組み込み用 Linux 向け動画像処理基盤ソフトウェア「MAlib」

Video Image Processing Software Library for Embedded Linux, “MAlib”

飯尾　淳 [1)]　谷田部 智之 [2)]　比屋根 一雄 [3)]
Jun IIO　Tomoyuki YATABE　Kazuo HIYANE

1) 株式会社三菱総合研究所 情報技術研究部（〒100-8141 東京都千代田区大手町二丁目３番６号
E-mail: iiojun@mri.co.jp ）

2) 株式会社三菱総合研究所 情報技術研究部（〒100-8141 東京都千代田区大手町二丁目３番６号
E-mail: tyatabe@mri.co.jp ）

3) 株式会社三菱総合研究所 情報技術研究部（〒100-8141 東京都千代田区大手町二丁目３番６号
E-mail: hiya@mri.co.jp ）

ABSTRACT. Recently the processing ability of CPU in the personal computer is highly developed, and the digital video/still camera became very popular devices. In addition, because of the interfaces between PC and camera such as IEEE1394 or USB, which are commonly equipped to recent PCs, the video image processing became familiar technology. But there did not exist the software libraries which are used to build video image processing applications easily. So we have developed the general purpose video image processing library and released it to the Internet as an open source software project.

1　背景

近年、デジタルビデオカメラやデジタルスチルカメラといった画像関連機器が普及する一方で、PC とのインタフェースである IEEE1394 あるいは USB が標準的に装備されるようになっている。またビデオキャプチャボードも安価で入手できるようになるとともに、PC 自体の性能が高性能化して大量の画像、映像データをデスクトップで簡単に処理できるようになりつつある。映像処理に関するこのような状況を背景としてノンリニア映像編集アプリケーションなどが実用化され、現在では一般向けに提供されるようにもなった。

しかし残念ながら市販の映像処理のアプリケーションのソースコードは公開されておらず、また他のアプリケーションから自由に利用できるライブラリ形式で提供されてもいない。映像データを対象とした画像認識系のライブラリとしては、ビデオキャプチャ装置や画像測定機器などのハードウェアベンダが提供するライブラリは販売されているものの、特定の機器向けであったり、ライセンスに制限があるなど、自由に利用できる形態で提供されているものはほとんど見受けられない。数少ないオープンソースの画像認識ライブラリとしては、インテルの研究所が提供している OpenCV [1] やマイクロソフトの VisionSDK [2] が存在するが、これらはコンピュータビジョン向けの機能セットを提供するものであり、我々の意図する汎用の映像処理とは指向がやや異なっている。

2　目的

前述の背景に基づき、本プロジェクトではより広範な映像処理向けの基盤ソフトウェアを構築することを目的として、映像処理ソフトウェアの C 言語ライブラリを作成した。

以下の章では、ライブラリの設計方針、ライブラリ構成の概要、ライブラリが提供する機能の概要、利用例としてのサンプルアプリケーション、今後の予定を順に述べる。

3　設計方針

（1）対象プラットフォームの選定

まずはじめに映像処理ライブラリを構成するプラットフォームの対象として Linux を選定した理由を述べる。本研究開発の前提となったプロジェクト [3] が Video4Linux を利用したものであったことに加え、我々は Linux の持つスケーラビリティに着目した。

本研究開発の最終ターゲットは組込みシステムであるが、開発自体は i386 アーキテクチャの通常の PC で動作する Linux で実施した。しかし Linux は PC だけでなく、組込み向けにも様々な実装が用意されている。Linux で開発する利点として、PC 上で開発されたソフトウェアを少ない労力で組み込み向けに移植できる点を挙げることができる。後に述べるとおり、現在、組み込み Linux の一例として、Compaq の iPAQ で動作する ARM アーキテクチャ向けの Linux を対象に、本研究成果の移植と応用アプリケーションの作成を検討しているところである。

