画像処理ワークステーション VIEW-Station のソフトウェアアーキテクチャ†

佐藤宏明†† 岡崎 洋†† 河合智明††
山本裕之†† 田村秀行††

VIEW-Stationは，画像専用プロセッサの高速処理と汎用ワークステーションの柔軟なソフトウェア環境を両立する画像処理ワークステーションである．VIEW-Stationのソフトウェアシステムは，応用システム開発のための拡張性と多様なハードウェア構成への移植性を実現するように階層的に構成されている．その中核をなすV-Sugarは，画像データ型の導入と画像処理アルゴリズムの関数型記述を特徴とする画像処理用プログラミングツールであり，画像処理プロセッサやウィンドウシステムを利用したプログラムを簡便に記述することができる．これは，プログラミング言語C++上のクラスおよび関数パッケージとして実現されているが，応用プログラマから見た場合一種の画像処理用言語とみなすことができる．画像データ型のメモリ管理は画像メモリの動的管理機構V-Serverにより実現する．V-Serverは，処理実行形態に応じてデータ形式を変換する機能と画像メモリを仮想化する機能により，画像データ操作を一元化しハードウェア固有の情報を隠蔽する．また，画像処理プログラムのためのウィンドウツールVIEW-Windowsは，ワークステーションのマルチウィンドウと画像処理プロセッサに付属する専用ディスプレイに対する操作を一元化するウィンドウマネージャと表示・操作のためのプログラム部品（widget）およびその部品を用いた表示・操作ルーチンから構成され，画像処理に特有なデータを用いたユーザインタフェースの構築に利用される．

1. まえがき

画像専用プロセッサの出現は，信号レベル処理の大幅な高速化をもたらし，様々な分野で画像処理技術の実利用化の道を拓いた．反面，こうしたハードウェアの多様性がソフトウェアの移植性の障害となり，アプリケーション・プログラマの育成を阻む要因となっている．これからの画像処理システムにとっては，

1）応用分野は極めて広いため，ニーズが多様であり，機能／規模を柔軟に変更しうる核システムが望まれている，また，
2）ハード，ソフト共に標準化が進んでいないため，基本部分での重複投資が大きい．応用システム開発を促進するには，基盤となるソフトウェア環境が必要である．

一方，ソフトウェア環境という点では，UNIXオペレーティングシステムをベースとしたエンジニアリングワークステーション（以下，EWSと略す）が優れている．マルチウィンドウを利用したユーザインタフェース構築の容易さも，対話型処理を重視する画像処理システムには好都合である．処理能力においても，旧来の大型計算機・ミニコンピュータを凌駕するパワーを持ち，かつ画像処理用高速プロセッサ（Fast Image Processor；以下FIPという）とのバス接続も容易になりつつある．

こうした背景から，我々はハードウェア処理の高速性とソフトウェア処理の多機能性を両立しうる画像処理ワークステーション **VIEW-Station** (Vision and Image Engineering Work-Station) アーキテクチャを提案し，基幹ソフトウェアから順に開発を進めている．VIEW-Station計画の主眼は，応用プログラム開発を促進するソフトウェア基盤の確立であり，EWS環境下からのFIPの有効利用を目指している．特に，FIPの画像処理機能の違いをソフトウェア的に吸収し，FIPの存在自体をユーザに意識させないプログラミング機構の実現が第一の目標である．また，画像処理に適したプログラミングパラダイムからユーザフレンドリィな画像処理実行環境に至るソフトウェア・アーキテクチャを構築する．

これまで，画像処理分野におけるソフトウェア蓄積としては，SPIDER[1]に代表されるアルゴリズムライブラリが整備されている．一方，画像処理プロセッサの機種に依存しない基本画像演算の導入[2]や，画像処理アルゴリズム記述のための専用言語の開発[3]や，画像アクセスを統一するための標準的な画像アクセス関数の導入[4]など，ソフトウェアの移植性を高めようとする試みも行われている．また，画像処理プログラムの記述を高度化あるいは簡易化するために，関数型の言

† The Software Architecture of Vision and Image Engineering Work-Station by HIROAKI SATO, HIROSHI OKAZAKI, TOMOAKI KAWAI, HIROYUKI YAMAMOTO and HIDEYUKI TAMURA (Canon Inc.).

†† キヤノン(株)

語[6]や，画像データ型の導入[6]が検討されている．一般のEWSのソフトウェア環境では，ウィンドウシステムを利用した応用プログラム開発をより効率化するためにX Toolkit[7]などのツールの開発が進められている．また，このような状況を踏まえ，次世代画像処理ソフトウェアシステムの調査[8]や，画像処理の標準化の活動[9]も行われている．VIEW-Stationソフトウェアは，こうした動向を反映し，実用的なプログラミングの枠組を提供する．

本論文では，まず，2章で本システムの対象とするユーザレベルとハードウェアおよびソフトウェアアーキテクチャに関して述べる．続いて，画像処理プログラミングツールV-Sugarと画像処理ウィンドウシステムVIEW-Windowsを中心に，ハードウェアに独立なプログラミング環境の実現について述べる．

2. VIEW-Stationの位置付けと基本アーキテクチャ

2.1 ユーザレベル

画像処理技術の応用範囲は極めて広範であり，そのユーザレベルも様々であるため，画像処理システムの形態も多岐にわたる．VIEW-Stationの目指すところを明らかにするため，まず我々は，ユーザの目的や技量，システムの利用形態を**表1**のように分類・整理し，各々の支援環境を検討した．ここで，ユーザレベルの差は，画像処理技術に対する知識や意識の違いによるものであり，プログラミング能力・嗜好による分類ではない．すなわち，レベル0ユーザとは，画像処理自体を研究対象として，新規アルゴリズムを開発するユーザである．一方，レベル1ユーザとは，既存の処理モジュール群を駆使して応用システムを構築するシステムエンジニア／プログラマである．また，レベル2ユーザとは，いわゆるターンキーシステムのオペレータである．

通常，レベル2ユーザのためのシステムは特定目的の専用機となるため，その機能／規模の範囲は対象分野の機能／コスト要求に応じて大きく変動する．一方，レベル0およびレベル1ユーザのためのシステムは，アルゴリズムおよびシステム／プログラム開発を目的とする研究開発用システムであり，同一の基本アーキテクチャ上に構築可能である．我々のVIEW-Stationは，レベル0およびレベル1ユーザを対象とし，レベル2ユーザの環境は考えない．

表1 ユーザレベルと支援環境
Table 1 User levels and required supports.

