで，format_3 ～ 5 はメーカーが設定した独自のビデオモードのための予約となっており，format_6 は Exif フォーマットの静止画が定義されている。本環境のように画像処理を目的として DCAM を使用する場合には通常，format_0，1，2 または次に示す 7 のいずれかを用いる。

2.3　format_7

format_7 は他のフォーマットとは違って，カメラに映る画像の位置，サイズとカラーコーディングを自由に指定して出力させることができる。このほか 1 フレームのデータサイズやインターバルも設定できる。高速動作が必要な画像処理を行う場合には，format_7 に設定することでフレームレートを上げることができる。

またカラーコーディングを RAW に設定した場合，他と比較してデータの転送量が少なく，カメラでの変換処理を行わないので最も高速になる。使用したカメラではカラーコーディングに RAW8を選択することによって640×480サイズの画像を80fps で取り込むことができる。

本研究ではロボカップ小型リーグの画像処理をテーマとしているため，できるだけ高速な移動体追跡を行う必要がある。この目的のためには RAW モードでベイヤーデータを PC に取り込み，RGB 形式へデコードして利用するのがよさそうである。

3．システム

3.1　環境

開発に使用したライブラリを表 1 に，またその依存関係を図 2 に示す。Linux 上で DCAM を制御する場合，libraw1394[3]と呼ばれる IEEE-1394インタフェースライブラリと libdc1394[5]と呼ばれる DCAM 制御ライブラリを利用する。libraw1394は libdc1394から呼び出される IEEE-1394のドライバプログラムである。libdc1394は IIDC に定められている DCAM の機能を利用するための関数群で，本環境ではこのライブラリの操作を GUI で行うのでプログラマが DCAM の制御をプログラミングしなくて良い。

GUI ライブラリには Qt Software 社（旧 TrollTech 社）の Qt（cute)[7]を利用した。Qt はオープンソースのソフトウェアライブラリで，OS に依存せずに C++と Java でのプログラム開発が可能である。またシグナルとスロットと呼ばれる仕組みが，緩い結合でのオブジェクト間通信を実現しており，ソフトウェアをコンポーネント化しやすいという特徴を持っている。

ユーザがキャプチャ画像や画像処理結果を表示する場合には OpenGL と呼ばれる2D/3D グラフィックスライブラリの利用が便利である。OpenGL は Qt からも直接呼び出すことができ，キャプチャ画像を見ながら取り込み領域を設定するようなことができる。Qt から OpenGL を呼ぶと，マウスをはじめとするウインドウに関連した機能をそのまま利用することができる。

表 1　開発環境
Table 1　Developing environment

OS	PlamoLinux 4.22
1394ドライバ	libraw1394 1.3.0
DCAM ドライバ	libdc1394 2.0.2
GUI ライブラリ	Qt 4.3.0
フレームバッファ	OpenGL 1.2

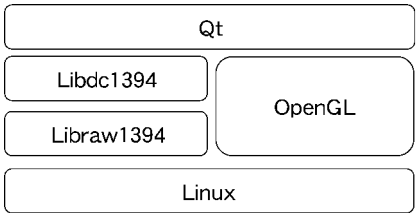

図 2　プログラムライブラリの依存関係
Fig. 2　Dependencies of program libraries

3.2　構成

図 3 に開発したシステムの構成を示す。本環境ではキャ

プチャ部と画像処理部を分離させる目的でプラグイン方式を採用したので，キャプチャ部分を単体で動作させることもできる。この場合は単にキャプチャした画像を表示するソフトウェアとして働く。

本環境においてユーザの作成する画像処理モジュールは本ソフトウェアから接続されるプラグインプログラムとしてコンパイルされる。図4にはカメラから画像データを取得し，処理を終えるまでのデータの流れを示している。

