コンピュータビジョン 47－3
（１９８７．３．１７）

対話型画像処理支援ハンドラ

高木幹雄　鈴置 雅一
東京大学生産技術研究所

本システムは，それぞれ独自の操作環境を持つさまざまの画像処理ソフトウェア／ハードウエア資源を統合し，ユーザに一貫したコマンドレベルでの対話環境を提供するものである。したがって本システムは，あらかじめ具体的な画像入・出力装置あるいは画像処理プログラムモジュールが組み込まれているのではない。全ての入出力は仮想化され，ハードウエアからの独立性を保っている。処理モジュールはインストラクションリストと呼ばれるリストとともにシステムに登録され，登録された処理モジュールを実行する環境はそのインストラクションリストの指示に基づいて整えられる。
データは変数リストで管理され，変数リストは階層的構造を持ち，サブフレームの定義に有用である。これによって，一度に主記憶上に乗りきらないような大容量のデータを扱うことを可能にしている。

INTERACTIVE IMAGE PROCESSING HANDLER

Mikio TAKAGI　　Masakazu SUZUOKI
Institute of Industrial Science, University of Tokyo
7 - 22 - 1, Roppongi, Minato-ku, Tokyo, Japan

This system integrates several image processing hardware/software resources which have their own operation environment, and give the user a standardized interactive environment ftff command string level. Therefore the system has no built-in image I/O devices and processing program modules in it. All data I/O is operated virtually to keep fhe machine independence. Processing modules are registered at the system with the list called Instruction List. The environment to execute the module is prepared according to the instruction in the Instruction List.
The data is handled by the Variable List which has hierarchy structure. This enables the handling of the data that is too large to set on the main memory at once.

1．まえがき

計算機によるディジタル画像処理技術の蓄積は，近年相当数にわたり，画像入出力装置，画像処理プロセッサ，画像データベース，画像処理サブルーチンパッケージ等，さまざまなハードウエア・ソフトウエア資産が多数開発されてきている。

しかし，それらは互いに独立した設計思想に基いて開発されているため，それぞれをサブシステムとして組合せ，統合的に活用したい場合に様々な整合性の問題が生じる。

特にソフトウエア的な不整合は，ユーザに不必要な負荷を課す結果となる，各サブシステムが独自の操作環境・データ構成を取っているために，複数のサブシステム上にまたがって開発する場合ユーザは，サブシステムごとにその環境を使い分けなくてはならないからである。

このため，個々のサブシステム内ではきわめて合理的，統一的に設計されたシステム体系も単にユーザの労力を増大させるだけとなり，その結果，現在にいたるまでの有益な開発資源が，十分に活用されていないのが現状である。

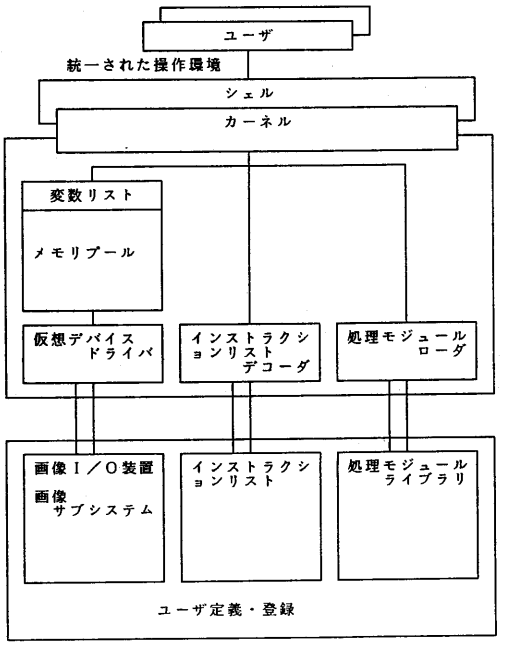

FIG 1. 外部デバイス・処理モジュールの登録

2．システムの設計方針

2．1 外部デバイス，処理モジュールの登録

本システムは既存の画像処理資源を有効に活用することが目的であり，従って具体的な画像処理ソフトウェア，あるいはデバイスドライバを持っているわけではない。それらは，個々のシステムの目的，あるいは現在もっている画像処理資源に応じて，定義・登録されるものである。