ユーザレベル	レベル0	レベル1	レベル2
対象ユーザ	画像処理技術者	応用システム・機器開発者	応用システムのオペレータ
使用目的	新規アルゴリズム開発等	応用アルゴリズム・システム開発	特定アプリケーションの実行
利用形態	画像処理モジュールのプログラミング	モジュールの組合せ等によるプログラミング	パラメータの設定等
支援環境	基本プログラミングツール	簡易プログラミングツール，実験環境	専用インタフェース

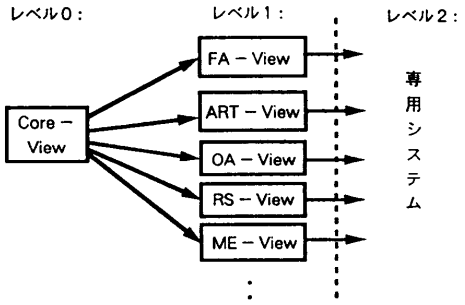


図1 VIEW-Station構想
Fig. 1 System levels of VIEW-Station.

ここで，VIEW-Stationとは，ユーザレベル・使用目的に応じて設計する複数のシステムの総称である．応用分野に依存しないレベル0ユーザ向きの核マシンを**Core-View**と呼ぶ．これを母体とし，OA, FAや医療(ME)等の応用分野に応じた拡張変更を加えることで，レベル1ユーザ向き応用システムが構築される(**図1**)．レベル1システムは，レベル2実用マシンの応用ソフト開発システムと位置付けることもできる．

2.2 基本ハードウェア構成

Core-View基本アーキテクチャとして，汎用EWSにバス結合した画像処理ハードウェアの構成を採用する．FIPのハードウェアは，画像処理用のメモリ(Image Memory；以下I-Memと呼ぶ)と画像表示用のメモリ(Display Memory；以下D-Memと呼ぶ)および必要に応じてプロセッサ・モジュールの追加が可能な形式の画像プロセッサ群(IPU)から構成されるものとする(**図2**)．

FIPをEWSの外付けのプロセッサとしたのは，なるべく多種類のEWSおよびFIPに対応できる体系を目指したからである．ソフトウェア的に観て，この構成と同等であれば，両者を一体化した統合アーキテ

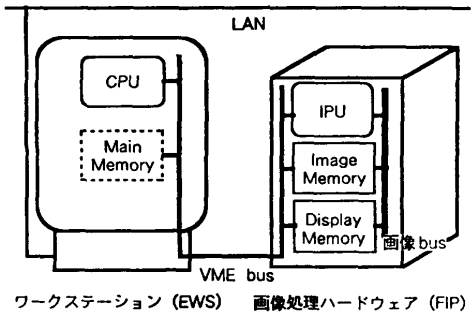


図 2 基本ハードウェア構成
Fig. 2 Core-View basic architecture.

クチャや十分高速な EWS 単体であっても構わない.

EWS と FIP が外部バスで結合されている場合，画像データの転送に伴うオーバヘッドを最小にしたい. VIEW-Station では，画像処理の対象となるあらゆる画像データはすべて I-Mem にあるものとし，以下の仕様を満足するものとした.

i) I-Mem は EWS の CPU のアドレス空間にマッピングされ，CPU から直接アクセス可能とする.

ii) I-Mem 上でのデータの配置は CPU プログラムが取り扱う2次元配列の順序（以下，連続配置と呼ぶ：**図 3**）とする.

iii) I-Mem の基本データは1画素当たり 16 ビット長を持つ.

画像メモリ I-Mem を一元化することにより，EWS の CPU によるソフトウェア処理と FIP の IPU によるハードウェア処理を融合することができる. ただし，この仕様では，通常の IPU が採用している画素の配置（固定配置）ではなく，連続配置に対し直接処理を実行できる（すなわち，連続配置アドレシング可能な）画像プロセッサが必要となる.

現時点では，EWS は UNIX マシン，バスは VME バスを採用する. I-Mem と D-Mem を別に設けることを基本とするが，EWS 側から見てソフトウェア的に矛盾がなければ，両者を共用しても構わない. 一方，システム的に可能であれば，D-Mem をウィンドウシステムの管理下に置き，EWS 標準の CRT に画像を表示することも許している.

2.3 ソフトウェアアーキテクチャ

VIEW-Station のソフトウェアシステムは，これまで画像処理ソフトウェアの開発効率を低下させていた FIP ハードウェア依存の問題を解決するとともに，最新の EWS ソフトウェア環境を十分活用する構造を採用している.

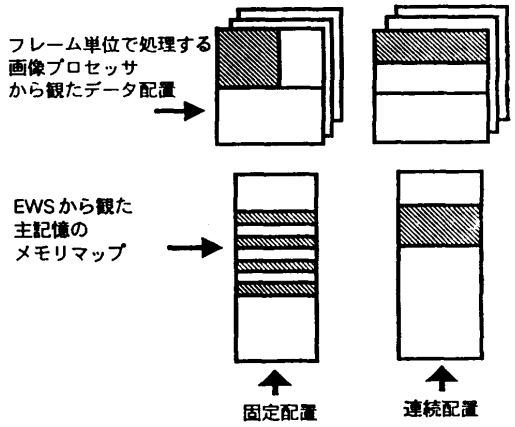


図 3 画像メモリの利用形態
Fig. 3 Data arrangement in image memories.

レベル0およびレベル1ユーザのプログラミングツールとしては，従来からの手続き型プログラミングをベースとした上に，オブジェクト指向やビジュアルプログラミングを含んだソフトウェア・アーキテクチャとなっている（**図 4**）. 主要なソフトウェア・モジュールは以下のとおりである.

(1) **V-Sugar**

VIEW-Station のソフトウェア体系の中核をなすプログラミングツールで，一種の画像処理用言語と考えることもできる.（後述するように専用の言語処理系を持たないため，既存言語のライブラリ群となっている.）V-Sugar では，画像処理プログラミングのために，画像処理特有のデータ型の導入と画像処理操作を関数型記述により簡便化したことが大きな特徴である.

V-Sugar では，V-SugarLib に用意された多数の画像処理アルゴリズムを関数の形で引用するだけでなく，ウィンドウシステム **VIEW-Windows** やファイ

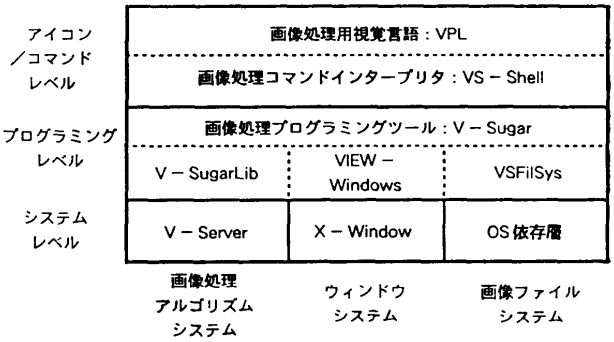


図 4 VIEW-Station ソフトウェアアーキテクチャ
Fig. 4 VIEW-Station software architecture.

ルアクセスライブラリ (VSFilSys) とのインタフェースも内包している.