① カメラ設定インタフェースから要求を受けたキャプチャモジュールが指定された設定に従い，DCAMからデータを取り込む。
② 取得した画像はコピーされ，それぞれキャプチャ動画表示モジュールとプラグインインタフェースモジュールに渡される。動画表示用として渡されたものはウインドウに表示される。
③ プラグインインタフェースモジュールに渡された画像は画像処理モジュールへ送られ，プラグイン内で処理される。
④ 画像処理が完了すれば，プラグインインタフェースへ処理結果の画像とデータを返す。

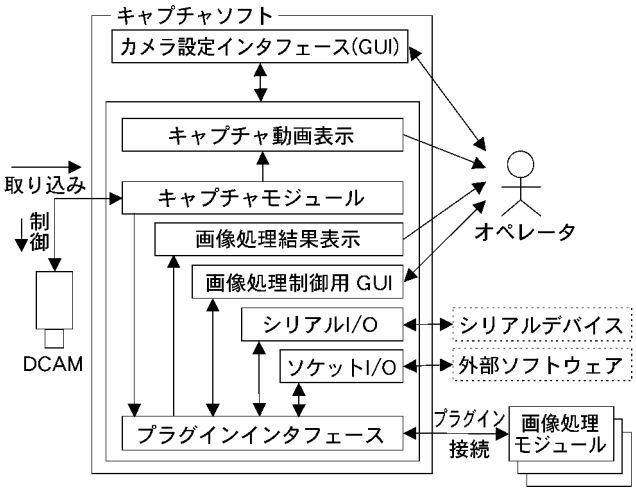

図3　システム構成
Fig. 3　System structure

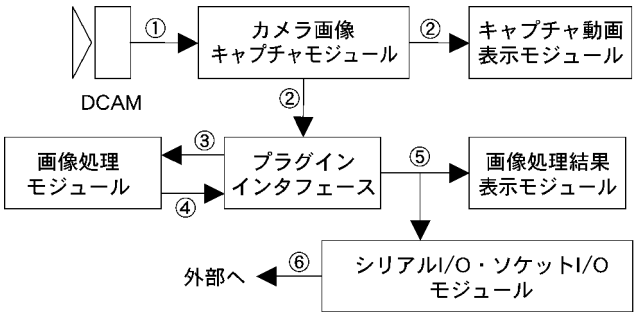

図4　データフロー
Fig. 4　Data flow

⑤ 処理結果の画像は画像処理結果表示モジュールへ渡され結果画像を表示する。一方で，ソケットI/Oモジュールもしくはシリアル I/O モジュールへ送られる。
⑥ I/O モジュールへ送られた処理結果のデータは外部へ送信される。

4．DCAM の制御

DCAM が持つ最も基本的な設定は次の3つである。

- ビデオモード（画像サイズとカラーコーディング）
- フレームレート
- バスの転送速度

ビデオモードに関して，基本的には画像サイズとカラーコーディングが関連付けられているが，format_7 に設定した場合はこの限りではない。このため，format_7 選択時には画像サイズとカラーコーディングを設定する必要がある。設定しない場合は標準の設定か，前回の設定が用いられる。

フレームレートは使用するバスの帯域によって最高速度が決まる。設定可能なバスの帯域は DCAM が持つIEEE-1394コネクタの種類により異なり，1394a コネクタであれば400Mbps 以下，1394b コネクタであれば3.2Gbps 以下に設定できる。ただし，1394b に関して，現状では1600Mbps 以上の帯域を持つ製品は出ていない。

