VPシリーズ互換
画像認識ライブラリ

SoftVP

Fine Vision Processor

ユーザーズマニュアル

HIRECON

はじめに

このたびは、ＶＰシリーズ互換画像認識ライブラリ　Ｓｏｆｔ ＶＰをお買い上げいただきまして、誠にありがとうございます。

本マニュアルは、ＶＰシリーズ製品を使用したアプリケーション作成のためのシミュレータソフトウェア、「Ｓｏｆｔ ＶＰ」について記載しております。ハードウェアおよびアプリケーション作成のための基本ソフトウェアについては、各ＶＰシリーズ製品のハードウェアマニュアルおよびＳＤＫのマニュアルをご参照ください。

⚠ ご注意

●システムの構築やプログラム作成などの操作を行う前に、本マニュアルの記載内容をよく読み、書かれている指示や注意を十分理解して下さい。誤った操作によりシステムの故障が発生することがあります。

●本マニュアルの記載内容について疑問点または不明点がございましたら、弊社営業窓口までお知らせください。
また、弊社ホームページのお問い合わせのページからも受け付けておりますのでご利用ください。
http://www.systemtech.maxell.co.jp/solution/vp/

●お客様の誤った操作に起因する、事故発生や損害につきましては、弊社は責任を負いかねますのでご了承ください。

●弊社提供のハードウェアおよびソフトウェアを無断で改造しないでください。この場合の品質および安全につきましては、弊社は責任を負いかねますのでご了承ください。

― 目次 ―

第1章 製品概要

1.1 SoftVPの概要

本ＳｏｆｔＶＰは、ＶＰ－Ａｘシリーズの画像処理機能を使用した画像認識アプリケーションの開発を行うための開発キットです。パソコンにてＶＰ－Ａｘシリーズの画像認識コマンドをソフト的にシミュレートすることにより画像認識アプリケーションのデバッグ、評価をＷｉｎｄｏｗｓ上で行うことができます。

1.1.1 特徴

・ＶＰ－Ａｘシリーズの画像認識ライブラリと機能的に互換性があります。
・Windowsパソコン上でMicrosoft VisualStudio C/C++ を使用しコンパイル、デバッグが可能です。
・映像ファイルからの映像入力をサポートしています。

1.1.2 動作環境

本ＳｏｆｔＶＰを使用した画像認識アプリケーション開発には、以下の環境が必要です。

表1-1-1 Ｗｉｎｄｏｗｓ上のＶｉｓｕａｌ Ｃ／Ｃ＋＋環境でのアプリケーション開発環境

項目		内容
ハードウェア	パソコン	Ｗｉｎｄｏｗｓ７またはＷｉｎｄｏｗｓＸＰが動作するパソコン メモリ容量：１Ｇバイト以上推奨 ＣＤ－ＲＯＭドライブ×１、ＵＳＢインタフェース×１
ＯＳ		Ｍｉｃｒｏｓｏｆｔ Ｗｉｎｄｏｗｓ７（ 日本語 ） （ＳｅｒｖｉｃｅＰａｃｋ１以上）（３２ビット版）
		Ｍｉｃｒｏｓｏｆｔ ＷｉｎｄｏｗｓＸＰ（ 日本語 ） （ＳｅｒｖｉｃｅＰａｃｋ２以上）
コンパイラ		Ｍｉｃｒｏｓｏｆｔ Ｖｉｓｕａｌ Ｓｔｕｄｉｏ ２０１０（ 日本語 ） （Ｖｉｓｕａｌ Ｃ/Ｃ++／ＳｅｒｖｉｃｅＰａｃｋ１以上）
		Ｍｉｃｒｏｓｏｆｔ Ｖｉｓｕａｌ Ｓｔｕｄｉｏ ２００５（ 日本語 ） （Ｖｉｓｕａｌ Ｃ/Ｃ++／ＳｅｒｖｉｃｅＰａｃｋ１以上）
インストールディスク容量		約３００ＭＢ

・ＳｏｆｔＶＰはマルチプロセス、マルチスレッドのアプリケーションには対応していません。

1.1.3 SoftVPの構成

Ｗｉｎｄｏｗｓ上のＳｏｆｔＶＰは、下の図に示すように、ＤＬＬ（Dynamic Link Library）と映像入力と画像メモリ表示ツールで構成されます。

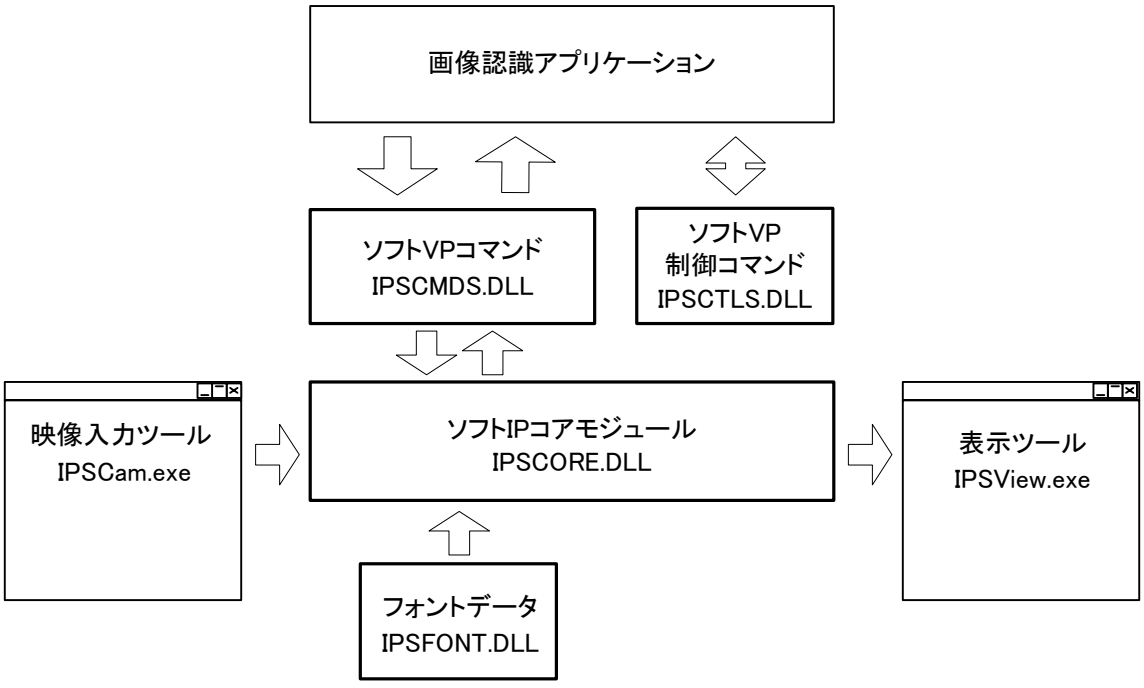


図1-1-1 ＳｏｆｔＶＰの構成

第2章 パッケージ内容

本ＳｏｆｔＶＰは以下の内容を提供します。

・ＳｏｆｔＶＰコマンドライブラリ（Ｌｉｂ、ＤＬＬ、ヘッダファイル含む）
・映像入力、画像メモリ表示ツール

2.1 SoftVP コマンドライブラリ

ＳｏｆｔＶＰ画像認識ライブラリは、Ｍｉｃｒｏｓｏｆｔ　Ｖｉｓｕａｌ　C/C++用のライブラリです。このライブラリは、ＬｉｂファイルとＤＬＬ（Dynamic Link Library）で提供され、Ｌｉｂファイルをアプリケーションとリンクすることでアプリケーションを構築できます。