(2) **V-SugarLib**

V-SugarLib は，画像処理プログラム作成のための素アルゴリズムを集めた関数ライブラリである. SPIDER などの既存ライブラリ，FIP 付属のライブラリは，入出力データを V-Sugar が提供するデータ型とすることにより，V-SugarLib 化することができる. EWS によるソフト処理と FIP によるハード処理，あるいは複数の FIP の処理機能の違いを吸収するには，同名の V-SugarLib 関数を各々用意する. すなわち，V-SugarLib を差し換えることによりハードウェア構成に依存しないプログラミングが実現できる.

(3) **V-Server**

V-Server は，画像メモリの動的な資源管理を行い，FIP/EWS の両方からアクセスするための基本操作を提供するモジュールである. V-Sugar 処理系において画像データの割当，解放を自動的に行う機能は，画像メモリの管理機構 V-Server を利用して実現されている. V-Server は，V-Sugar 処理系により起動されるため，V-Sugar プログラマはその存在を意識する必要はない.

V-Sugar 処理系が，プログラム（プロセス）内でのメモリの利用を決定すると，V-Server はプログラム（プロセス）間にまたがる大域的な画像メモリの利用状況に応じて，データ領域の占有・アクセス・解放などを動的に実行する. ハードウェア構成を意識せずに，V-Sugar 上で画像データ型を利用可能とするために，V-Server は FIP の機種や構成に応じて適切なデータアクセスを行う機構を有する.

(4) **VIEW-Windows**

VIEW-Windows は，画像処理プログラム向きの表示操作環境を提供する. これは，既存のウィンドウシステム (X Window System[10]) の上に，EWS のマルチウィンドウと FIP の画像ディスプレイの表示を管理する専用のウィンドウマネージャおよび表示操作ルーチンからなる.

(5) その他

V-Sugar より上位の層には，応用アルゴリズム検討のための対話型コマンドシェル VS-Shell やプログラミング技術の低いユーザでも使用できる画像処理用の視覚言語 VPL (Visual Programming Language) が提供される. V-Sugar のハードウェア独立という性質から，VIEW-Station ではツールの蓄積が容易であり，今後もビジョン用幾何モデラ，エキスパートシェルなど，多様なツールの開発が期待されるが，本論文では上位レベルのツールに関しては触れず，V-Sugar 以下を詳述する.

3. V-Sugar プログラミング

V-Sugar は，画像処理プログラミングを簡潔で容易なものとするために，画像処理のためのデータ型を導入し，画像処理アルゴリズムの記法として関数形式を採用している.

3.1 画像処理のためのデータ型の導入

画像データ型には次の 2 つの側面が存在する.

1) 2 値画像，ラベル画像，レンジ画像等のように画像の意味する内容によって分類される型
2) 整数／実数，符号の有無，画素あたり 8 bits/16 bits/32 bits など，画素値の格納形式の違いにより分類される型

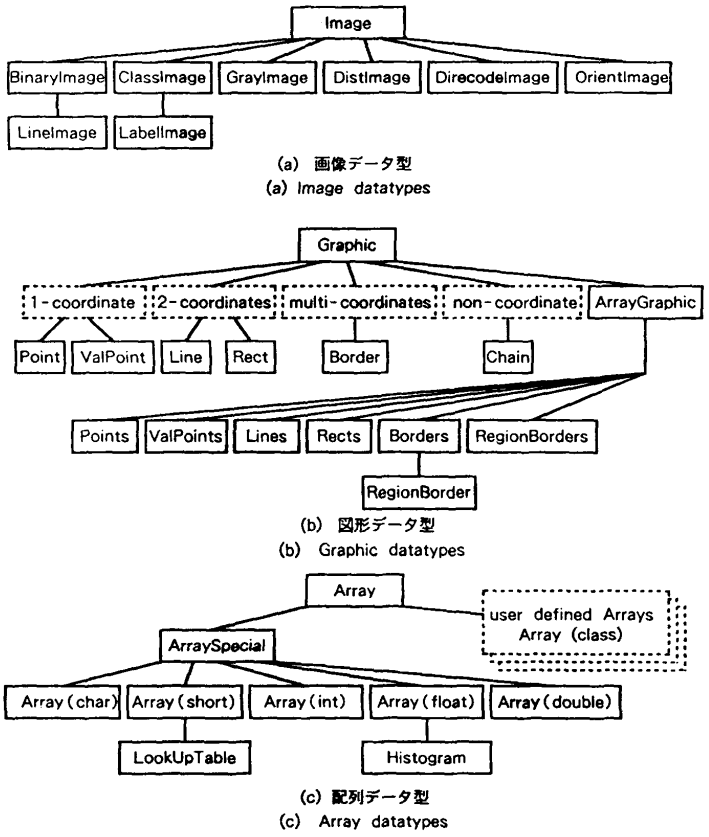
この 2 つの型をそれぞれ，「意味画像データ型」，「物理画像データ型」と呼ぶことにする.

物理画像データ型は，通常のプログラミング言語の枠内で画素データを操作する際のメモリ・アロケーションの形式であり，メモリ節約と計算精度というアルゴリズムのインプリメント時の要請から決定される. 一方，意味画像データ型は，そのデータの内容・性質に応じて，適用可能なアルゴリズムの種類や適当な表示方法等を限定する.

V-Sugar では，応用プログラマの利用する画像データ型は，意味画像データ型のみとする. すなわち応用プログラマには，アルゴリズムのインプリメントに依存するデータ形式の違いを隠蔽し，意味的なデータ型のみ意識してプログラミングできるようにする. 意味型の区分は，文献 6) による分類を基礎として，**図 5** のように階層化されている.

また，画像処理の中間結果・処理結果によく現れる画像以外のデータ型についても，代表的なデータ型を与える. 主なものとして，点，線，矩形，チェイン符号などの図形データ型や，ヒストグラム，ルックアップテーブルなどの各種配列データ型を提供している (図 5). 画像およびこれらのデータ型を合わせて **V-Sugar データ型**と呼ぶ.

各 V-Sugar データ型には，以下の機能が提供される.



(a) 画像データ型
(a) Image datatypes

(b) 図形データ型
(b) Graphic datatypes

(c) 配列データ型
(c) Array datatypes

図 5 V-Sugar データ型
Fig. 5 V-Sugar datatypes.