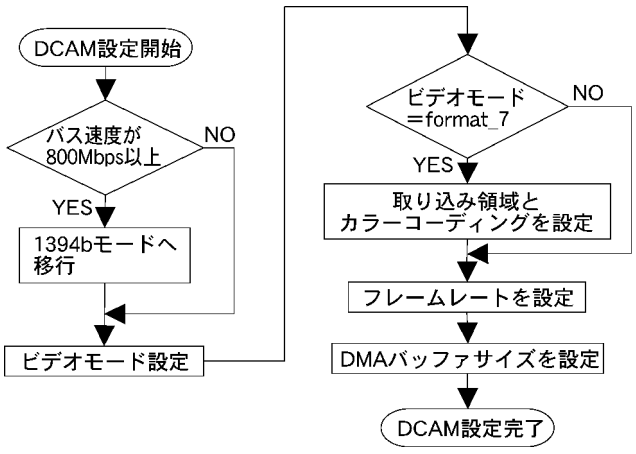

図5　DCAM 設定の流れ
Fig. 5　Flow of DCAM settings

図5に DCAM を制御しバスの転送速度，フレームレート，ビデオモードを設定するために必要な手順を示す。バスの転送速度を800Mbps 以上に設定した場合1394a の範囲では対応できないため1394b モードへ移行する。

ビデオモードが format_7 に対応するものであれば取り込み位置とサイズそしてカラーコーディングを設定する。ビデオモードが format_7 以外であれば対応した設定が存在するため特に設定する必要はない。

フレームレートはカメラの持つビデオモードに対応した項目の中から選択する。ただし，バスの転送速度に応じて設定できるビデオモードやフレームレートなどが制限される。このため設定の組み合わせによってはキャプチャを開始できない。

DMA バッファサイズの設定は DCAM の設定ではなく PC 側の設定である。DMA バッファはカメラから取り込まれた画像の一時保管場所であり，プログラムはこのバッファから画像を取得する。このバッファはリングバッファになっており，通常10フレーム程度の大きさに設定する。

5．GUI による DCAM の制御

5.1　基本インタフェース

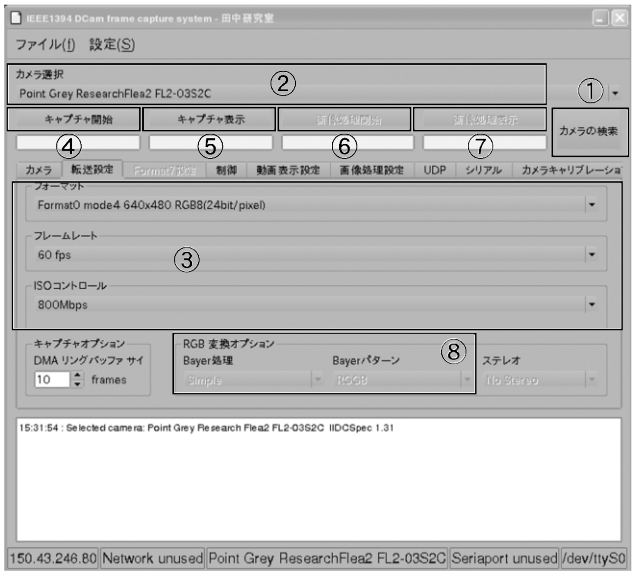

図 6　基本インタフェース
Fig. 6　Basic interface

本研究で開発した環境の基本インタフェース画面を図 6 に示す。

ユーザは本環境を起動した後，[カメラの検索]ボタン(図 6 -①) を押すことによって画面上部のドロップダウンリストボックス (図 6 -②) へ1394バス上に接続されているカメラの一覧を格納する。カメラが 1 台以上見つかった場合，本環境は最初に見つかったカメラを自動的に選択する。

カメラが選択されればそのカメラを設定することが可能になる。画面中央部にある，フォーマット，フレームレート，バスの転送速度を設定する(図 6 -③)。ただし，フォーマットに format_7 を設定した場合フレームレートは 1 フレームのデータ量に依存するため設定できない。また，カラーコーディングは GUI の「Format7設定」タブで設定できる。

これらの設定を行えばカメラから画像を取り込むことが可能になる。ユーザが［キャプチャ開始］ボタン (図 6 -④) を押下げることによって指定された設定で画像を取り込む。この際，図 5 に示した DCAM の設定が行われる。