・仮想デバイス

システムからは，画像処理デバイスはオブジェクトと考えられ，総て単一の仮想デバイスとして扱われる。システムは，各デバイスの操作をする代りに各デバイス固有に割当てられたポインタを操作し，デバイスを稼働する代りにデバイスへのポインタと必要なパラメータをその仮想的デバイスドライバへ送るだけである。実際のデバイスドライバは，その下にユーザの責任のもとで定義される。

従って外部からのデバイスドライバの登録・接続は，十分柔軟性があり，かつ簡便であるようにこころがけた。

・処理モジュール

ここでは処理モジュールとは一般の画像処理サブルーチンプログラムのロードモジュールを考える。処理モジュールはデバイスに比べて頻繁に登録・削除されることが考えられる。このため処理モジュールの登録・削除にはより汎用性・容易性をもたせる必要がある。さらに，処理モジュールを登録する際には，処理モジュール側をシステムにあわせて変更・書換えをするようなことは避けなくてはいけない。

こうした不都合を回避するため，本システムは，これら独自の操作環境を持つさまざまの画像処理ソフトウェア／ハードウエア資源を統合し，ユーザに一貫したコマンドレベルでの対話環境を提供することを目的として設計された。

2．2　環境の設定

システムの対象とする処理モジュールは，独立したプログラムではなく，引数であたえられる変数・配列は外部的とみなされ，モジュール内にデータ領域を持っていない。従って．これらの処理モジュールを起動するに先立って，未解決の外部変数のための領域を確保・設定し．そのポインタ（アドレス）を引数の形で処理モジュールに渡す必要がある。このような処理モジュールを対象とする場合，システムは処理モジュールを起動する際，その内部構造にまで立ち入って環境を整なくてはならない。

これら各処理モジュールジュールに関する情報をシステムは何らかの方法で知る必要がある。そのためシステムでは処理モジュールの仕様を記したインストラクションリストと呼ばれるリストを定義した。処理モジュールを登録するには，ユーザはこのインストラクションリストをシステムに定義しさえすれば良い。

2．3　対話性

こうした方式は，実行可能な処理モジュールを対象とする場合よりもシステム側の負担は重くなるものの，システムにより一貫的な主記憶の管理が可能となるため，動的な領域の割当て・変更・表示が可能となる。システムではこれらの処理は，変数リストと呼ばれるテーブルで統一的に取り扱い，ユーザが処理モジュールを取り扱う操作環境には，十分な対話的・逐次的環境を提供した。