(a) 生成と消去

プログラム中で利用されるデータは，画像ファイル・TV カメラなどからの入力，あるいは処理の結果として得られる．記述例を下記に示す．

```
GrayImage gimg ("filename");          (1)
BinaryImage bimg=Binarize
  (gimg, 128);                        (2)
```

ここで，(1)は filename という名前のデータファイルからの入力によるデータ生成，(2)は Binarize という関数の結果によるデータ生成である．このように，応用プログラムの記述において配列領域の明示的な宣言は不要である．データ領域の確保は入力・処理の実行時に動的に行われる．

画像処理プログラムにおいて，メモリ領域の有効活用は重要であり，不要となったデータを消去してメモリ領域を再利用したい．V-Sugar プログラムでは，局所変数の消滅時に対応するデータを自動的に消去する．プログラム記法としては，「{」と「}」とで区切られたブロックを局所変数の有効な範囲とする．この機能により，関数内の中間処理結果などは関数の終了時点で消滅する．

(b) ファイル出力・ディスプレイ表示機能

データファイルへの出力やディスプレイへの表示は，すべての V-Sugar データ型に対して同一名の関数で利用する．

```
gimg. disp (frame);                   (3)
bimg. save ("filename");              (4)
```

ここで，frame は VIEW-Windows が管理する表示枠の番号であり，ディスプレイ上の表示場所を示す (5.2 節参照)．操作の内容は，必ずしも同一ではなく，対象とする変数のデータ型に応じて適切なルーチンが実行される．

(c) 基本画像演算

ここでは，画像間算術・論理演算などの低レベルな画像用の演算を基本画像演算と呼ぶことにする．これらは多くの画像処理プロセッサで実行可能であり，これらを用いてプロセッサの機種によらない処理アルゴリズムの記述が可能である．V-Sugar で利用可能な基

表 2 VIEW-Station の基本画像演算セット
Table 2 Primitive operators for image processing provided by VIEW-Station.

		入力データ	出力データ	機能
単項演算	算術演算	画像	画像	符号反転
	ビット演算	画像	画像	ビット反転
	その他	画像	画像	平行移動・拡大縮小
2項演算	算術演算	画像・画像	画像	加・減・乗・除・剰余
		画像・数値	画像	同上
	ビット演算	画像・画像	画像	AND・OR・XOR
		画像・数値	画像	AND・OR・XOR・ビットシフト
	比較演算	画像・画像	画像	等号・不等号・その他の大小関係
		画像・数値	画像	同上
		画像・画像	論理値	同上
		画像・数値	論理値	同上
	その他	画像・画像	画像	最大値・最小値出力

本画像演算を表2に示す.

3.2 画像処理関数

応用アルゴリズムの記述を簡潔なものとするため，個々の画像処理アルゴリズムを実行する画像処理関数の定義に以下のような制限を与える.

①画像データなどの引数には，仮想記憶アドレス，フレーム番号等ではなく，V-Sugar データ型を用いる.

②出力データは関数の値のみとし，引数は入力データのみとする．入出力の兼用は行わず，副作用のないものとする.

③出力データ領域・作業領域などの確保はすべて関数内部で行う.

④パラメータ・スイッチなどには，できるかぎりデフォルト値を与える.

次に画像処理関数の定義例を示す.

```
LabelImage Labelling
        (BinaryImage input, int connect=8)
```

これは，連結成分のラベリング処理の例であり，二値画像 (BinaryImage)，整数値 (連結数：connect) を入力とし，ラベル画像 (LabelImage) を出力としている．"=8" は引数 connect が省略された時のデフォルト値8を与える.

V-Sugar データ型と V-SugarLib を用いた応用プログラムの例を**図6**に示す．画像処理関数は，中間的変数を用いずに処理の連結を直接記述できる．ここで，一連の処理の連結は関数の入れ子で記述される．V-Sugar プログラムでは，このような入れ子も含めて，コンパイル時に引数の型の整合性が検証される．また，表示関数 disp は濃淡画像データ g1 とラベル画像データ l1 ではカラーマップの設定などが異なり，それぞれのデータ型に応じて適切な形式で表示を行う.

図6に示された応用プログラムは，レベル1ユーザが書く典型的なプログラムである．このレベルのユーザは，抽象データ型としての V-Sugar データ型と画像処理関数 V-SugarLib を用いて概念的なレベルで処理が記述でき，画像サイズ，画素形式，処理形式 (CPU/IPU) などを意識する必要はない.

4. V-Sugar 処理系の実現と画像メモリ管理機構

4.1 V-Sugar データ型のインプリメント

V-Sugar の実装に専用の言語処理系を作成することも可能であるが，既存のソフトウェア資源を有効活用するため，C言語の拡張であるプログラミング言語 C++[11] を親言語とし，この上に V-Sugar データ型と画像処理関数のパッケージを構成した.

```
#include <vsugar/VSugar.h>
#include <vsugar/Vlib.h>

main()
{
  LabelImage func1(LabelImage, Range, Range);
  Range crng(0.5, MAXFLOAT), arng(250, 1330);

  GrayImage g1(query_filename("input image:"));
  BinaryImage b1=Holefill(
                Binarize(g1, Threshold(g1), LT));
  LabelImage l1=func1(Labelling(b1), crng, arng);
  g1.disp(1);
  l1.disp(2);
  Area(l1).print();
  Compactness(l1).print();
}

LabelImage func1(LabelImage input, Range crng, Range arng)
{
  Array(float) cdata=Compactness(input);
  LabelImage w1=ExtractRegion(input, cdata, crng);
  Array(int) adata=Area(w1);
  LabelImage output=ExtractRegion(w1, adata, arng);
  return output;
}
```

関数 main
1. ファイル名を端末から入力後，画像の読み込み.
2. しきい値選択，二値化，穴埋め後，処理 func1 を適用.
3. 原画像（濃淡画像）と処理結果（ラベル画像）の表示.
4. 処理結果の各領域の面積，円形度を端末出力.

関数 func1
指定範囲の面積，円形度を持った領域を取り出す.

図6 V-Sugar による応用プログラム例
Fig. 6 An example of application programs.

データ領域管理などのデータ型特有の取り扱いを可能とするために，V-Sugar データ型は以下を構成要素とするクラスとして定義した.

- データの実態である配列へのポインタ，もしくは V-Server の管理する画像 ID（内容は後述）
- 配列・画像の大きさなどの付属情報
- 型特有の付属情報
- リファレンスカウンタへのポインタ
- メモリ管理関係の手続き
- ディスプレイ表示手続き
- ファイル I/O 手続き

各データ型が共通に持つ情報は上位クラス Image 型等に定義することにより下位のすべてのクラスに継承される．型特有の情報としては，例えば，LabelImage のラベル数などがある.