2.2 インストール内容

ＳｏｆｔＶＰのインストールで以下の内容のファイルがインストールされます。

(1) フォルダ構成

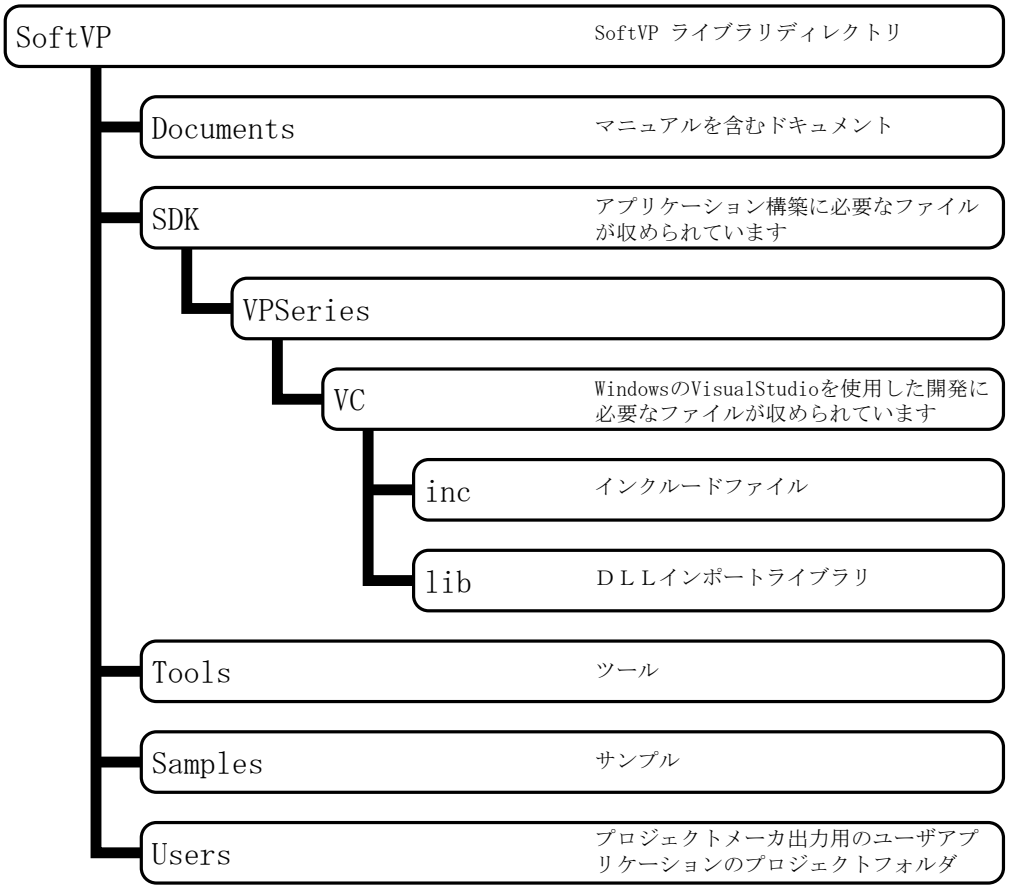


図2-2-1 ＳｏｆｔＶＰのフォルダ構成

(2) モジュール構成

SoftVPライブラリで提供する主なモジュールを以下に示します。

表2-1-1　SoftVPの主なモジュール

項目	ファイル	内容
ヘッダーファイル	IPXDEF.H	画像認識ライブラリ用ヘッダファイル
	IPXSYS.H	画像認識ライブラリ用ヘッダファイル
	IPXPROT.H	画像認識ライブラリ用ヘッダファイル
ライブラリファイル	IPSCMDS.LIB	SoftVPコマンドライブラリ
	IPSCTLS.LIB	SoftVP制御コマンドライブラリ
ＤＬＬ	IPSCMDS.DLL	SoftVPコマンドライブラリＤＬＬファイル
	IPSCTLS.DLL	SoftVP制御コマンドライブラリＤＬＬファイル
	IPSCORE.DLL	SoftVPコアモジュールＤＬＬファイル
	IPSFONT.DLL	SoftVPフォントデータＤＬＬファイル
ツール	IPSCam.exe	映像入力ツール
	IPSView.exe	画像メモリ表示ツール

第3章　SoftVPの使用方法

3.1 プロジェクトメーカー

ＳｏｆｔＶＰで用意しているプロジェクトメーカーを使用することで、ＶｉｓｕａｌＳｔｕｄｉｏ用のプロジェクトを簡単に作成することができます。また、プロジェクトメーカーにより、画像認識ライブラリを使用して画像認識アプリケーションを作成する際のメインプログラムのテンプレートとして、以下の「main.c」または「main.cpp」のソースコードも同時に作成します。プロジェクトメーカーの詳細は「ProjectMaker操作説明書.pdf」を参照して下さい。なお、プロジェクトメーカーを使用しないでプロジェクトを作成する場合は、「付録Ａ　コンパイラの設定」を参照しプロジェクトを作成してください。

```
#include <stdio.h>
#include "ipxdef.h"
#include "ipxsys.h"
#include "ipxprot.h"

void main( )
{
   int ret;
   ConfigDispPara cfgdisp;
   DispPaletteTbl PalTbl;
   ConfigVfwPara cfgvfw;
   int i;

   /* 画像処理システムの初期化  */
   ret = InitIP();

   /* 映像表示設定              */
   memset( &cfgdisp, 0, sizeof( ConfigDispPara ) );
   cfgdisp.xsize          = 640;
   cfgdisp.ysize          = 480;
   cfgdisp.dpx            = 0;
   cfgdisp.dpy            = 0;
   cfgdisp.disp_plane     = DISP_PLANE_6;
   cfgdisp.overlay_plane  = DISP_PLANE_5;
   cfgdisp.overlay_pallet = DISP_PALLET_1;

   ret = SetConfigDisp( &cfgdisp );

   memset( &PalTbl, 0, sizeof( PalTbl ) );
   for(i=0; i< 17; i++){
      PalTbl.Color[i].Alpha = lut01[i].Alpha;
      PalTbl.Color[i].Red   = lut01[i].Red;
      PalTbl.Color[i].Green = lut01[i].Green;
      PalTbl.Color[i].Blue  = lut01[i].Blue;
   }

   ret = SetDispPalette( DISP_PALLET_1, &PalTbl);

   /* 画像メモリ領域確保 */
   for( i=0; i<4; i++ ){
      gImgID[i] = AllocImg( IMG_FS_256H_256V );
   }

   /* 映像入力設定 */
   memset( &cfgvfw, 0, sizeof( ConfigVfwPara ) );
   cfgvfw.xlng            = 640;
   cfgvfw.ylng            = 480;
   ret = SetConfigCamera( CFG_BMP_LIST, &cfgvfw, sizeof( ConfigVfwPara ) );

   /* ここからユーザプログラムを記述してください */




}
```