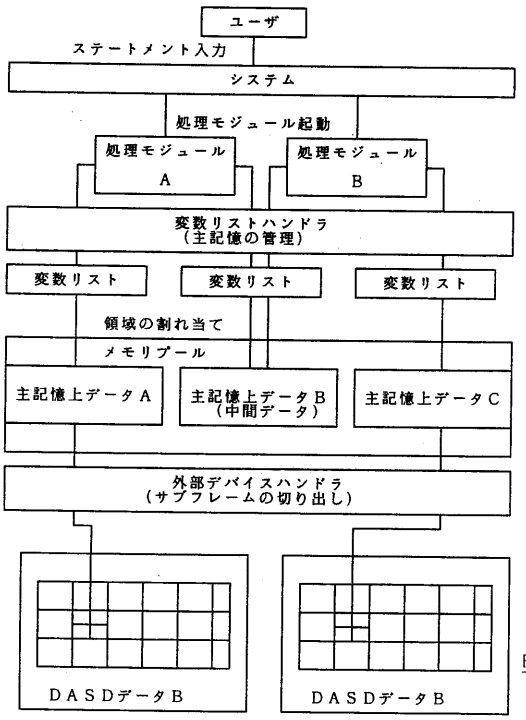

FIG 2.　環境の設定

2．4　柔軟性

接続される外部デバイスは，様々な画像データ構成をとるここが考えられる。これに対応するため，システムの持つ画像データ構成には．十分な柔軟性が必要である。特に．取り扱いができる画像データの大きさに制限があることは望ましくない。主記憶に乗り切らないデータでも統一的に扱うことのできるデータ形式をとる必要がある。このため，画像変数のデータ構成に階層性を持たせて，サブフレームの定義を自然に導入し，一貫的な処理を可能にした。大容量の画像であっても，そのサブフレームを定義し，サブフレームのみを順に主記憶上に展開することによって処理を進めていくことができる。

3．システムの構成

プログラムは，

a）　データ・パラメータ，およびそれらの属性を管理する変数リスト，

b）　処理モジュールにパラメータを割当て起動させ

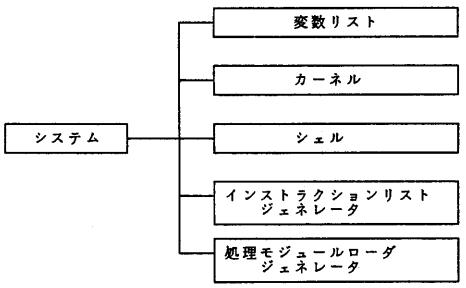

FIG 3.　システムの構成

るカーネル，

c） 対話的インターフェースを提供するシェル，

d） 登録する処理モジュールのソースリストを解析してそのインストラクションリストを生成するインストラクションリストジェネレータ

e） インストラクションリストよるシステムのローダルーチンを生成するローダジェネレータ

から構成される。変数リストは内部にリスト処理部，記憶管理部を持ち，カーネルは各処理モジュールのコーリングシーケンスの仕様を記したリスト（インストラクションリスト）を持つ。さらに，処理モジュールのソースリストからインストラクションリストを生成するソースリストアナライザ，およびインストラクションリストからローダルーチンを生成するローダジェネレータがある。

3．1 インストラクションリスト

システムが，登録されている外部処理モジュールへ渡すパラメータを割当てる際，必要なパラメータ列は，個々の処理モジュールによって異なるため，システムは処理モジュールのコーリングシーケンスの仕様を定めたリスト（インストラクションリスト）を処理モジュールごとに持っている。

リストには処理モジュールのパラメータごとに入力されたコマンド列の解釈を指示したフォーマット，および各パラメータの属性（必要な領域の大きさ，入力・出力等）に関する情報が記されている。システムは，

インストラクションリストのレコード構成

1	（処理モジュール名）（語長）（パラメータの総数）
2	（インストラクションリストの本体１）
3	（インストラクションリストの本体２）
4	（ヘルプリスト）

８０バイト

登録すべきサブルーチンＦＬＷＬ１

```
SUBROUTINE FLWL1(IP, JP, ISX,
                 ISY, IWHGT, ISX2, ISY2,
                             GAIN, IERR)
```

ＦＬＷＬ１に対するインストラクションリスト

```
FLWL1  4  16
1DT(SP3P4) 2DT(AP34) 1FX1FY
3DT(SP6P7) 3FX3FY4ES(S) 5ES(S)
```

システムからの起動

```
FLWL1  IP  JP  WEIGHT  1.0;
```

コマンドの第一引数をキーにインストラクションリストを読みこみ，リストを解釈しながら，コマンドの第二引数以下を評価する。ついで，変数リストの属性を調べてサイズの整合，入力・出力の整合の検査を行う。エラーがなければ，処理モジュールをリンク・ロード・実行する。