代入に伴う配列データのコピーを避けるために，V-Sugar データ型は実体を格納した配列を持たず，ポインタのみを持つ．実体は複数の変数から指し示され得るため，各実体を参照している変数の数（参照数）を

管理するためにリファレンスカウンタを用いる．

データの生成消滅などのメモリ管理は，変数宣言・代入・局所変数の消滅に対して起動される特別の関数，構成子・代入演算子・破棄子を用いて実現する．変数宣言（3.1 節の記述例(1)）に対する構成子は，実体となるメモリ領域・リファレンスカウンタをアロケートし，初期化する．初期化の内容は引数の型に応じて異なり，文字列である場合には，これをファイル名とみなしてデータファイルからの入力を行う．変数宣言（同記述例(2)）や代入演算子・破棄子はリファレンスカウンタの値を増減して参照数を管理し，参照数が0となった場合に実体をデアロケートする．なお，既存のプログラム形式との整合を取るために配列領域を明示的に宣言することも可能である．

ディスプレイ表示手続き・ファイル I/O 手続きなど（同記述例(3)(4)）は，各クラスの持つメンバ関数として，基本画像演算は Image 型に対するオペレータ等として，C++ の多重定義（オーバローディング）機能を用いて実装する．

4.2 画像処理関数のインプリメント

画像処理関数は，基本画像演算，EWS のソフト処理，FIP コマンドによる処理という複数の形態で実現される．**図7**はそのような各種形態で実現された画像処理関数のプログラム例である．これらはレベル0ユーザが書く典型的なプログラムであり，新規アルゴリズムが V-Sugar 上でどのように記述されるかを示している．(a)は画像基本演算を利用してアルゴリズムを記述したもので，画像プロセッサの機種によらない記述が可能となっている．switch 文中の各 case 文では，画像基本演算としての比較演算（gt など）を適用後，二値画像に型変換し（引数の1は黒画素値を指定）関数の戻り値としている．(b)，(c)はそれぞれ CPU，IPU を利用した既存ルーチン等を用いてアルゴリズムを実現したもので，以下のような方法でアクセスのための情報を得ている．

V-Sugar は，仮想記憶アドレスや物理アドレス・フレーム番号等に対応するアクセス用のデータ型を用意する．EWS 処理に対する型としては，vaddr_uchar（1バイト整数），vaddr_short（2バイト整数），vaddr_int（4バイト整数），vaddr_float（4バイト実数），vaddr_double（8バイト実数）がある．また FIP 処理に対する型は FIP の機種ごとに用意する．このように画像データ型は任意の画素形式で取り扱え，物理画像データ型の違いはこのレベルで吸収される．

```
#include <vsugar/VSugar.h>
#include <vsugar /Vllib.h>
BinaryImage Binarize(GrayImage gimg, int th, BIN_SW btype = GT)
{
        switch(btype) {
          case GT: return BinaryImage(gt(gimg, th), 1);
          case GE: return BinaryImage(ge(gimg, th), 1);
          case LT: return BinaryImage(lt(gimg, th), 1);
          case LE: return BinaryImage(le(gimg, th), 1);
          default: {
                ErrorInfo err(INVALID_PARM,
                        "binarize:BIN_SW error");
                BinaryImage bimg;
                bimg.error=err.Handler();
                return bimg;
                }
        }
}
```

(a) 基本画像演算を用いた実装例
(a) An example of V-SugarLib routines implemented on the primitive image processing operators.

```
#include <vsugar/VSugar.h>
#include <vsugar /Vllib.h>
GrayImage SobelFilter(GrayImage gimg, SOBEL_TYPE type = ABS)
{
        void egsbl_(short *, short *, short *, short *, short *);

        short isx=gimg.Xsize();
        short isy=gimg.Ysize();
        GrayImage oimg(isy, isx);
        vaddr_short a1(gimg);
        vaddr_short a2(oimg);
        short itype=(type==ABS) ? 2:1;
        egsbl_(&a1(), &a2(), &isx, &isy, &itype);
        return oimg;
}
```

(b) 既存ライブラリを用いた実装例
(b) An example of V-SugarLib routines implemented on an existing libraries.

```
#include <vsugar/VSugar.h>
#include <vsugar /Vllib.h>
BinaryImage Holefill(BinaryImage bimg, int connect =8)
{
        short isx=bimg.Xsize();
        short isy=bimg.Ysize();
        BinaryImage oimg(isy, isx);
        nexus_frame8 f1(bimg);
        nexus_frame8 f2(oimg);
        nexus_frame8 t1, t2;
        sprintf(command, "FHFILL%d, M%d, M%d, M%d, M%d",
                connect, f1(1), t1(1), t2(1), f2(1));
        ipu_put(command);
        oimg.blackvalue = 255;
        return oimg;
}
```

(c) 画像処理プロセッサを用いた実装の例
(c) An example of V-SugarLib routines implemented on a fast image processor.

図7 画像処理関数の実装例
Fig. 7 Implementations of V-SugarLib routines.

4.3 画像メモリの管理機構

V-Server は画像メモリ領域の割り当て・アクセスを大域的な使用状況に応じて動的に管理する機構であり，V-Sugar データ型の画像メモリ操作を実現するために，以下の機能を有する．

1）画像データアクセス方法の一元化

実際の形態によらず画像を一意に指示する識別子

(画像 ID) を導入し，処理ルーチンの実行形式に応じたデータ形式へと変換する機能を提供する．

2）ハードウェアに固有な情報の隠蔽

FIP の有するメモリの構成に固有な情報をすべて管理プログラム内で処理し，応用プログラムからメモリ構成に依存する部分を取り除く．同時に画像メモリを仮想化し，ハードウェアの持つメモリ量の制限を緩和する．

4.3.1 画像メモリの利用方法

画像 ID と実際の処理データとの対応付けはデータのアクセス時に行う．特別のアクセス関数を用いて画素単位での対応付けを行った場合，

- 既存プログラムの流用ができない，
- 画素ごとにオーバヘッドが発生し，処理速度の低下を招く，

などの問題があるため，画像単位のアクセス管理のみを行う．画像メモリの操作として，V-Server は，**表 3** の関数群を提供する．

ある画像処理プロセスが起動されたとき，V-Sugar データ型およびアクセス用データ型の生成消滅は**図 8** のように V-Server の要素関数の系列を発生する．この結果，各処理関数では現在の画像の配置形式とは関係なく，処理形式 (CPU/IPU)，画素形式 (8/16/32 ビット，整数／実数) の組合せで決まる様々な実行形態を取ることができる．

4.3.2 V-Server のインプリメント方式

V-Server の実現方式としては，UNIX の多重プロセス機能を利用し，処理プロセス（画像処理プログラム）とは独立な管理プロセスを用いる方式を採用した (**図 9**)．この方式による画像データの管理には，

1）画像処理プロセスも多重化が可能，
2）(ライブラリを用いる場合と比して) 各処理プロセスがコンパクトになる，
3）(特殊なドライバを用いる場合と比して) 管理プログラムの移植性が高い，

等の利点がある．

またこの方式を用いることで，プロセス間通信によるオーバヘッドが予想される．しかし，本方式では画素アクセスごとに通信が生じるのではなく，アクセスする画像に対して 1 回の通信のみなので，通信量自身はそれほど大きくはない．実際，図 6 のプログラムに対して EWS 単体で実行した場合 (EWS 単体での実現に関しては，次項 4.3.3 参照)，独立した管理プロセスとして実現すると 27.9 秒，メモリ管理を単なるライブラリとして実現すると 26.6 秒であった．

表 3 V-Server の機能
Table 3 Functions of V-Server.