（2）C 言語によるオブジェクト指向設計

汎用性を高めることを考慮し本ライブラリは C 言語で実装した。ただしライブラリの保守性と拡張性を高めるために、オブジェクト指向設計を採用し、すべてのモジュールはクラスとして実現している。なお C 言語にはオブジェクト指向の実装メカニズムは用意されていないので、構造体を利用してクラスを構成することにより、擬似的なオブジェクトシステムを実現している。このオブジェクトシステムは、GTK+ [4] で用意されているオブジェクトシステムを簡易化したものである。

クラスの継承は、構造体を拡張することで実現される。また public / private といったアクセス制限は、それぞれのクラスを独立したファイル単位で記述し、ファイルスコープを利用することにより実現した。またそれぞれのク

ラスは関数ポインタを格納するバーチャル関数テーブルを独自に持ち、いくつかのメソッドに関してはバーチャル関数としてアクセスすることが可能である。

本ライブラリの利用者は、ライブラリに既に用意されている機能をそのまま利用するだけでなく、用意されている構造体をラップする構造体を定義することで、既存の機能を継承した独自のクラスを作成して利用することができる。本ライブラリで提供するオブジェクトシステムは、できるだけ少ない労力で新たなクラスを作成することができる点に留意して設計した。

4 モジュール構成

(1) クラス階層

本ライブラリが提供する (提供予定を含む。(*) 印がついているクラスは現在のところ未実装である) クラスの構成を以下に示す。

```
MalibObject
  +- MalibHolder
  |    +- MalibGtkDisplay
  |    +- MalibFileOutput (*)
  |    +- MalibNetworkOutput (*)
  |    +- ...
  |    +- MalibBuffer
  |         +- MalibPlainBuf
  |         +- MalibRingBuf
  |         +- MalibLineBuf
  +- MalibFrame
  |    +- MalibFrameAV (*)
  +- MalibSource
       +- MalibBttv
       +- MalibMpegFile
       +- ...
       +- MalibFilter
            +- MalibDelay
            +- MalibFrameDiff
            +- MalibGenericFilter
            +- MalibGrey2Bw
            +- MalibNegative
            +- MalibRgb2Grey
            +- MalibSepia
            +- MalibMovingAve
            +- MalibSpatial3x3
            +- ...
            +- MalibMerger
                 +- MalibOverlap
                 +- ...
```

`MalibObject` は全てのルートクラスである。本クラスでは、オブジェクトに対する基本的な機能を提供する。具体的には後述する参照カウントを利用したオブジェクトの自動消去のメカニズムや参照カウンタそのものが、ルートクラスに用意されている。

(2) 処理プロセスのリンク構造

本ライブラリの各クラスを組み合わせて構成する画像処理のプロセスは、`MalibSource`、`MalibBuffer`、`MalibFilter`、`MalibHolder` の組合わせによる構成が基本となる (以下では順に Source、Buffer、Filter、Holder とする)。これらは全て抽象クラスである。実際の処理手順を構成する場合には、それらのサブクラスを構築して接続するが、とくに必要な場合以外はこれらの抽象クラスのオブジェクトとして取り扱うことで抽象度の高い記述を行なうことができ、プログラムの可読性を高めることができる。

処理のプロセスは、画像データを生成する Source がデータを Buffer に書込み、さらに後段に接続された Filter がそのデータに基づきフィルタリング処理を行なう、との手順で進められる (図 1)。処理のプロセスの記述は、Source → Buffer → Filter → Buffer → Filter → ⋯ → Holder とのように、必要なだけ Filter を繰返し適用することで行なう。なお Buffer は Source を入力にとり、Filter は Holder に対して処理データを出力する。Filter は Source のサブクラスであり、Buffer は Holder のサブクラスであるため、このルールをそのまま適用することができる。