例えば空間フィルタリングサブルーチン　ＦＬＷＬ１，

```
SUBTOURINE FLWL1(IP, JP, ISX, ISY, IWHIGHT,
                                ISX2, ISY2. GAIN, JERR)
```

をシステムに登録すれば，その起動は，例えば，

```
FLWL1  IP  JP  IWHGT  1.0
```

ですむ。変数ＩＰ，ＪＰのサイズ（ＩＳＸ，ＩＳＹ）などは自動的に渡される。

3．2 変数リスト

画像データは，すべて変数と考えられ変数リストで管理される。変数リストは，変数名といくつかの属性，データへのポインタを含むリストであり，主記憶を動的に管理し，領域の割当て・開放をダイナミックに行う。

・変数の次元

画像は一般には２次元データ構造であるが１画素の語長も含めた３次元の空間として考えた方が一般性が高い。そこで．システムでは変数を３次元直方体の空間として考え，そのサイズとして　ＳＢ，ＳＸ，ＳＹ　の属性をもたせる。画素の単位はバイトである。例えば　１バイト長の　５１２ｘ４８０　の画像であれば

（ＳＢ，ＳＸ，ＳＹ）＝（１，５１２，４８０）

となる。

画像以外のデータ，例えばベクトルデータ，スカラーデータは

ＳＹ＝１，或いは　ＳＸ＝ＳＹ＝１

FIG 4. インストラクションリスト

の属性を持つ変数として扱う。このようにシステムではフラグ等のパラメータもチェーンコードのようなベクトルデータも画像も全て同列の変数で扱えるようにしてあり，操作上の統一性を考慮している。以下　画像・スカラー・ベクトル　を統一して全て変数と呼ぶ。

・変数リストのポインタ

変数リストは，各変数の属性を与えるだけで実際のデータ本体は持っていない。各リストは，データ本体の代りにそのデータへのポインタを持つ。ポインタは，主記憶へのアドレスであったり，他の変数名であったりする．システムの主記憶管理ルーチン，および外部のデバイスハンドラはこのポインタを参照して実際のデータの取り扱いをおこなう。

ポインタの指すものには，

a）主記憶へのアドレス，
b）データセット名・ディスプレイプレーン等の外部論理デバイス名，
c）他の変数リスト

のいずれかであり，これに応じて，変数はＲＥＧＩ，ＥＸＴＲ，ＳＣＯＰの状態をとる。このように，リストのポインタに他の画像変数を指定することによって，階層的なデータ構造がとれる。他の変数へリンクされた変数をサブフレームと呼ぶ。

・フレームとサブフレーム

サブフレームは，一種のウィンドウであり実際の領域，外部デバイスをもたない論理的なものである。これに対し，実在の領域，デバイスを持つ変数をフレームと呼ぶ．一つのフレームに複数のサブフレームをリンクすることができる．サブフレームの下にサブフレームをリンクすることもできる。

このように，変数の階層的な定義が自然にできため，ディスプレイ画面を複数のウインドウに分割して表示したり，あるいは，主記憶に乗りきらないような大画像をサブフレームに分割して処理する際にきわめて効果的である。