画像データの登録と消去	登録	画像の物理的属性を管理テーブルに登録し，画像 id と画像データのリンクをとる．(IMalloc)
	消去	画像データ領域を消去し，管理テーブルから画像 id に関する情報を削除する．(IMdealloc)
画像データの占有と開放	占有	画像 id に対する画像データ領域を確保し，他のプロセスからアクセスできなくする．(IMoccupy)
	開放	占有されている画像データ領域を開放する．画像データ領域は消去されない．(IMrelease)
名前による画像 id の参照	付加	画像に名前を付ける．名前を付加された画像は消去できない．(IMnamed)
	削除	画像に付加された名前を削除する．(IMunnamed)
	参照	画像に付加された名前と画像 id のマッピングをとる．(IMgetid, IMgetname)

アプリケーションレベル
1 サーバ、IPU との結合 (IMopen)
2 入力画像の割当 (IMalloc, IMgetid)
(サーバプロセスが管理している画像を利用する場合は，名前より画像 id を参照出来る．)
3 処理関数の呼出し (入力：入力画像 id，　出力：出力画像 id)
4 必要な画像の保持 (IMnamed)
5 画像の消去 (IMdealloc)
(名前の付いている画像は，消去されない．)
6 サーバ、IPU との結合解除 (IMclose)

処理関数レベル
3a 出力用画像の設定 (IMalloc)
3b 入力／出力画像の占有 (IMoccupy)
(処理の内容に応じた画像参照情報 [仮想アドレス、フレームメモリ番号等] が，得られる．)
3c 処理 (SPIDER など)
3d 入力／出力画像の開放 (IMrelease)
3e 出力画像 id をリターン

図 8 V-Server の動作系列
Fig. 8 Sequence of V-Server's commands for utilizing image memories.

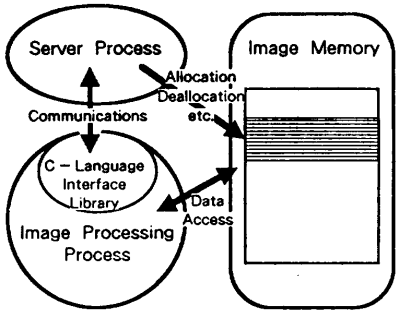


図 9 処理プロセスとサーバの関係
Fig. 9 Image processing process and V-Server.

V-Serverは，画像メモリの空領域リストと画像データの配置情報を格納したテーブルを用いて，画像データの割り当てを行う．ここで，占有操作時に以下の3つの動作により，実メモリの割り当てを行う．

(i) 配置形式の変換

占有要求に対し，V-Server は，現在の配置形式と要求された配置形式が一致しなければ，要求された形式に必要なメモリ容量を新たに確保し，現在のデータを型変換して新領域にコピーする．旧領域は開放する．

(ii) 画像データの退避

新たな領域を割り当てる際に，画像メモリ中の空領域に不足が生じた場合，V-Server は，占有状態にない画像データを EWS 内のメモリ上に退避させ，空領域を作成する．これにより，プログラム全体で利用可能なメモリ量は EWS の最大仮想記憶サイズ分（スワップサイズにより決まる），実際の画像メモリサイズより拡張されることになる．

(iii) アクセス用のデータへの変換

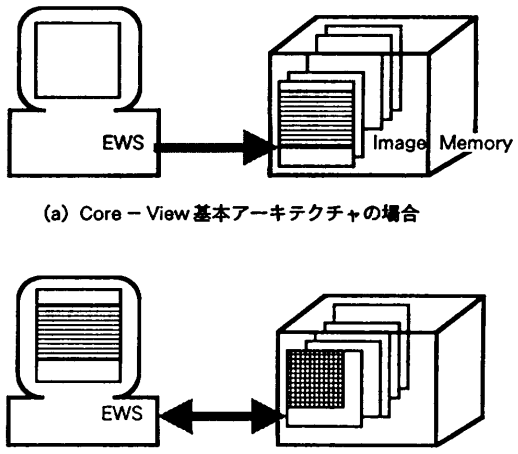
FIP 処理においては，データ領域の物理アドレスを直接返し，CPU 処理では，メモリ・マッピングを行って得る仮想記憶アドレスを返す．

4.3.3 非 VIEW-Station アーキテクチャへの対応

VIEW-Station 仕様では，画像メモリが EWS からダイレクトマッピングされていることを要求している．この純正仕様を満足していないシステムについても，V-Server が上述の配置形式の変換，アクセス用データへの変換を各画像メモリ形式に応じて行うことにより，かなりの程度対応可能である．

実際，ダイレクトマッピング可能であるが固定配置を採用している FIP の場合は，図 10 (b)に示すように，EWS 内のメインメモリと FIP 内のフレームメモリの両方を利用することで対応できる．すなわち，図 10(a)のような純正仕様では画像をすべてフレームメモリ上に置くのに対し，図 10(b)の場合は CPU 処理ではメインメモリを，IPU 処理ではフレームメモリを利用する．

また外部メモリを持たない EWS 単体での利用に対しては，EWS 内のメインメモリ上に画像を置くことで対応できる．このように V-Server 内部でシステム構成の違いを吸収することで，共通のインタフェースを上位のプログラミングレベルに対し提供することが可能となる．



(a) Core－View基本アーキテクチャの場合

(b) フレーム単位でアクセスする画像プロセッサの場合

図 10 異なるハードウェア構成への対応
Fig. 10 Implementations for different types of hardware configuration.

5. 画像処理に適したウィンドウシステム

5.1 ウィンドウシステムへの要求

画像処理システムは，画像，ヒストグラム，ルックアップテーブル等の画像処理特有の配列やデータを取り扱うとともに，高精細画像やカラー画像といった多ビットデータを頻繁に取り扱う必要がある．VIEW-Station 基本アーキテクチャは EWS と FIP という構成であり，EWS のディスプレイ以外に FIP 側に独立のディスプレイ（以下，画像ディスプレイとする）を有する場合も多い．

これらを考慮して，画像処理のためのウィンドウシステム VIEW-Windows の目標を以下のように設定した．

①多様な表示デバイスへの対応

画像表示装置として，マルチウィンドウの表示管理が可能な通常の EWS ディスプレイとフルカラー表示可能な画像ディスプレイの単独利用／併用を許す．また，EWS ディスプレイのプレーン数 (1, 8, 24 ビット／ピクセル等) の違いを意識せず取り扱えるプログラム／操作環境を提供する．

②画像処理向き表示ツール

画像処理プログラムで頻繁に用いられるヒストグラム・LUT 等に対して，これらの表示・対話操作のためのツールを提供する．

③V-Sugar データ型の利用

V-Sugar で導入した画像データ型（濃淡画像，二値画像，ラベル画像等）に対し，各型に適合した形式で

の表示を可能とする.

ここで，他のウィンドウツールとの整合のために，VIEW-Windows 実現のベースとして X Window System/X Toolkit を採用することとした.

5.2 VIEW-Windows の機能と実現

5.2.1 画像表示の一元化

表示デバイスの違いを吸収し，画像表示に対するユーザプログラムからの切り口を一元化するため，以下の方式を採用した.