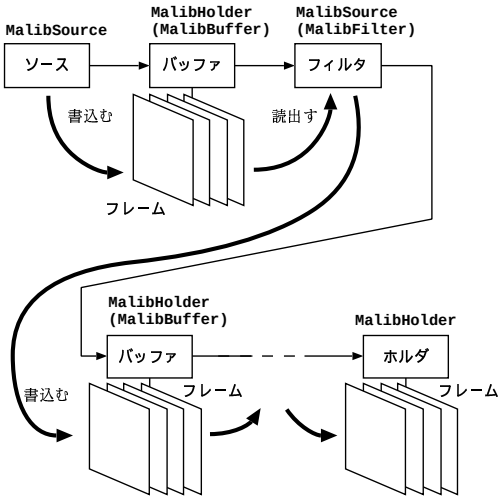


図 1: 動画像処理の流れ

本ライブラリを利用した映像処理アプリケーションは、まず初期設定として処理オブジェクトを生成し、各オブジェクトを処理手順に従って接続する。全てのオブジェクトを接続した後で最後段のオブジェクトに対してデータ更新のメソッド呼び出しを行なうことにより、再帰的にデータの更新が行なわれる。各フレームに対する処理を繰返し行なうことで、動画像に対する処理を連続的に実現することが可能となっている。

5 MAlib が持つ機能の概要

前述のとおり、個々のフィルタを適用した動画像処理が MAlib の主要な機能であるが、MAlib をより利用しやすいライブラリとするためにライブラリ全体にわたるいくつかの機能を用意している。ここではその機能について、説明を加える。

(1) 参照カウントの利用

処理手順のオブジェクトはそれぞれ前段のオブジェクトを参照することによってリンク構造を成すことを説明した。ところで処理オブジェクトのリンクは、一般には直列とはならず複雑な有向グラフを構成する (図 2)。図 2 は、公開しているソースファイル一式に附属するサンプルアプリケーションを構成する処理オブジェクトのリンク構造を表したものである。

このように複雑なリンク構造を成すデータ構造を適切に取り扱うためには、参照カウンタを導入して参照の回数を数えておく必要が生じることがある。本ライブラリでは、以下に挙げる機能の実現のために参照カウントを用意し、それぞれの動作を実現した。

- メモリ管理 (オブジェクトの自動消去)
- フレームデータ更新における同期の確保
- サンプルフレーム情報の共有管理

a) メモリ管理

まず不要となったオブジェクトの自動消去の機構を導入することで、ユーザのメモリ管理の負担を軽くした。具体的には、あるオブジェクトを消去しようとした場合、関連するオブジェクトも自動的に消去する手段を提供した。ユーザは全てのオブジェクトを管理する必要が無く、実際

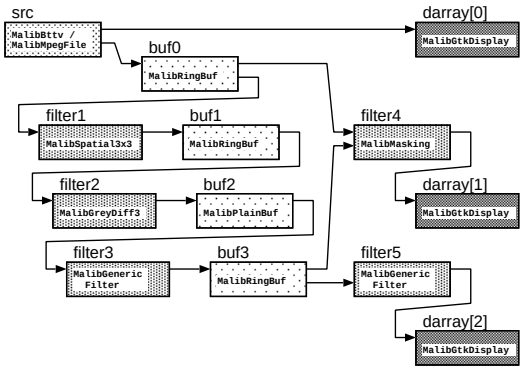


図 2: 処理オブジェクトの成すリンク構造

には処理結果の出力を行なうオブジェクトのみを管理しておけばよい。

後段のオブジェクトを消去しようとした場合、そのオブジェクトの参照関係を辿って関連するオブジェクトの消去も試みる。ただし複数のオブジェクトがひとつのオブジェクトを参照しているケースが考えうるので、そのような場合に不用意にオブジェクトを消去してしまわないために、参照カウンタを用いた。

b）同期の確保

次にフレームデータの更新における同期の確保に対する参照カウントの利用について解説する。

先ほど例に挙げた図 2 の処理手順では、3 つの出力が存在する。データの更新処理は各出力から再帰的に呼び出されることになるが、単純にリンクを遡ることとすると、入力となる Source オブジェクトでは複数回の呼び出しが行なわれることになる。

毎回同一のデータを生成する Source であれば問題ないが、一般には本ライブラリの対象は映像ソースであり、一回の呼び出し毎に次のフレームのデータが生成される。したがって単純に参照を辿って処理を行なったのでは、全体として呼び出しのパスによって対象のフレームが異なる可能性があり、都合が悪い。とくに、複数の処理を経たデータを比較するような手順を構成した場合、その問題は致命的なものとなる。

