2003(平成 15)年度　機械システム工学科　卒業論文要旨

アクティブロボットビジョンの開発における フレームワークの作成と評価

001120　坂本昌彦

1.　緒言

2004 年現在，アクティブロボットビジョン(ARV)システムをはじめとする画像処理・認識システムは既に実用化のフェーズに入っているが，比較して，それらシステムの設計から開発に至る一連のプロセスまでもが進歩を遂げたとは言えない．

システムの設計・開発をトータルに眺めるにはプログラミングやアルゴリズムの知識だけでは不十分であり，ハードウェアからオペレーティングシステム(OS)にいたるまで幅広い知識と経験をトータルに眺めなければ，優れたシステム設計と開発プロセスを実現することはできない．

最先端技術を研究する大学という場では，これまでそうしたシステム設計・開発プロセスの研究には光が当てられてこなかった．その場限りのコーディングを繰り返したプログラムは，これから研究を志す者にとっては負の遺産となる可能性がある．

だからこそ，システム設計・開発プロセスを見直し，柔軟性と拡張性に富んだ再利用性のあるモジュールとフレームワークを作成する必要性がある．

本研究ではアクティブロボットビジョンシステムを対象としたプログラミングフレームワークを提案し，実際にシステムを構築し，その効果を検証する．

2.　システムの構成

2.1　システムの分析とモジュール化

フレームワークを提案するために，まず ARV システムに必須の機能を抽出し，モジュール化を試みる．

本研究では図 2-1 に示すように三つの機能にモジュール化できると考えた．

(a)画像入力機器から画像データを取得する．
(b)画像データを処理し，意味のある値を計算する（例えばボールの位置）．
(c)計算された値を用いて何らかのアクションを実行する．

本研究ではこれらの機能を独立したプログラムとして実装することにより，モジュール化を試みる．

独立したプログラム間でのデータのやりとりには，図 2-2 で示すように共有メモリを使用する．

注意点として，メモリへの読み書きが同時に行われないように排他ロック機能(Semaphore)

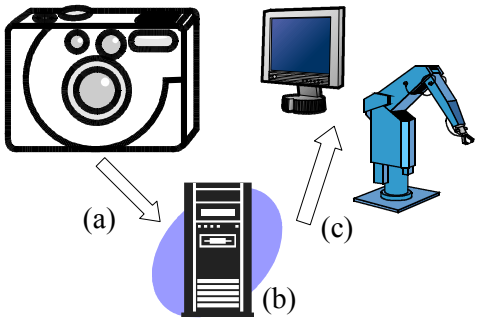

図 2-1 ARV システムのモジュール化

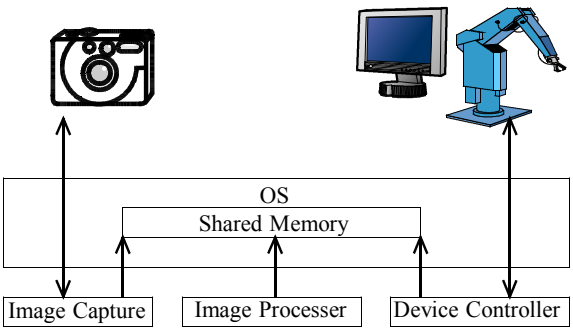

図 2-2 フレームワークの概要

を使用する必要がある．

2.2 OSの選択

以上述べたようなフレームワークを実装するための OS の選択を行う．本研究では以下の条件に見合った OS として Linux を採用した．

・安定したマルチタスク管理
・安定した共有メモリ・セマフォ管理
・C 言語の開発環境が標準で含まれている．

使用する開発言語には，一般的な C 言語を用いる．

2.3 フレームワークの提案

前述のモジュール化を実現するためのライブラリ群を作成することにより，以て本研究の提案するフレームワークとする．

本研究では次の三つのコンポーネントによるフレームワークを提案する．

・GSMTD(汎用共有メモリ透過デーモン)
・libgsmtdclient(GSMTD 専用通信ライブラリ)
・libgsmtdarv(ARV 向け共有メモリ・セマフォアクセス用ライブラリ)

これらのコンポーネントの関係を図 2-3 に示す．また，それぞれのコンポーネントについて説明する．

2.3.1 GSMTD(Generic Shared Memory Transparent Daemon)

システムを構成するプログラム群は共有メモリやセマフォにアクセスする．その際，それらシステムリソースに OS が割り当てた識別番号を知っている必要がある．

GSMTD を利用するプログラム群は GSMTD に対して TCP/IP 接続による通信を行い，それらの情報を取得する．

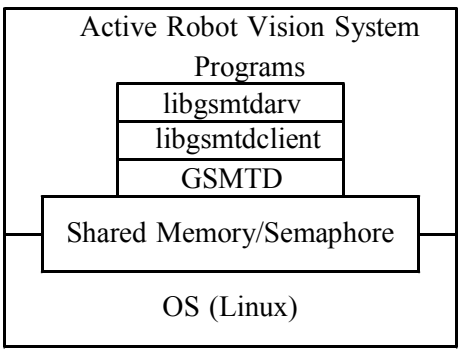

図 2-3 コンポーネント階層図

2.3.2 libgsmtdclient

TCP/IP 接続による定型的な通信手順をまとめたライブラリ．これを用いることにより，TCP/IP 接続に関する知識が無くとも GSMTD と通信し，システムリソースに関する情報を取得できるようになる．