5.2　キャプチャ中の操作

取り込みを開始することで，[キャプチャ開始]ボタン(図 6 -④) の右側に並ぶ 3 つのボタンが利用可能になる。一つ目は［キャプチャ表示］ボタンで（図 6 -⑤)，取り込んだ画像を表示することができる。二つ目は［画像処理開始］ボタンで（図 6 -⑥)，読み込まれたプラグインのうち選択されたアルゴリズムを適用する。プラグインについては後述する。 3 つ目は［画像処理表示］ボタンで（図 6 -⑦)，おもに画像処理プログラムのデバッグに使用する。この機能は画像処理プログラムから指定された領域へ画像を格納することにより処理結果を表示し，確認するためのものである。

また，format_7 においてカラーコーディングを RAW に設定した場合，カメラから送られてくるベイヤーパターンとそのデモザイキングアルゴリズムを指定しなければならない。この設定はキャプチャ中であるかどうかにかかわらず［RGB 変換オプション］で設定できる（図 6 -⑧)。

6．画像処理プラグイン

6.1　プラグインの利点

画像処理プログラムはキャプチャソフトからプラグインプログラムとして接続される。プラグイン方式にする利点は，キャプチャソフトとビルド単位が別になるため，キャプチャソフト側のプログラムを気にしなくてよいということ。もう一つはキャプチャソフト側にプログラム的なインタフェースを準備していれば画像処理側で DCAM の詳細を考える必要がなくなるということである。これらの利点は，画像処理プログラムを簡潔にし，最小化できるという副次的な効果も生む。このため本環境を前提とすることによって画像処理プログラムは可読性が高まり，資産として引き継ぎ易くなる。

6.2　画像処理プラグインプログラムの作成

本環境に読み込まれるプラグインプログラムはC言語での作成を前提としている。プラグインプログラムは関数の集合で構成されており，その中に“GetPluginName”という名前のプラグイン名取得関数と“PictureProc”という名前の画像処理関数を備えていれば本環境に接続することができる。プラグイン名取得関数はプラグインロード時にキャプチャソフトから呼ばれる。この関数はプラグインプログラムの名前へのポインタを返すのでキャプチャソフトで GUI に表示されるプラグイン名として利用する。画像処理関数は［画像処理開始］ボタンを押すことによって，キャプチャソフトが画像を取得するごとに呼び出される。

リスト 1 にコンパイルして実際に動作させることのでき

```
#include "plugin_util.h"
char *GetPluginName( void )
{return "Plug-in A";}                              (1)
void PictureProc( uint8_t *img,
                  uint16_t w, uint16_t h,
                  uint8_t *view ) {
  int x, y;  uint16_t step;  step = w * 3;
  for ( y = 0; y < h; y++ ) {
    for(x=0;x<w;x++) {
      view[(y*step)+((x*3)+0)] =                   (2)
        255-img[(y*step)+((x*3)+0)];
      view[(y*step)+((x*3)+1)] =
        255-img[(y*step)+((x*3)+1)];
      view[(y*step)+((x*3)+2)] =
        255-img[(y*step)+((x*3)+2)]; } } }
```

リスト１　プラグインプログラム
List. 1　Plug-in program

る画像処理プラグインの作成例を示す。先頭でインクルードされるファイルにはキャプチャソフトを通して利用するネットワーク機能や閾値などを設定するGUIを作成するための関数が宣言されている。

リスト１(1)はプラグイン名取得関数，リスト１(2)は画像処理関数である。画像処理関数の引数には，

・img：キャプチャした画像データへのポインタ
・w, h：幅，高さ
・view：処理結果の画像を格納するメモリ領域へのポインタ

が渡される。このプラグインプログラムはコンパイル時にgccへ -nostartfilesオプションと -sharedオプションを付加することによりプラグインとして動的リンクのオブジェクトとして作成する。

コンパイル例：

```
$ gcc -o 出力ファイル名.so ￥
> -nostartfiles -shared ソースファイル名.c
```

リスト１のプラグインプログラムはキャプチャソフトのGUIに「Plug-in A」と表示される。画像処理を動作させた際には，引数imgで取得した画像の色を反転し，引数viewの指すメモリ領域へ返す。