図3-1-1　main関数のテンプレート出力例

3.2 コーディング

3.2.1 インクルードファイルの記述

Ｗｉｎｄｏｗｓアプリケーションやターゲットボードで動作するダウンロードモジュールアプリケーションを作成する場合、以下に示すインクルードファイルを以下の順番でプログラムにインクルードしてください。

表3-1-1 インクルードファイル一覧

ファイル名	内容	インクルードする順番
ipxdef.h	define,enum定義	1
ipxsys.h	構造体定義	2
ipxprot.h	プロトタイプ定義	3

3.2.2 画像処理ライブラリの初期化

画像処理ライブラリを使用する場合は、画像処理システムの初期化（InitIP）を実行し初期化します。その後ユーザアプリケーションを実行してください。

```
#include <stdio.h>
#include "ipxdef.h"
#include "ipxsys.h"
#include "ipxprot.h"

void main( )
{
    int  ImgID1,ImgID2,ImgID3;

    /* 画像処理システムの初期化 */
    InitIP( );

    /* ユーザプログラム  */
    ActiveVideoPort( VIDEO_PORT0 );
    SelectCamera( CAMERA_PORT0, BW_CAMERA );
    SetVideoFrame( INTERLACE, VIDEO_FS_512H_480V );
    ImgID1 = AllocImg( IMG_FS_512H_512V );
    ImgID2 = AllocImg( IMG_FS_512H_512V );
    ImgID3 = AllocImg( IMG_FS_512H_512V );
    GetCamera( ImgID1 );
    IP_AddConst( ImgID1, ImgID2, 20 );
    IP_Binarize( ImgID2, ImgID3, 120 );
}
```

このインクルードファイルは必ずインクルードして下さい

画像処理システムの初期化

図3-2-1　コーディング例

3.3 VP-Axシリーズ画像認識ライブラリとの相違点

本ＳｏｆｔＶＰは、ＶＰ－Ａｘシリーズ画像認識ライブラリとＣ言語インタフェースで互換性を持った画像処理コマンドを実装していますが、映像入出力、画像メモリの容量や処理時間は、ユーザーのターゲットシステムの環境に依存します。また、画像認識ライブラリ内部でのフローティング演算での丸め誤差、一部のＳＨ標準関数ＬＩＢとＶｉｓｕａｌＣ標準関数ＬＩＢとの微妙な差異で処理結果が全く同じにならない場合があります。

以下に機能的な相違点を示します。

※ダミー関数：何も処理せず、戻り値が常に正常終了の関数

3.3.1 API初期化

・StartIP ･･･ ダミー関数
・StopIP ･･･ ダミー関数
・ActiveIP ･･･ ダミー関数
・OpenIPDev ･･･ ダミー関数（戻り値：１が返される）
・CloseIPDev ･･･ ダミー関数
・EnumAttachIPDev ･･･ ダミー関数（戻り値：１、ボード番号：０が返される）
・ResetIP ･･･ ダミー関数
・GetIPLastError ･･･ ダミー関数

3.3.2 映像入力

映像入力ツール：IPSCamからの映像入力をサポートします。

・ActiveVideoPort ･･･ ダミー関数
・SetVideoFrame ･･･ ダミー関数
・SelectCamera ･･･ ダミー関数
・GetCamera ･･･ ＵＳＢカメラまたは動画ファイル(AVI)からの映像入力
・SetVFDelay ･･･ ダミー関数
・SetTriggerMode ･･･ ダミー関数
・SetStrobeMode ･･･ ダミー関数
・SetPartialMode ･･･ ダミー関数
・SetVideoLUTMode ･･･ ダミー関数
・GetSceneCounter ･･･ ダミー関数
・SetSceneCounter ･･･ ダミー関数
・SetCameraData ･･･ ダミー関数
・GetCameraData ･･･ ダミー関数
・LoadCameraFile ･･･ ダミー関数
・SaveCameraFile ･･･ ダミー関数
・CaptureContinuous ･･･ ダミー関数
・StopCaptureContinuous ･･･ ダミー関数
・GetCameraReqScene ･･･ ダミー関数（GetCameraと同じ動作）
・GetCameraResScene ･･･ ダミー関数（GetCameraと同じ動作）

3.3.3 映像出力

映像表示ツール：IPSViewへの映像表示をサポートします。

・SetConfigDisp ･･･ ダミー関数（SetConfigViewと同機能(IPSView起動)になります）
・SetDispPalette ･･･ ダミー関数（パレットNo.1～No.6は共通になります）
・SelectDisp ･･･ ダミー関数
・DispCamera ･･･ ダミー関数
・DisableDisp ･･･ ダミー関数
・DispImg ･･･ IPSViewに映像を表示します。
・DispOverlap ･･･ IPSViewに表示されている映像にオーバレイします。

3.3.4 パイプライン制御

ＳｏｆｔＶＰではパイプライン制御は動作しません。

・EnablePipeline ･･･ ダミー関数
・DisablePipeline ･･･ ダミー関数

3.3.5 2値画像形状変換

・IP_Shrink4 ･･･ IP_Shrink8と同じ動作になります。

3.3.6 コマンドエラー発生時の動作

本Ｓｏｆｔ ＶＰの画像処理コマンドドライバでは、コマンドエラー発生時、エラー情報のメッセージボックスを表示する機能がありますが、デフォルトでメッセージ出力が無効になっています。

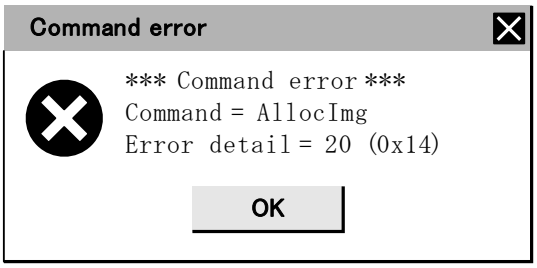


図3-3-1　エラー情報のメッセージボックス

3.3.7 その他 VP-Axシリーズの制御コマンド

その他のＶＰ－Ａｘシリーズの制御コマンドのサポート状況を以下に示します。
なお、下記ダミー関数を使用する場合は、IPSCtls.libをリンクする必要があります。

(1) システム制御コマンド

下記コマンドは、ダミー関数で実装しています。

・GetIPVersionInfo

(2) ダウンロードモジュール制御

ダミー関数で実装しています。

(3) ボード制御

下記コマンド以外は、ダミー関数で実装しています。

・GetPerformanceFrequency ･･･　WindowsAPI QueryPerformanceFrequencyをラップ
・GetPerformanceCounter ･･･　WindowsAPI QueryPerformanceCounterをラップ
・IPSleep ･･･　WindowsAPI Sleepをラップ
・SetIPTime ･･･　WindowsAPI SetLocalTimeをラップ
・GetIPTime ･･･　WindowsAPI GetLocalTimeをラップ