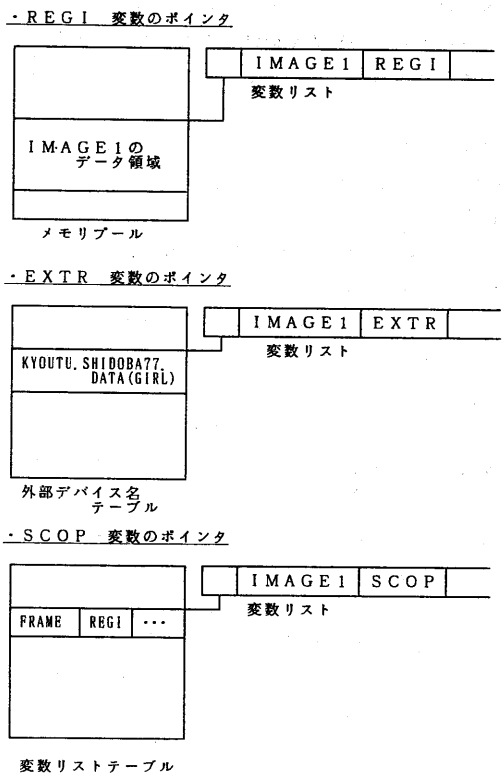

FIG 5.　変数リストのポインタ

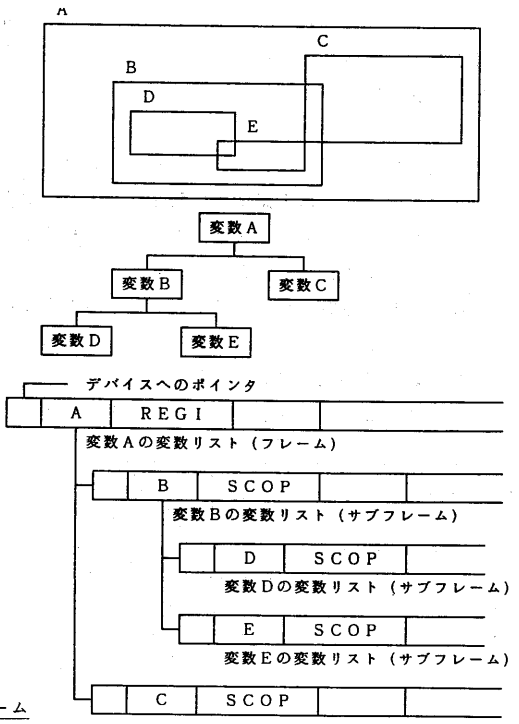

FIG 6.　フレームとサブフレーム

例えば，１枚の１ｘ５１２ｘ４８０のディスプレイ・プレーンに割り当てられた変数に１ｘ２５６ｘ２４０の４枚のサブフレームをリンクすれば１枚のプレーンを４枚の独立したプレーンとして扱うことができる。

・主記憶の動的割当て

ＲＥＧＩＯＮ属性が割当てられた変数リストが宣言されたときにのみ，主記憶内に領域がとられる。このリストが開放されると，割当てられていた領域も開放される。　従って処理に先だってあらかじめ画像のサイズを定義する煩わしさはない。作業用領域はそのモジュールを起動する前に宣言しモジュールの処理が終了したら消去すればよい。このようにして常に必要なデータだけを主記憶上に常駐させることができるため，主記憶の効率化がはかることができる。

例えば濃淡画像を二値化し細線化を行う場合二値化処理モジュール　ＳＬＴＨ１　の出力は，細線化処理モジュール　ＴＨＩＮ１　の入力に用いた後は不要になる。こうした変数は一時的に宣言し処理の後に開放すればよい。

```
ＳＥＴＵＰ　ＷＯＲＫ　Ｒ．ＳＯＵＲＣＥ；
ＳＬＴＨ１　ＳＯＵＲＣＥ　ＷＯＲＫ　ＴＨ
                                ＳＷＩＴＣＨ
ＴＨＩＮ１　ＷＯＲＫ　ＲＥＳＵＬＴ　ＳＴＥＰ；
ＳＥＴＵＰ　ＷＯＲＫ　Ｆ．；
```

この例では第一行で変数　ＷＯＲＫ　が宣言される。Ｒ．ＳＯＵＲＣＥ　は主記憶内に領域　をとり，変数　ＳＯＵＲＣＥ　とおなじ属性をもつことを指定するものである。第４行では変数　ＷＯＲＫ　に　ＦＲＥＥ　属性が与えられ領域が開放される。こうした主記憶の割当て・開放は，すべて自動的におこなわれ，ユーザは意識する必要はない．ＳＥＴＵＰ　はシステムコマンドで変数リストを生成（ＳＥＴＵＰ）するものである。