そこで本ライブラリでは、全ての処理手続きにおいてフレームの同期を確保するために参照カウンタを利用した。静的なカウンタに加え、一時的に増減を繰り返す参照カウンタを導入することにより、全ての参照先から呼び出しが行なわれるまではキャッシュされている処理データを返すようにした。この機構によって、一連の処理手続きのリンク構造においては、全ての処理におけるフレームデータの同期の確保が実現されている。

c）フレーム情報の共有

また各 Source (Filter) は、そのオブジェクトがどのような形式のフレームを出力するかといった情報を持つ (図 3)。例えば RGB カラーデータを白黒濃淡画像に変換するフィルタの場合、出力の画像サイズや画像データの種別 (この場合は白黒濃淡画像) などの情報を格納したサンプルフレーム情報を用意する。

画像情報が変化しないフィルタの場合は、前段の情報を共有することが適切と考えられる (図 4)。そこでこのデータの共有により複数からの参照関係が生じるため、オブジェクト同士の接続と切断、あるいは消去に関連して参照カウンタの利用が必須となる。

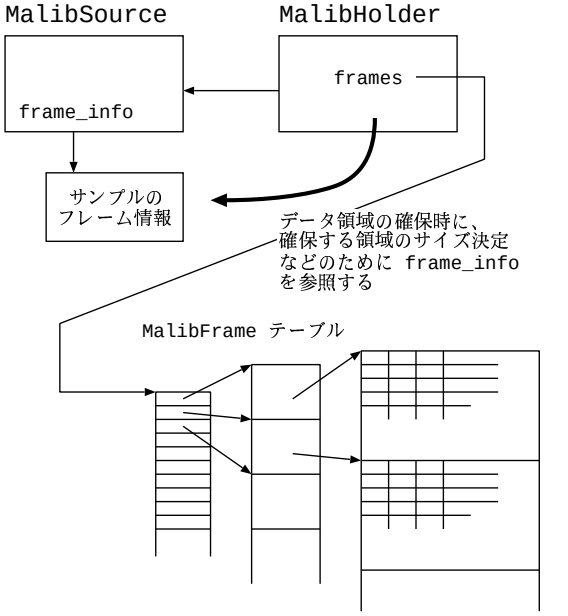
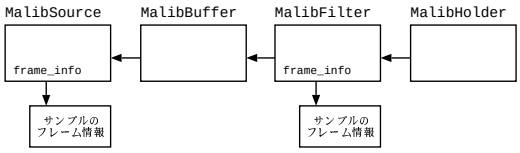


図 3: 生成フレーム情報の保持

フレーム形式が変化する場合



フレーム形式に変化が無い場合

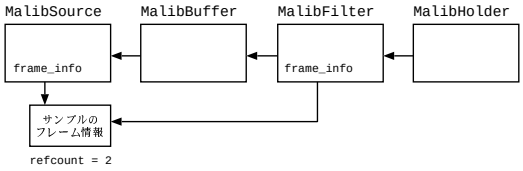


図 4: 生成フレーム情報の共有

6 利用例

本ライブラリの利用例として、道路を通行する車両を認識し、追跡を行なうアプリケーションを構築した。本アプリケーションは、サンプルアプリケーションとして公開ソースファイルの一部に含まれている。

（1）移動物体の追跡アプリケーション

本サンプルアプリケーション objtraq は、白黒濃淡画像の動画像を入力対とする。画像中の移動物体を認識し、追跡した結果を表示する。追跡した結果は同一物体であることを示す色で識別され、追跡している位置とともに別ウィンドウに表示される。

入力する動画像 (図 5)、移動物体の認識結果 (図 6) および追跡結果 (図 7) のスナップショットを順に示す。この例では 3 台の車両が映像中に出現しているが、右上のトラックは駐車車両である。移動物体として認識されるオブジェクトは、画面中央を左から右へ向かって進行している小型車、および画下部を右から左に進んでいるタクシーの 2 点