(4) ファイルアクセス

実装していません。

(5) μITRONサービスコール

実装していません。

(6) Windowsアプリケーションサポートコマンド

ダミー関数で実装しています。

(7) イーサネット通信コマンド

実装していません。

3.3.8 画像メモリ領域管理

画像メモリの最大確保数が異なります。

ＶＰ－Ａｘシリーズ：画像メモリ及びシステムメモリが１２８ＭＢの場合は最大２５６面、
２５６ＭＢの場合は最大５１２面以上

Ｓｏｆｔ ＶＰ　　：最大２５６面

第4章 画像処理の概要

4.1 画像処理コマンドの構成

画像間演算、空間フィルタリングなどの画像処理コマンドは、サブルーチン形式のコマンドとしてＣ言語で利用できます。

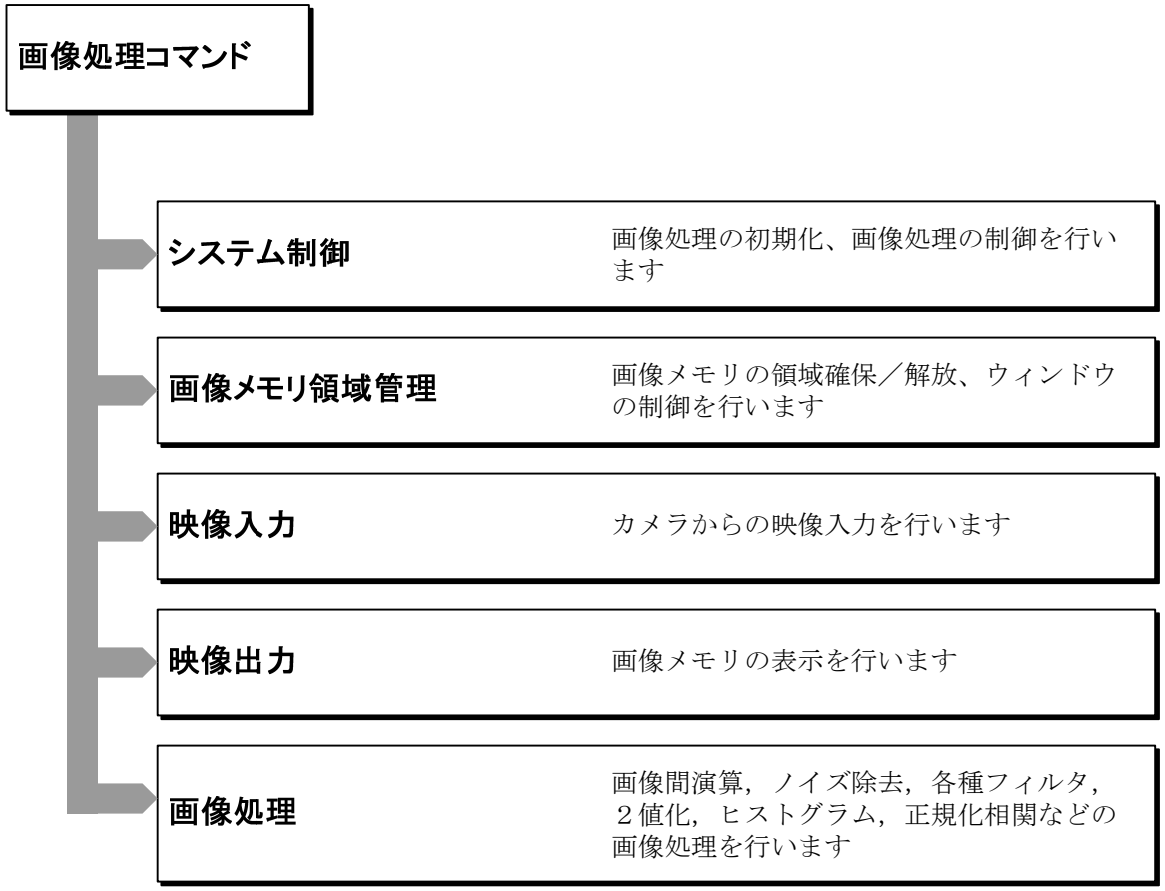


図4-1-1 画像処理コマンドの構成

4.2 システム制御

システム制御コマンドは、システムの初期化やエラー情報読み出し等、システム全体を制御します。

4.2.1 システムの初期化

システムの初期化は『InitIP()』コマンドにより行います。
以下にシステムの初期化のフローを示します。

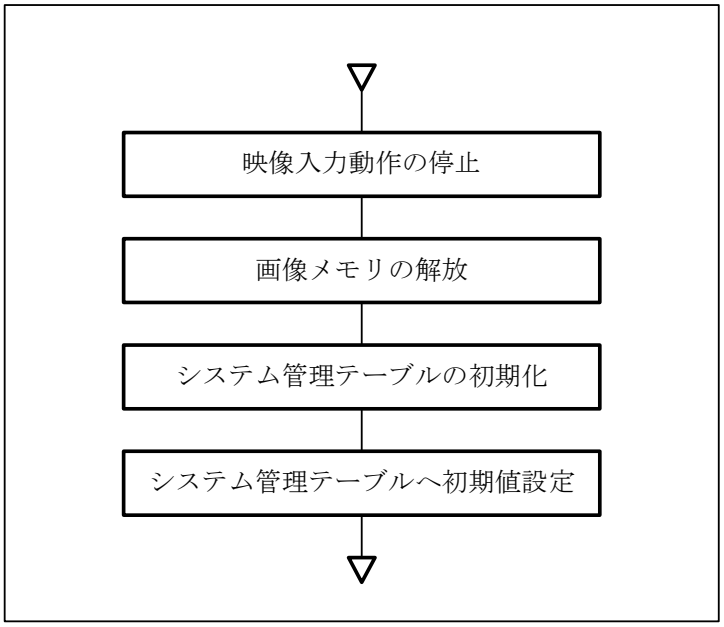


図4-2-1　システムの初期化

『InitIP』コマンドにより初期設定される項目と内容を以下に示します。本コマンドでは、有効ビデオポート番号設定、ビデオフレーム設定、カメラタイプの設定、カメラ映像ライブ表示は行いません。

表4-2-1　初期設定値

設定項目	設定内容
システムデータタイプ	符号なし８ビット
ウィンドウ（全種）	有効（５１２（X）×４８０（Y））
文字描画属性（サイズ）	標準サイズ（１６×１６）
文字描画属性（文字間隔）	０ドット
文字描画属性（行間隔）	０ドット
正規化相関マスク	無効
正規化相関途中演算打ち切り	無効
正規化相関打ち切りしきい値	0
パイプラインモード	解除

4.2.2 エラー情報管理

画像処理コマンドでエラーが発生すると、コマンドのリターンコードにはエラーコードが返されますが、詳細エラー情報はリターンコードに反映されずシステムに登録されます。また、エラー発生後の画像処理は、エラー情報をクリアしない限り全て実行することができません。そのため、ユーザは『ReadIPErrorTable』コマンドでエラー情報を読み出すか、または『ClearIPError』コマンドでエラー情報をクリアする必要があります。

また、『EnableIPErrorMessage』でエラーメッセージ表示を有効にすると画像認識ライブラリの画像処理コマンドでエラーが発生した場合、エラー情報のダイアログボックスが表示され、ＯＫボタンを押すまで処理が中断します。このエラー情報のダイアログボックスは『DisableIPErrorMessage』及び『EnableIPErrorMessage』で出力を制御できます。