ＩＮ	入力機番に変更
ＲＥＴＵＲＮ	入力機番に復帰
ＬＯＧ	ロギング機番の指定
ＩＦ／ＥＬＳＥ／ＥＮＤＩＦ	ブロック化ＩＦ
ＧＯＴＯ	無条件ジャンプ
ＢＹＥ	システム終了

FIG 7.　シェルコマンド

３．３　カーネル

カーネルは，入力されたコマンドに応じてインストラクションリストを読み込み，この指示にもとづき，パラメータの設定・チェックを行い処理モジュールをロード・実行する。ユーザは，インストラクションリストを書き替えることによって自由に，既存のサブルーチンパッケージ・開発中のソフトウェアをシステムに登録・削除できる。カーネルは，処理モジュールの制御に，完全に対話的，インタープリティブな環境を提供する。従って処理モジュール単位の処理の途中で，動的に変数を宣言・開放を行ったり，変数の内容の表示および変更が可能である。

３．４．シェル

シェルは，カーネルの上位に位置し，コマンド入出力管理機能・対話的機能を提供するとともに，制御ステートメントを持ち，シェル上でのプログラムが可能である。

システムへの入力はステートメントと呼ばれる単位で行われこれの入力は端末からでもランファイルからでもよい。入出力の切り替えはシェルによって動的に制御される。処理を頻繁に行うようであれば，これをあらかじめランファイルに保存しておけばよい。　端末からランファイルを起動するには，ファイルの割り当てられた機番を　ＩＮ　コマンドによって指定すればよい。

コマンドにシンタックスエラーが検出されると，シェルは入力を自動的に端末へ切り替え，再入力を促進する。

またコマンドおよび，システムからの応答は，全て端末へエコーされるが，この出力を他のファイルに切り替えることによりロギングが可能である。ロギングファイルはそのままの形で再び入力ファイルとして使用できるフォーマットで出力され，処理のトレースバックが容易である。

3．5．システムルーチン

変数リストの制御および処理支援のために，いくつかのシステムルーチンが容易されている。これらは全てサブルーチンモジュールで，他の処理モジュールと同様に，インストラクションリストとともにカーネルに登録されている。従って，システムルーチン群も，ユーザ定義の処理モジュール群も，全く同等の呼び出し手順でコールされるために，なんら操作に替ることはない。

4．システムの操作

4．1　引数の評価

引数の評価は通常次の基準で行われる。

```
LET  DATA:=  123     整数123を代入
LET  DATA:=  123.    実数123を代入
                     (小数点がついている)
LET  DATA:="1234"    文字列123を
                     代入
LET  DATA:=  TEMP    変数TEMPを
                     代入
LET  DATA:="TEMP"文字列TEMPを
                     代入
```

・NE属性

システムでは文字列引数の評価をとめるために，NE属性を設けている。ここのフィールドの変数は評価されない。これは例えば，

```
SETUP  "IMAGE1"  "R. IMAGE"
LET  A  ":="  B
```

のように，煩雑な"（ダブルクォーテーション）の汎用を避けるためである。

4．2　整合性の検査

システムの検出するエラーには，シンタックスエラー，整合性のエラーがある。さたに整合性の検査には，領域の整合性の検査，入出力の検査がある。これらの

setup	変数リストの生成
例	setup DISPLAY E. NX. DDDA; setup SWR.; setup WORK F.
copy	変数リストのデータの転送
例	copy SIDBA GIRL 2; copy MATRIX *;
alter	変数リストの属性の変更
例	alter WINDOW LX := 128; alter WINDOW LX :+ STEP;
list	変数リストの一覧の表示
例	list;
print	変数の内容の表示
例	print ERRORCODE INT; ptint "INPUT FILE NAME";
let	変数の代入
例	let STR := "ABCDE"; let COUNT :+ 1; let SIZE := IMAGE. FX;
comp	変数の比較
例	comp SIZE != DATA. FX; comp WINDOW. LX <= NEXUS. FX
status	システム変数の変更
例	status FWRN 1;