2.3.3 libgsmtdarv

libgsmtdclient は単に GSMTD との通信を簡略化するだけにすぎない．取得した識別番号を用いて共有メモリやセマフォにアクセスするには煩雑な手続きが必要である．また，ARV システムでは必ず画像データを保存する共有メモリが必要であり，そのためのアクセスも定型的かつ共通である．

そうした，ARV 向けに特化した GSMTD 通信ライブラリが libgsmtdarv ライブラリである．

2.4 実験・評価方法

フレームワークの有用性の検証方法として，本研究は次の二点を調べる．

・プログラム実行時のマシンリソースの消費量
・プログラム開発期間

いくらフレームワークによりプログラムの保守や再利用が簡単になろうと，システムの処理速度が低下すれば無意味である．

プログラム開発期間は，フレームワークを用いたマルチタスクから，フレームワークを用いないシングルタスクのプログラムに落とし込むのに要する作業である．計算機の処理能力が期待できない組み込み機器などで，シングルタスクで ARV システムを構築する場合を想定する．

題材としては差分検出による動体検知プログラムと，三菱電機製ロボットアームの MOVEMASTER EX を用いた動体追跡プログラムを作成し，マシンリソースと開発期間を調べる．

3. 画像処理

差分検出による動体検知プログラムでは，入力画像(320x240pixel)を 10x10pixel のセルに分割し，それぞれのピクセル値の平均値を直前と比較し，閾値を超えていれば差分が検出されたものと判定する．

また ARV システムでおそらく必須になるで

あろう，輪郭線検出とその角度検出処理も加えた．輪郭線は対象ピクセルの右と下のピクセルを比較し，どちらかが閾値を超えれば色が急激に変化する輪郭線と判定する．角度検出は10x10pixel のセルごとに，そのセル内の輪郭線検出ピクセルの座標を最小二乗法で直線近似し，その傾きを 8 段階に分ける．

MOVEMASTER を用いる動体追跡プログラムでは，標的座標を近似色の重心位置とする．近似色の判定はセルごとに行い，RGB 成分それぞれのセル毎の平均値を算出し，標的セルの平均値と比較して閾値以内に収まっていればそのセルを近似色と判定し,標的の一部と考える．

4. MOVEMASTER専用シリアル通信デバイスドライバの作成

本研究で用いるロボットアームは三菱電機マイクロロボット MOVEMASTER EX である．5 自由度の垂直多関節型ロボットで，既に生産終了となっており，旧式といえる．

MOVEMASTER の制御は，PC 側からコマンド文字列をシリアル又はパラレルポートを用いてドライバユニットに送信する．本研究では RS232C によるシリアルポートを用いる．

MOVEMASTER は日本電気の PC-9800 シリーズの RS232C シリアル通信に適合する通信方式を採用しており，2004 年現在主流となっている IBM PC/AT 互換機上で動作している OS はデフォルトでは対応していない．

受信データの取りこぼしを防ぐため送信側に対して送信許可を知らせるためのハンドシェイク処理において，現在殆ど使われていない信号線を MOVEMASTER は使用していた．

そのため通信ケーブルを自作し，さらに MOVEMASTER 専用の RS232C 通信ドライバモジュールを作成した．

5. 実験と考察

5.1 諸元

・コンピュータシステム

CPU: Intel Pentium 3 700MHz

Memory: 192MB HDD: 60GB

OS: TuboLinux 8 workstation（kernel-2.4.22）

・カメラ

Creative 社 WEBCAM PRO eX（USB 1.1）

30 万画素 CCD イメージセンサ（320x240）

感度：6 ルクス

ビデオフォーマット：24 ビット RGB

5.2 差分検出による動体検知プログラム

最初にフレームワークを使用した ARV システムを構築する．次にフレームワーク部分を削り取り，シングルタスクに落とし込む．

マルチタスクとシングルタスクとでは，単純に数値のみではパフォーマンスの比較はできない．本研究では XOSView というグラフィカルなパフォーマンスメーターを利用しておおよその比較をする．

図 5-1 がフレームワークを使用したシステムが稼働中のもので，図 5-2 がフレームワーク未使用のシングルタスクが稼働中の XOSView のスクリーンショットである．

フレームワークの使用版と未使用版との間で大きなパフォーマンスの差はみられない．すなわち，本研究が提案するフレームワークを用いてもシステムの処理能力に大きな影響を与えないことが実証された．