- 画像表示のための多数の枠 (frame) を設定し，画像データは必ずこの frame に表示されるものとする.
- EWS の標準ディスプレイとは別に用意される画像ディスプレイは，frame の1つと考える. すなわち，画像ディスプレイを EWS ディスプレイの外延領域とし，すべての表示操作を frame への操作として一元化する.
- frame の表示管理を統一的に行うため，アプリケーションプログラムとは独立した **Frame Manager** なるプロセスを導入する.

5.2.2 Frame Manager の実現

Frame Manager の実現には，サーバ・クライアント方式を採用した (**図 11**). 画像表示に対する各プロセスの動作は次のとおりである.

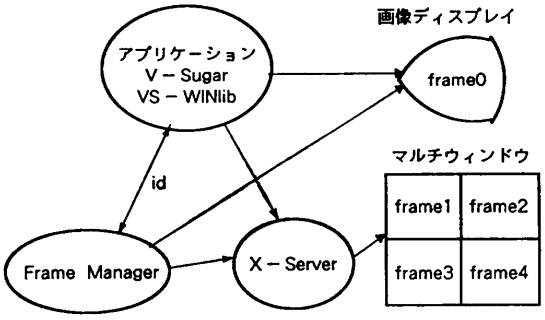


図 11 プロセス構成
Fig. 11 Process configuration of VIEW-Windows.

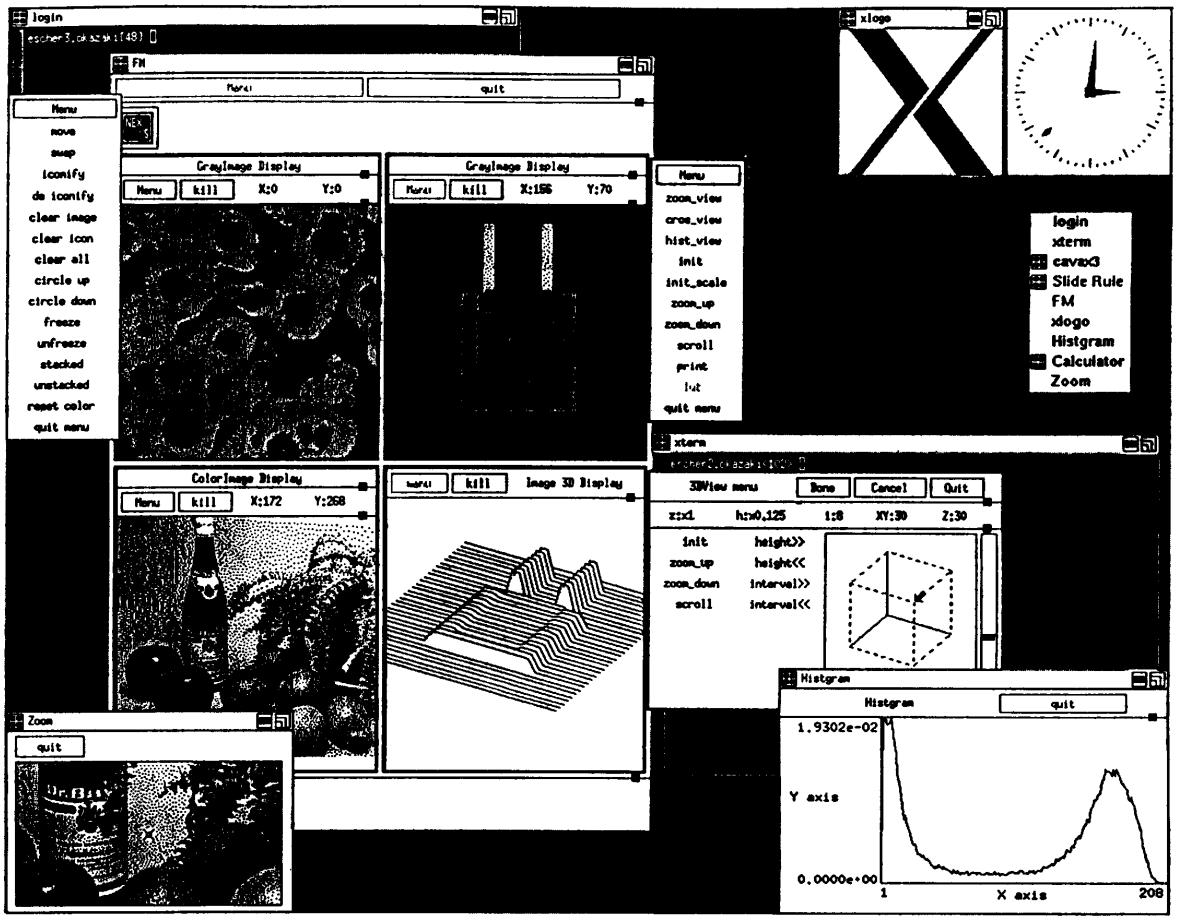


図 12 VIEW-Windows の一画面
Fig. 12 A snapshot of VIEW-Windows.

①アプリケーションプログラム（クライアント）は，Frame Manager（サーバ）に対して，frame 番号を指定する．

②Frame Manager は，指定 frame について，X-Server の扱うウィンドウ id もしくは画像ディスプレイを表す id を返す．

③アプリケーションプログラムは，この id に従い，EWS のウィンドウもしくは画像ディスプレイへの描画を実行する．

VIEW-Windows では，移植性の高さを実現するために，X-Server 自体には手をいれず，画像ディスプレイへの表示操作はアプリケーションプログラム自体が行う．画像の表示位置の変更などに対応する X-Server の機能を代行するために，画像ディスプレイへの表示管理では，クライアントメッセージ機構を用いて Frame Manager から応用プログラムへイベントが送られる．

表 4 VIEW-Windows の表示機能
Table 4 Display functions of VIEW-Windows.

表示オブジェクト	機　能	ライブラリルーチン
・ヒストグラム ・LUT ・断面像，等	ズーム スクロール スケール変更 データ値の表示 位置・データ値の入力	(グラフ表示用ルーチン) disp_1d_short () 等 (対話型パラメータ入力ルーチン) get_lut_val () 等
・濃淡画像・ラベル画像等の画像データ ・パワースペクトラム等の2次元配列データ	スケール変更 ズーム LUT 操作 スクロール データ値の表示 断面の表示 位置・データの入力	(画像表示用ルーチン) disp_2D_short () 等 (対話型パラメータ入力ルーチン) get_img_val () 等
・距離画像等の画像データ ・周波数フィルタ等の配列データ	視点の移動・回転 ズーム，等	(鳥瞰表示ルーチン) disp_3D_short () 等