以下に画像処理コマンドでエラーが発生した際の処理手順を示します。

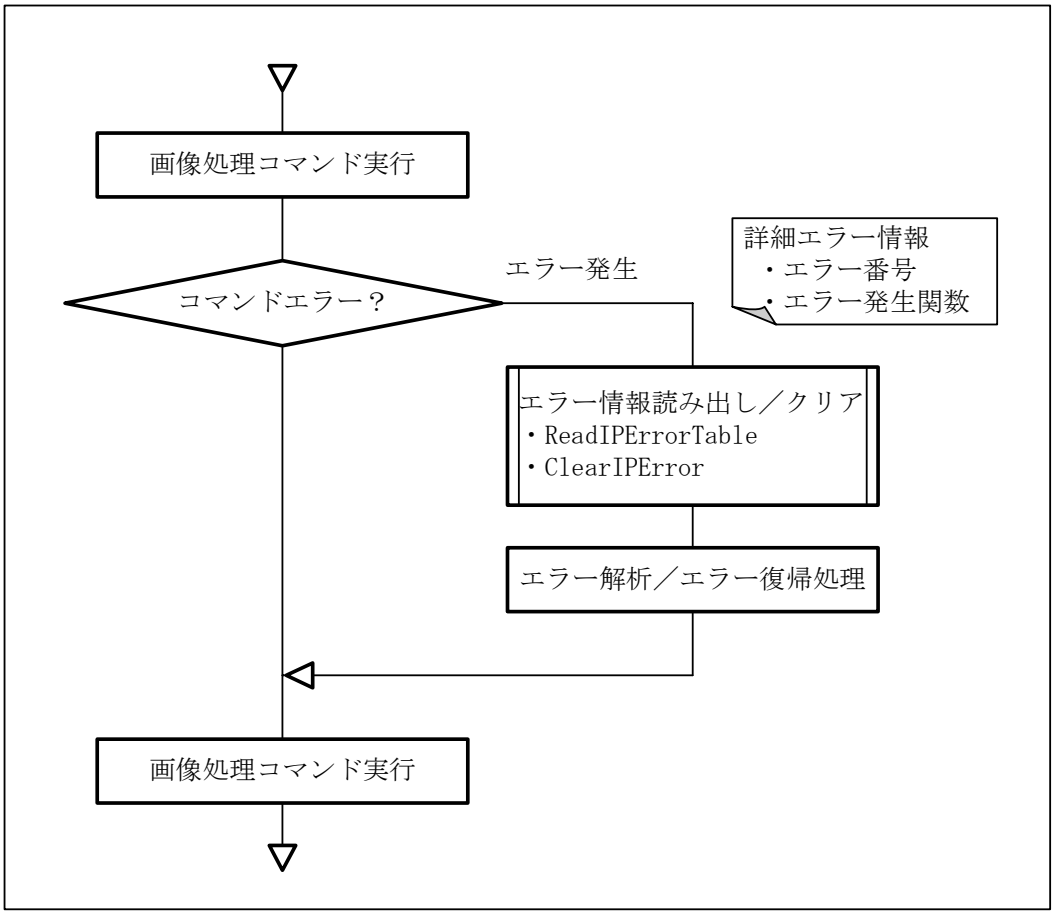


図4-2-2　コマンドエラー処理手順

ＳｏｆｔＶＰは、『InitIP()』コマンド、および『InitIPExt()』コマンドでＳＲＭ（ソフトウェア著作権管理）エラーチェックを行います。ＳＲＭ エラーは以下の場合に発生し、エラー発生時にはエラーメッセージ表示の有効／無効に関わらずエラー情報ダイアログボックスを表示します。

（1）ＵＳＢ ライセンスキーのドライバ（Sentinel HASP Run-time）がインストールされていない
（2）ＵＳＢ ライセンスキーが装着されていない

ＳｏｆｔＶＰ Ｖ２．００以降は、『DefSRMErrorHdr()』コマンドでＳＲＭエラーチェック関数のエラーハンドラを登録することで、ＳＲＭエラーメッセージを変更することができます。
詳細は、『コマンドリファレンス』の第１章　初期化コマンドを参照ください。

4.3 画像メモリ領域管理

画像メモリを使用するための領域の確保、解放と、ウィンドウの制御を行います。

4.3.1 画像メモリの概要

画像メモリは、カメラ映像の入力データや画像処理演算結果を格納するメモリです。画像メモリは、AllocImg等の画像メモリ確保コマンドでモノクロ画面、同時取り込み用画面、コンポーネントＲＧＢ画面、１６ｂｉｔＲＧＢ画面を確保します。また、８Ｂｉｔのモノクロ画面５１２×５１２画素で２５６面確保できます（システムのメモリ容量により確保できる画面数が制限されます）。

画像メモリには濃淡画像メモリと２値画像メモリの区別はなく、２値画像として使用するときは、黒は０、白は２５５の値で使用します。

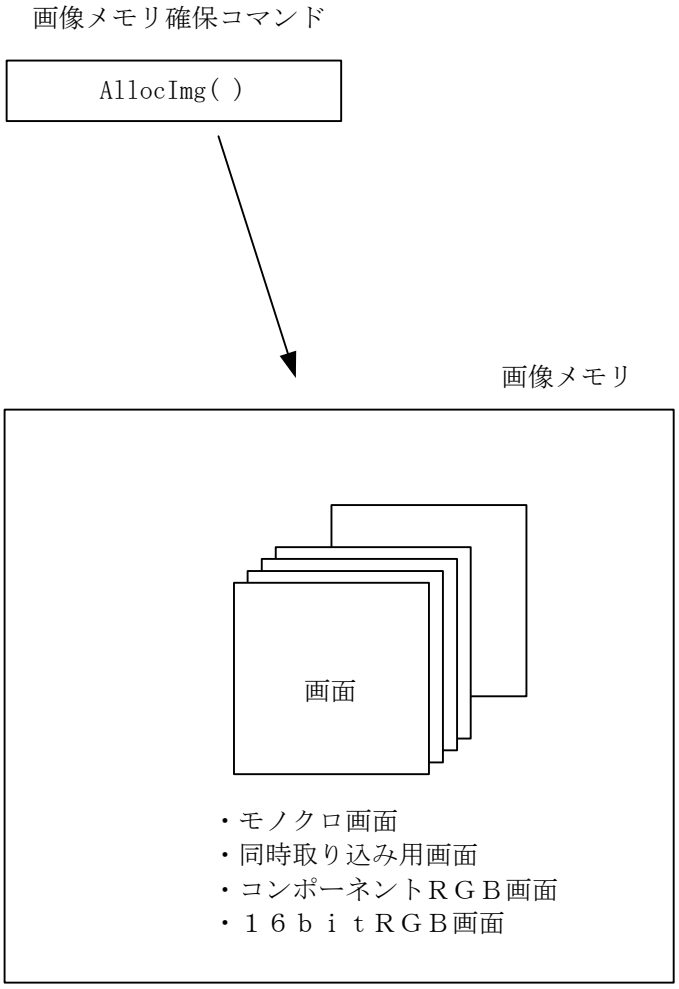
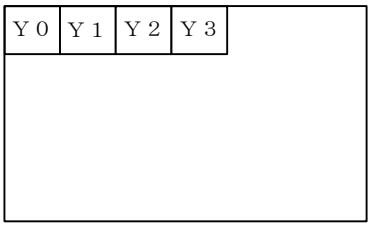
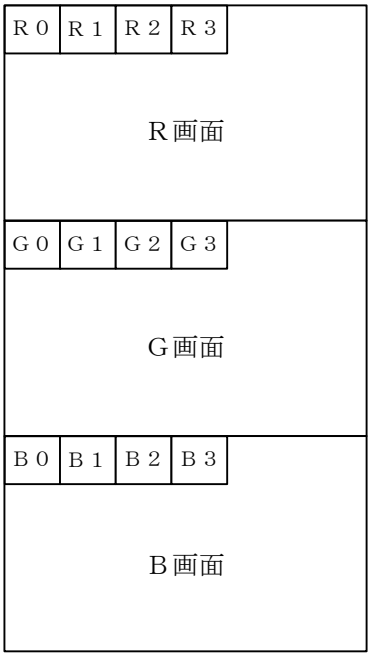
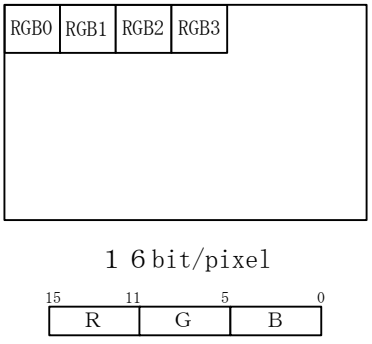