である。画面上部に写っている歩行者は認識対象としないよう、しきい値を設定している。

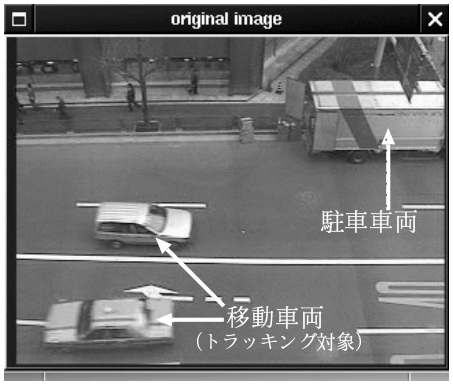


図 5: 入力映像

図 6: 移動物体の認識結果

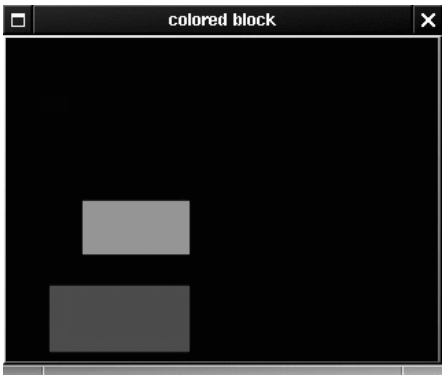


図 7: 移動物体の追跡結果

なお実際のアプリケーションの実行においては、これらの認識はリアルタイムに処理が行なわれ、連続して認識結果が画面上に示される。

7 組み込み Linux への適用

現在、テーマとして掲げた組み込み Linux への応用の実現に向けて、ライブラリの移植作業を行なっているところである。実験環境で対象としている組み込み Linux は、Linux familiar v0.5 pre-release (カーネル 2.4.7)、ハードウェアプラットフォームは 64MB の RAM を搭載した iPAQ H3660 である。また開発環境には、式神 SDK 1.0 α版を流用した。

MAlib を利用したアプリケーションが iPAQ 上で動作している様子を図 8 に示す。図で表示しているアプリケーションは開発中のものであるが、外部に接続されたネットワークカメラから取得した画像を無線 LAN で受信し、表示しているところである。

図 8: iPAQ で動作する MAlib アプリケーション

8 まとめ

本プロジェクトでは、組み込み用 Linux 向け汎用動画像処理ソフトウェアライブラリ MAlib を開発した。MAlib の開発にあたり、C 言語を用いたオブジェクト指向設計によって汎用性と保守性を両立させることに留意した。またその利用例として、移動物体認識アプリケーションが簡単に実装できることを示すとともに、組み込み Linux での適用状況を報告した。

MAlib は現在、次の URL で公開している。開発に参加してくれる人物やユーザを広く募集しているので、興味のある方はぜひとも上記 URL を参照していただきたいと考えている。

`http://www.malib.net/`

9 参加企業及び機関

本プロジェクトは平成 12 年度未踏ソフトウェア創造事業において、高田広章プロダクトマネージャのもと本論文の筆者が開発メンバとなり遂行された。プロジェクト管理会社は株式会社アーク情報システムが担当した。

なお本プロジェクトの契約件名は、「組み込み用 Linux 向け動画像処理基盤ソフトの開発」である。

10 参考文献

[1] Davis, J. and Bradski, G：“Real-time Motion Template Gradients using Intel OpenCV”, IEEE ICCV’99 Frame-Rate Workshop, 1999. Vol.1,NO1, p6〜10(1996)

[2] The Vision Technorogy Group (Microsoft Research)： “The Microsoft Vision SDK”, `http://research.microsoft.com/projects/VisSDK/`, May 2000.

[3] 飯尾淳：“Linux による画像処理プログラミング”, オーム社, 2000.

[4] Tony Gale and Ian Main： “GTK+ 1.2 Tutorial”, `http://www.gtk.org/tutorial/`, Mar 2001