FIG 8.　システムコマンド

```
setup        IMAGE1  R.   STB  256  256
対応する属性  NE      NE   ES   ES   ES
```

引数NO.	引数	属性	処理モジュールへ渡される値
1	IMAGE1	NE	文字列"IMAGE1"
2	R.	NE	文字列"R."
3	STB	ES	変数 STB の値
4	256	ES	整数 256
5	256	ES	整数 256

・正しくは，処理モジュールへ送られるものはデータ本体ではなくポインタである。

引数	タイプ	システムの評価
VALUE	1	変数 VALUE のデータ
"VALUE"	2	文字列"VALUE"
256	0	整数256
256. 256. 0	0	実数256. 0
"256"	2	文字列"256"
VALUE. FX	0	変数 VALUE の FX 属性
VALUE. FN	0	変数 VALUE の FN 属性 (フレーム名)
12E5	−	受け付けない
5*4	−	受け付けない

(ES 属性の場合)

FIG 9.　変数の評価

エラー検出には処理モジュール内部に関する情報が必要であるが，システムはそれをインストラクションリストを解読することによって得る。

システムは，インストラクションリストの指示に従って必要な領域量を計算し，これが変数の領域と異なると領域不整合のエラーを出力する。また，システムは変数の最終履歴の属性を見て，変数が参照される前に変更されると入出力のエラーを出力する。

5．システムの実装

5．1　システムのハードウエア

システムは，富士通ＦＡＣＯＭ　Ｍ－１７０　ＯＳ－Ⅳ上で　ＦＯＲＴＲＡＮ７７　を用いて開発した。デバイスハンドラとして，ＳＩＤＢＡ　デコーダ，イメージプロッセサ　ＮＥＸＵＳ　への画像データ転送サブシステムを作成した。ＳＩＤＢＡ　デコーダは，ＳＩＤＢＡ　のサブフレームの制約をうけずに，自由に部分画像を切り出し，読み込むことが可能である。ＮＥＸＵＳ　へのデータ転送には，ホストとの通信オーヘッドを減少させるためにデータの圧縮（パック）・転送・復元（アンパック）を一元的に管理し高速化をはかった。ＮＥＸＵＳ　は表示デバイスとともに，多機能な画像処理機能を備えておりこれを活用するために　ＮＥＸＵＳ　コマンドルーチンも設けた。

画像処理モジュールライブラリには，ＳＰＩＤＥＲを用いた。処理モジュールの登録には，ソースリストアナライザを用い，登録した，処理モジュール数は，約４００個である。

5．2　処理モジュールの登録

・インストラクションリストジェネレータ

システムは外部処理モジュールの登録のよって稼働することを前提にしている。しかしシステムに新しく処理モジュールを登録するためには，インストラクションリストを作成する必要がある。画像処理ソフトウェアパッケージのように，多くの処理モジュールをまとめて登録する場合のために，インストラクションリストを自動生成するプログラムを作成した。これは，標準的なＦＯＲＴＲＡＮ７７で書かれたサブルーチンのソースリストから，そのコールングシーケンスを解析しインストラクションリストを生成する。

プログラムはソースリストアナライザとインストラクションリストリストジェネレータからなる。ソースリストアナライザは登録すべきサブルーチンモジュールのソースプログラムを解析して，その仕様（コーリング・シーケンス）を自動的に求める。サブルーチンの各パラメータごとに必要な容量，サブルーチン内で値が変更されるかが調べられ，パラメータ間の関係が求められる。