6.3　画像処理プラグインの読み込み

キャプチャソフトがプラグインを読み込む際には，指定されたディレクトリに対してプラグインを検索する。このときファイル名が‘.so’の拡張子を持つファイルの読み込みを試み，関数(1)と(2)が存在すればプラグインとしてロードされる。

プラグインの読み込みはキャプチャソフトからGUIを操作して行う。このGUIの外観を図７に示す。

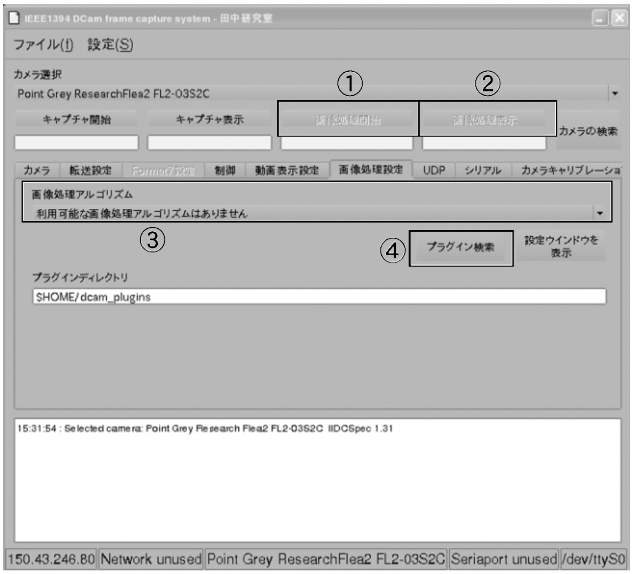

図７　GUIによるプラグインの読み込み
Fig. 7　Loading of plug-ins by GUI

図７において，画面中央のボタン[プラグイン検索]（図７-④)を押すことにより，その下にあるテキストボックスに指定されたディレクトリを検索する。プラグインが一つ以上見つかれば［画像処理アルゴリズム]ドロップダウンリストボックスへ一覧が格納される（図７-③)。

このとき最初に見つかったプラグインが自動的に選択される。画像処理を読み込むことができれば ボタン[画像処理開始]を押すことで画像処理を開始することができる(図７-①)。

リスト１(2)のプログラム例では結果画像を引数が指すメモリ領域(view)へ格納しているが，この画像はボタン[画像処理表示]を押すことによって表示できる（図７-②)。

７．おわりに

動画像処理を行うためにはカメラの制御をおこなうことが必要になる。しかし，カメラだけでなく装置の制御はOSに特有の処理を行わなければならない上に対象の詳細を知っておく必要があり煩雑である。一般的にこれらの作業は日常的に行うようなものではなく，必要になった場合に行うことが多い。このため画像処理を行おうとするプログラマはカメラの制御プログラムを一から作ることが多くなる。本環境はカメラの制御を肩代わりすることで画像処理プログラマが行わねばならなかった煩雑で本質的でない作業をなくすことができた。

DCAMは現在多くの製品が市販されており，今後もさらに多くのものが登場することが予想される。ここで開発した環境の検証はロボカップサッカー小型リーグを対象に行ったが，小型リーグ以外にも汎用的に使用することがで

きる。

参考文献

1）吉田位史，田中卓史：IEEE1394カメラを利用した画像処理開発環境，電子情報通信学会九州支部学生会講演論文集，D-49，2008.
2）1394 Trade Association: IIDC 1394-based Digital Camera Specification Ver.1.31, http://www.1394ta.org/, 2006.
3）A. Bombe, D. Dennedy: libraw1394, http://www.linux1394.org/, 2007
4）D. Douxchamps: Coriander2.0.0 rc6, http://Damien.douxchamps.net/ieee1394/coriander, 2007.
5）D. Douxchamps: libdc1394, http://Damien.douxchamps.net/ieee1394/libdc1394/, 2008
6）Point Grey Research: Flea2, http://www.ptgrey.com/products/flea2/index.asp, 2008
7）Qt Software: Qt4.3.0, http://trolltech.com/about-jp, 2008