図4-3-1　画像メモリ

4.3.2 画像メモリの画面タイプ

画像処理コマンドでは、モノクロ画面（Ｙ画面）とＲＧＢカラー画面の画像メモリを確保可能です。Ｙ画面は映像入力、画像処理および映像表示に使用できます。コンポーネントＲＧＢ画面は映像入力(※)と画像処理、ＲＧＢ１６画面は映像表示に使用できます。以下に画面タイプの一覧を示します。

表4-4-1　画像メモリの画面タイプ（１）

画面タイプ	フォーマット	画面確保コマンド	詳細
Ｙ	 ８bit/pixel	AllocImg	８bit/pixelの画面。濃淡及び２値の画像処理、モノクロ映像入力表示に使用できます。
コンポーネント ＲＧＢ	 ２４bit/pixel	AllocRGBImg	８bit/pixelのＲ画面、Ｇ画面、Ｂ画面で構成される画面。Ｒ、Ｇ、Ｂの３画面でＲＧＢカラー情報を構成します。ＲＧＢカラーの映像入力(※)に使用できます。映像表示はできません。 Ｒ画面、Ｇ画面、Ｂ画面はそれぞれモノクロ画面として画像処理に使用できます。
ＲＧＢ１６	 １６bit/pixel 	AllocRGB16Img	１６bit/pixelの画面。１６bitデータは上位からＲデータ５bit、Ｇデータ６bit、Ｂデータ５bitで、１画面でＲＧＢカラー情報を構成します。ＲＧＢカラーの映像表示に使用できます。映像入力、画像処理はできません。

4.3.3 画像メモリのサイズ

画像処理に使用できる画面サイズは、以下に示すサイズで画像メモリを確保できます。なお、画像処理ハードウェアの制限により１０２４×１０２４を超える大きさの範囲では画像処理ができないコマンドがありますので注意してください。

表4-4-2　確保容量一覧

論理画面サイズ	物理画面サイズ	備考
	Y画面	
256H×256V	←	
256H×512V	←	
320H×256V	←	
320H×512V	←	
512H×256V	←	
512H×512V	←	
640H×256V	←	
640H×512V	←	
768H×512V	←	
768H×576V	←	
1024H×768V	←	
1024H×1024V	←	
1280H×1024V	←	
1344H×1024V	←	
1600H×1024V	←	
1920H×1080V	←	

4.3.4 画像メモリの座標系

画面の座標系は、画面の左上隅を原点とした下図のような座標系となります。

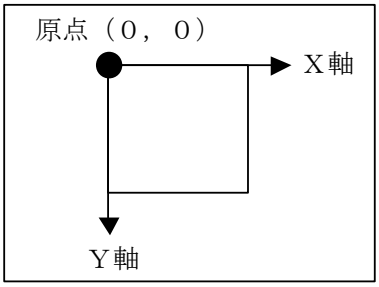


図4-3-2　画像メモリの座標系

4.3.5 画像メモリのデータタイプ

画像メモリに格納されるデータは、濃淡データと２値データの２種類があります。濃淡データは２５６階調を持ち、符号なし８ビット（０～２５５）と符号付８ビット（－１２８～１２７）の２種類があります。基本設定は符号なし８ビットです。

２値データは、黒と白の２階調のデータです。黒は０、白は２５５の値で画像メモリに格納されます。

4.3.6 ウィンドウの概要

ウィンドウとは、処理対象画面、または結果格納画面内の処理領域のことです。画像処理は、画像メモリから確保した画面のサイズ、またはウィンドウのサイズで動作します。よって、画面のサイズより小さいウィンドウを設定することにより、処理を高速に行うことができます。

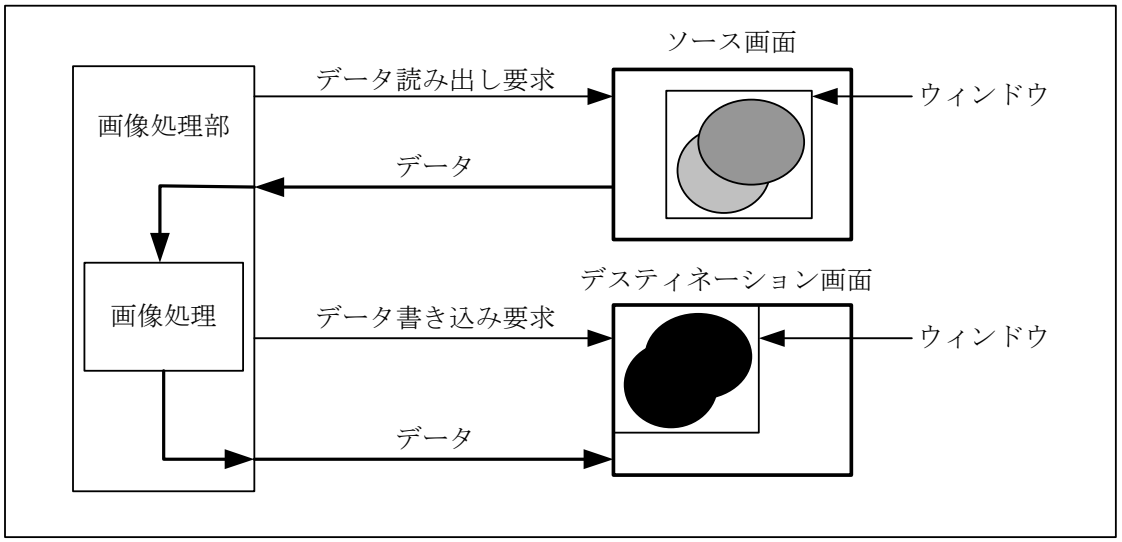


図4-3-3　ウィンドウの役割

4.3.7 ウィンドウの座標系

ウィンドウの座標系は、画面の左上隅を原点とした下図のような座標系となります。ウィンドウ設定値は、画面の原点相対の座標を指定します。また、ヒストグラム等の画像処理はウィンドウで指定した始点を原点として抽出座標を出力します。