ついでインストラクションリストジェネレータがこの情報をもとにインストラクションリストを自動生成する。

ＳＰＩＤＥＲ　のように数百あるサブルーチンライブラリをまとめて登録する場合には有効である。

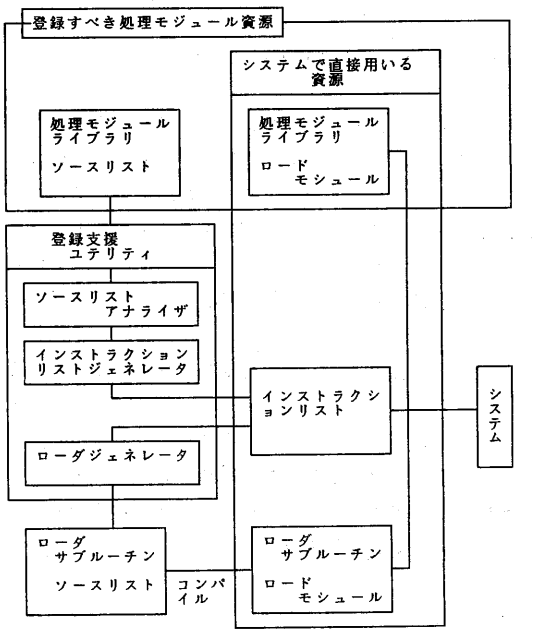

FIG 10. 処理モジュールの登録

5．3　システムの移植

移植性を考え，システムはすべてＦＯＲＴＲＡＮ７７で作成された．機械に依存する部分は注意ぶかく一つにまとめられている．プログラムに要した行数は．システム本体で約４０００行，ＮＥＸＵＳドライバ．ＳＩＤＢＡドライバ．および他の支援ツーツリティで約４０００行である。

変数用の領域量は．実用的な処理のためには．ほぼ１－２ＭＢが必要である。

6．結論

本システムは異なる画像処理サブシステム上にまたがるシェルとして機能し，既存のデバイス・処理モジュールを統合して，統一した対話的環境を提供する画像処理開発支援ツールである。システムは，以下の点に重点をおいて設計された。

1．　汎用性・移植性を考え，各画像処理デバイスを仮想化し，システムから独立させる。

2．　対象とする処理モジュールはサブルーチンロードモジュールを想定した。処理モジュールの外部仕様および内部の情報は，インストラクションリストによってあたえられる。このため多くの処理モジュールはそのコーリングシーケンスを書き換えることなくシステムに登録可能である。

3．　画像データ・スカラーデータとも一貫した，変数リストで扱い，変数の階層的定義を用いてサブフレームも統一的に扱う。

4．　システムの殆どをＦＯＲＴＲＡＮ７７で作成し，移植性に考慮をはらった

システム上からは，各デバイス・処理モジュールの起動は，対話的，インタープリティブな環境で行うことができる。外部からは，コマンド・変数・定数は，みな同等な文字列シンボルであり，システム上から処理モジュールを用いたマクロなプログラミングは純粋な文字列操作・記号処理に帰着される。したがって本システムの上により高度で不変的なシステムを展開していくことが容易と考えられる。

参考文献

1．田村秀行他　ポータブル画像処理ソフトウェアパッケージＳＰＩＤＥＲの開発　情報処理論文誌ＶＯＬ２１．ＮＯ３（１９８２）

2．高木幹雄他　ミニコンピュータによる対話型画像処理ソフトウェアシステム　電子通信学会　技術報告画像工学研究会資料

3．坂上勝彦他　処理モジュールの構造的知識を利用した画像処理プログラム自動生成システム　情報処理論文誌ＶＯＬ２６ＮＯ４（１９８５）

4．谷口倫一郎他　画像処理支援システムＩＰＳＳＥＮＳの開発　情報処理論文誌ＶＯＬ２６ＮＯ６（１９８５）

5．R.M.Haralick KANDIDATS An Interactive Image Processing System　COMPUTER GRAPHICS AND IMAGE PROCESSING 8, 1-15(1978)