図 5-1 フレームワーク使用版

図 5-2 フレームワーク未使用版

フレームワークを使用した ARV システムを，未使用のシングルタスクに落とし込むのに要した作業は実質二日間である．作業するプログラマーの技量に依るところも多いが，フレームワーク部分を削り落とし，一つのプログラムにまとめるだけなので作業内容は簡単である．

逆にフレームワークを考えずに作ったプログラムを，拡張のため分割しモジュール化するための手間を考えれば，本研究が提案するフレームワークを最初から使うことにより工数が大幅に削減できると考えられる．

5.3　移動物体追跡プログラム

本研究のフレームワークの柔軟性を示すデモンストレーションとして，近似色の重心検出による移動物体追跡プログラムを作成した．

・図 5-3 にカメラを装着した MOVEMASTER を示す．

・図 5-4 にプラスドライバーの橙色のグリップを追跡する追跡する様子を示す．

フレームワークを使用したことにより，画像入力・画像処理部分を殆ど変更することなく，MOVEMASTER との通信プログラムと重心位置検出処理を加えるだけで作成することができた．

6.　まとめ

本研究ではアクティブロボットビジョン向けのフレームワークの作成とその有効性の検証を行った．

・Linux 上でフレームワークを実装し，動体検知プログラムを作成することによりフレームワークの有用性を実証することができた．

・MOVEMASTER を使用した追跡プログラムを作成し，本研究の提案するフレームワークの柔軟性と拡張性の高さを実証することができた．

参考文献

(1) 奥脇学，Linux プログラミングガイドブック，(2001)，266，秀和システム

(2) 河野清尊，C 言語による UNIX システムプログラミング入門，(2003)，464，オーム社

(3) 山形孝雄，トランジスタ技術 SPECIAL No.72 特集パソコン周辺インターフェイスの全てⅢ，(2000)，173，CQ 出版社

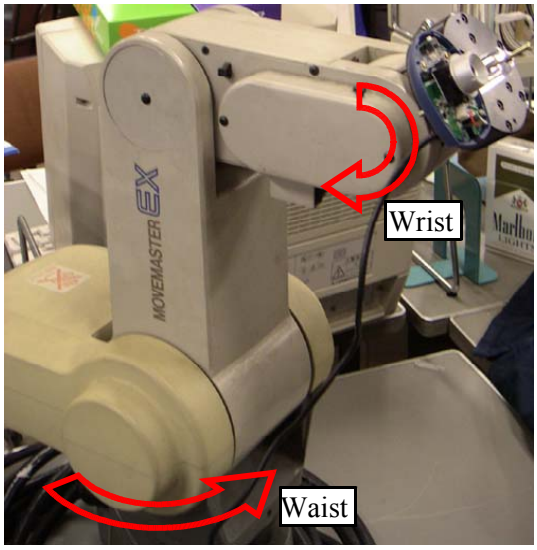

図 5-3 カメラを装着した MOVEMASTER

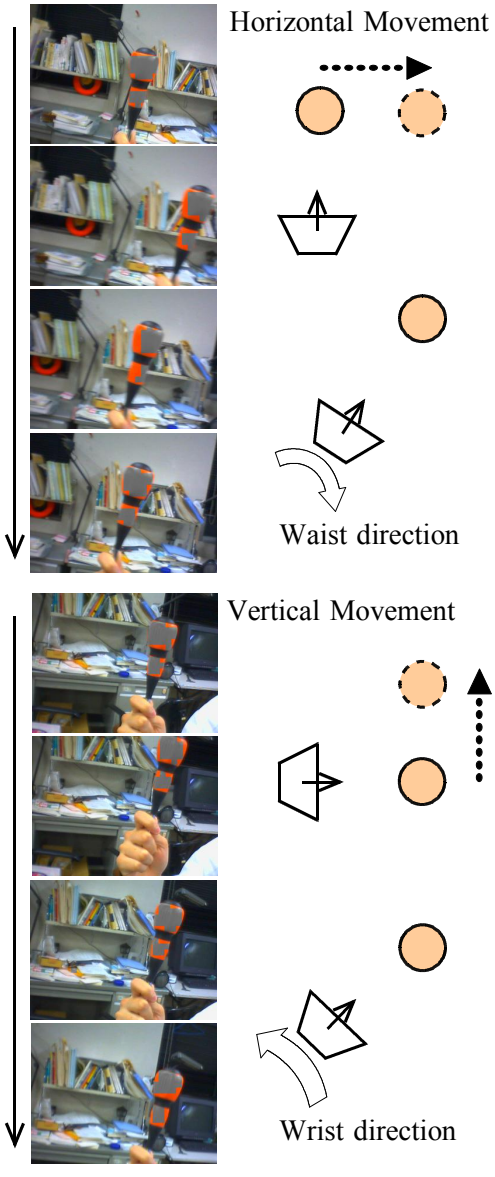

図 5-4 追跡中の連続画像