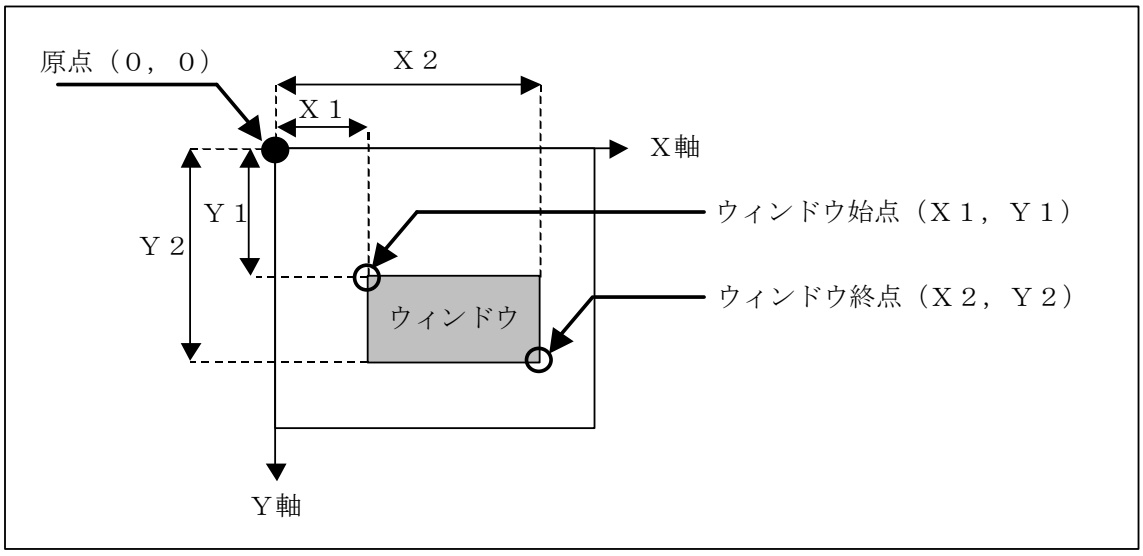


図4-3-4　ウィンドウの座標系

4.3.8 ウィンドウの種類

ウィンドウには、以下６つの種類があります。

表4-3-3　ウィンドウの種類

ウィンドウ種類	用　　途
ソース０画面 (SRC0_WIN)	画像処理でのソース画面に使用します
ソース１画面 (SRC1_WIN)	画像間演算処理でのソース画面に使用します （ソース画面を２画面使用するときの第２ソース画面）
ソース拡張画面 (SRC2_WIN)	画像処理でのソース拡張画面に使用します（未使用）
デスティネーション画面（DST_WIN）	画像処理でのデスティネーション画面に使用します
画像メモリアクセス（SYS_WIN）	画像メモリ制御コマンド（WriteImg, ReadImg, WritePixel,ReadPixel,WritePixelContinue, ReadPixelContinue）で有効なウィンドウ
デスティネーション拡張画面 (DST_EXT_WIN)	画像処理でのデスティネーション拡張画面に使用します （未使用）

ソース画面とデスティネーション画面で異なるサイズのウィンドウを設定した場合、それぞれのウィンドウサイズ内の最小のサイズで処理を行います。

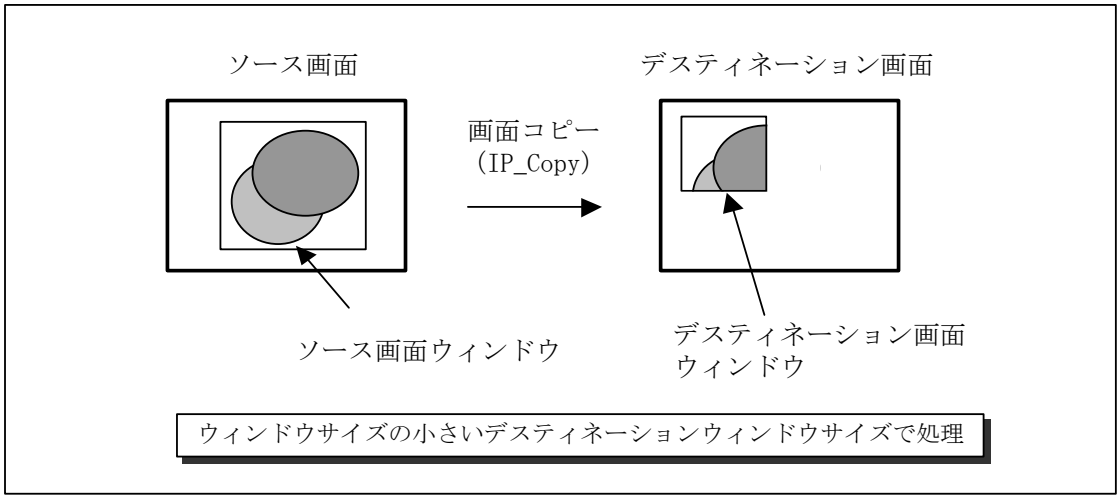


図4-3-5　ウィンドウ処理サイズ

4.3.9 ウィンドウの有効／無効

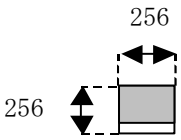
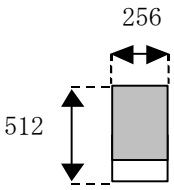
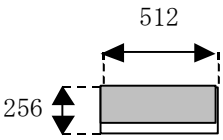
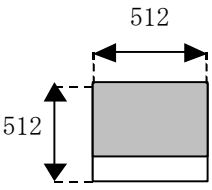
画像処理は、ビデオフレームサイズ、またはウィンドウのサイズで動作します。どちらで動作するかはウィンドウの有効／無効状態で決まり、ウィンドウが無効のときはビデオフレームサイズ、有効のときは設定したウィンドウサイズとなります。この設定は、EnableIPWindowおよびDisableIPWindowコマンドによって行います。

ウィンドウ無効は、DisableIPWindowコマンド、有効はEnableIPWindowコマンドで行います。

ビデオフレームサイズとは、カメラから取り込む映像サイズのことであり、SetVideoFrame コマンドで現在設定されている値です。

下記に、ウィンドウ無効時のウィンドウサイズの例を示します。

表4-3-4　ウィンドウ無効時のウィンドウサイズ例

画面サイズ	２５６×２５６	２５６×５１２	５１２×２５６	５１２×５１２
ビデオ フレーム サイズ	２５６×２２０	２５６×４４０	５１２×２２０	５１２×４４０
画面サイズ ウィンドウ サイズ				