5.2.3 frame の表示形式と操作

frame の表示は，画像どうし重なりのないタイリング形式を採用する．（画像以外のコマンドメニュー等はオーバラップを許す．）VIEW-Windows の画面の一例を**図 12** に示す．frame の並べ方，数は Frame Manager 起動時に指定する．Frame Manager の提供する機能には，frame 間での画像の移動・入れ替え等の操作，画像の一時待避のためのアイコン化等がある．これらは，ウィンドウ／画像ディプレイの別なく操作可能である．この意味で，Frame Manager は画像表示を一括管理する特殊なウィンドウマネージャであるといえる．

5.2.4 表示・対話機能

VIEW-Windows では，画像データおよび画像処理の結果生じる各種データを表示・ポインティングする機能を，VS-WINlib なるライブラリの形で提供する．**表 4** に代表的な機能を示す．また，実際に適切な表示を行うためには，表示ディスプレイに応じたビット数の圧縮が重要であり，VS-WINlib はディザなどの擬似濃淡表示，ピーノマップによる色空間の圧縮などの機能を有する．Frame Manager は，多数の画像を適切な形式で同時に観察するために，グレイスケール，シュードカラー，ラベル用カラー，2値表示などの多様なカラーマップとカラーマップ自体を分割管理する機能を有する．

また，VS-WINlib を構成する画像・グラフ・鳥瞰図などの表示モジュールは X Toolkit の widget（プログラム部品）として構成され，一般に流布する widget との組合せにより，多様なユーザインタフェースを実現することができる．

6. むすび

ハードウェアによる信号レベル処理の高速実行と，EWS 環境における柔軟なソフトウェア処理の両立を可能とする画像処理ワークステーション VIEW-Station を提案し，その基幹ソフトウェアシステムを開発した．

中核となるプログラミングツール V-Sugar は，画像処理特有のデータ型や画像処理関数を導入することにより，高次の画像処理プログラミングを可能にしている．また，V-Sugar は，画像メモリの動的管理機構 V-Server や画像用ウィンドウシステム VIEW-Windows などと一体となって，ハードウェア構成に依存しないソフトウェア環境を実現している．

VIEW-Station は，これから応用プログラミングを促進する基盤として，大局的な観点からのソフトウェアアーキテクチャを採用している．このため，従来の汎用言語によるプログラミングに慣れ親しんだユーザには，かえって V-Sugar の仕様は煩わしく感じるかもしれない．一方，VIEW-Windows の画像やその他

のデータ表示機能は斬新なものではないが，重複投資を避けるため敢えて標準的な機能を新しい枠組の上に提案したものである．この種のソフトウェアは，使い込まれて順次改良されていくべきものである．

V-Sugar および VIEW-Windows は，現在，市販の EWS として，Sun 3 または Sun 4 シリーズ，FIP として nexus 6810 または VICOM/VME の組合せの下で稼動している．これらは，準パブリックドメイン・ソフトウェアとして公開しているので，他の多くの EWS や FIP に移植されつつある．こうした基幹ソフトウェアの上に新しい画像処理ソフトウェアが蓄積され流通していくことを期待している．

謝辞 VIEW-Station ソフトウェアの検討・評価をして頂いた，阿部圭一（静岡大学），松山隆司（岡山大学），谷口倫一郎（九州大学），浅田尚紀（京都大学）の諸先生に深謝いたします．

参考文献

1) 田村，坂根，富田，横矢，金子，坂上：ポータブル画像処理ソフトウェアパッケージ SPIDER の開発，情報処理学会論文誌，Vol. 23, No. 3, pp. 321-328 (1982).
2) 坂上：イメージプロセッサとその基本画像演算について，テレビジョン学会技術報告，VVI 73-1 (1985).
3) Kanade, T. and Webb, J. A.: End of Year Report for Parallel Vision Algorithm Design and Implementation, CMU Technical Report, CMU-RI-TR-88-11 (1988).
4) Hamey, L., Printz, H., Reece, D. and Shafer, S.: A Programmer's Guide to the Generalized Image Library, Carnegie-Mellon University (1987).
5) 松山，村山，伊藤：画像処理演算の複合的合成，情報処理学会コンピュータビジョン研究報告，CV 43-4 (1986).
6) 阿部，田村，坂上，鳥脇：画像処理ソフトウェア・パッケージ SPIDER-II の新機能(1)—画像データ型の導入と引数の分類，第 32 回情報処理学会全国大会論文集，1N-2, pp. 1319-1320 (1986).
7) Swick, R. R. and Weissman, W.: X Toolkit Widget—C Language X Interface, X Window System Version 11, Release 2, Massachusetts Institute of Technology (1987).
8) 次世代画像処理ソフトウェアシステムに関する調査報告書，電子情報通信学会エキスパートビジョン研究会 (1989).
9) 藤村：画像処理—イメージング—の国際標準化動向 (SC 24 関連)，第 20 回画像工学コンファレンス論文集，4-1, pp. 105-108 (1989).
10) Gettys, J., Newman, R. and Scheifler, R. W.: Xlib—C Language X Interface, X Window System Version 11, Release 2, Massachusetts Institute of Technology (1987).
11) Stroustrup, B.: *The C++ Programming Language*, Addison-Wesley (1986).

(平成元年 8 月 31 日受付)
(平成 2 年 4 月 17 日採録)

佐藤 宏明（正会員）

昭和 35 年生．昭和 57 年東京大学工学部産業機械工学科卒業．同年キヤノン(株)入社．現在，同社情報システム研究所に勤務．平成元年より 2 年間の予定でスタンフォード大学客員研究員．画像処理システム・知識ベースシステムの研究に従事．

岡崎 洋（正会員）

昭和 37 年生．昭和 60 年京都大学工学部情報工学科卒業．昭和 62 年同大学院修士課程修了．同年，キヤノン(株)入社．現在，同社情報システム研究所に勤務．画像処理ワークステーションのソフトウェアに関する研究に従事．平成元年第 38 回全国大会学術奨励賞受賞．

河合 智明（正会員）

昭和 35 年生．昭和 59 年東京大学工学部電気工学科卒業．昭和 61 年同大学院修士課程修了．同年，キヤノン(株)入社．現在，同社情報システム研究所に勤務．画像処理ワークステーションの研究に従事．電子情報通信学会会員．

山本 裕之

昭和 37 年生．昭和 59 年大阪大学基礎工学部制御工学科卒業．昭和 61 年同大学院修士課程修了．同年，キヤノン(株)入社．現在，同社情報システム研究所に勤務．3 次元画像計測・認識，画像処理ワークステーションの研究に従事．

田村 秀行（正会員）

昭和 22 年生．昭和 45 年京都大学工学部電気工学科卒業．昭和 47 年電子技術総合研究所入所．SPIDER の開発等，パターン認識・画像情報処理の研究に従事．画像処理研究室主任研究官を経て，昭和 61 年 4 月キヤノン(株)入社．現在，同社情報システム研究所知能工学研究部長兼画像情報研究部長．工学博士．昭和 60 年度論文賞受賞．著書：「コンピュータ画像処理入門」など．IEEE, ACM, 電子情報通信学会，人工知能学会各会員．