4.3.10 データタイプの属性

本ライブラリでは、映像入出力、画像処理等の処理データを以下のいずれかのタイプであるとみなして処理します。

表4-3-5　データタイプ

データタイプ		値の範囲	オーバーフロー／アンダーフロー
符号付 ８ビット	SIGN8_DATA	－１２８ ～１２７	処理結果がアンダーフローの場合、－１２８（０ｘ８０） 処理結果がオーバーフローの場合、１２７（０ｘ７Ｆ）
符号なし ８ビット	UNSIGN8_DATA	０～２５５	処理結果がアンダーフローの場合、０（０ｘ００） 処理結果がオーバーフローの場合、２５５（０ｘＦＦ）
２値	BINARY_DATA	０または ２５５	なし

※　デフォルト設定は、符号なし８ビット

データタイプの種類として、システムデータタイプと画面データタイプの２種類があります。

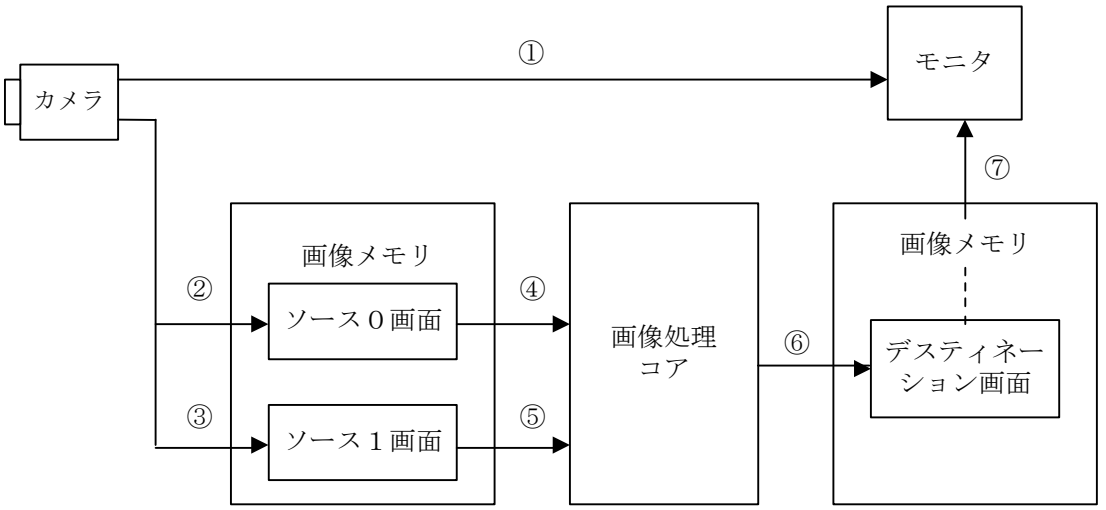
(1) システムデータタイプ

システムとしてのデフォルト設定のデータタイプ。カメラからの映像取り込み、画像処理結果のデスティネーション画面データタイプの設定等に使用されます。
システムデータタイプは、システムイニシャライズ時に符号なし８ビットに設定されますが、SetIPDataType コマンドで変更することも可能です。

(2) 画面データタイプ

画面ごとに設定されているデータタイプで、その画面のデータがどのタイプ（符号付８ビット、符号なし８ビット、２値）であるかを示します。
AllocImgコマンドで確保された画面の画面データタイプは、符号なし８ビットに設定されます。
画面データタイプは、ChangeImgDataType コマンドで変更できます。

以下に、各処理において使用されるデータタイプについて示します。
また、画像処理実行後の処理結果のデータタイプは、コマンドの出力属性により変更されます。
デスティネーション画面のデータタイプを次の表に示します。



①	カメラ映像の表示データタイプに依存しません。
②、③	カメラ映像の取り込み（GetCameraコマンド） システムデータタイプで処理されます。ただし、システムデータタイプが２値に設定されていた場合は、符号なし８ビットで処理します。
④、⑤	画像処理の際のソース画面のデータタイプ ソース画面の画面データタイプで処理します。
⑥	処理結果のデータタイプ 使用したコマンドによって異なります。また、デスティネーション画面のデータタイプは、コマンドの出力属性により変更されます。
⑦	画面表示のデータタイプ（DispImgコマンド） データタイプに依存しません。

図4-3-6　処理の流れとデータタイプ

表4-3-6（1）　デスティネーション画面のデータタイプ一覧

コマンド名	デスティネーション画面データタイプ
IP_ClearAllImg	システムデータタイプ
IP_ClearImg	システムデータタイプ　※１
IP_Const	システムデータタイプ
IP_Copy	ソース０画面データタイプ
IP_Zoom	ソース０画面データタイプ
IP_ZoomExt	ソース０画面データタイプ
IP_ZoomOut	ソース０画面データタイプ
IP_ZoomOutExt	ソース０画面データタイプ
IP_Shift	ソース０画面データタイプ
IP_ZoomS	ソース０画面データタイプ
IP_Rotate	ソース０画面データタイプ
IP_ZoomIn	ソース０画面データタイプ
IP_ZoomInExt	ソース０画面データタイプ
IP_Binarize	２値
IP_BinarizeExt	２値
IP_Invert	システムデータタイプ　※２
IP_Minus	システムデータタイプ
IP_Abs	システムデータタイプ
IP_AddConst	システムデータタイプ
IP_SubCons	システムデータタイプ
IP_SubConstAbs	システムデータタイプ
IP_MultConst	システムデータタイプ
IP_MinConst	システムデータタイプ
IP_MaxConst	システムデータタイプ
IP_ConvertLUT	ソース０画面データタイプ
IP_ShiftDown	ソース０画面データタイプ
IP_ShiftUp	ソース０画面データタイプ
IP_Add	システムデータタイプ
IP_Sub	システムデータタイプ
IP_SubAbs	システムデータタイプ
IP_Comb	システムデータタイプ
IP_CombAbs	システムデータタイプ
IP_Mult	システムデータタイプ
IP_Average	システムデータタイプ
IP_Min	システムデータタイプ
IP_Max	システムデータタイプ
IP_CombDrop	システムデータタイプ
IP_And	ソース０画面データタイプ　※３
IP_Or	ソース０画面データタイプ　※３
IP_Xor	ソース０画面データタイプ　※３
IP_InvertAnd	ソース０画面データタイプ　※３
IP_InvertOr	ソース０画面データタイプ　※３
IP_Xnor	ソース０画面データタイプ　※３

※１　ＲＧＢ＿Ｇ、ＲＧＢ＿Ｂ画面は符号無し８ビット
※２　ソース０画面が２値の場合、２値、それ以外はシステムデータタイプ
※３　ソース０画面が２値の場合、ソース１画面データタイプ、それ以外はソース０画面データタイプ

表4-3-6 (2)　デスティネーション画面のデータタイプ一覧

コマンド名	デスティネーション画面データタイプ
IP_PickNoise4	２値
IP_PickNoise8	２値
IP_Outline4	２値
IP_Outline8	２値
IP_Dilation4	２値
IP_Dilation8	２値
IP_Erosion4	２値
IP_Erosion8	２値
IP_Thin4	２値
IP_Thin8	２値
IP_Shrink4	２値
IP_Shrink8	２値
IP_SmoothFLT	システムデータタイプ
IP_EdgeFLT	システムデータタイプ
IP_EdgeFLTAbs	システムデータタイプ
IP_Lapl4FLT	システムデータタイプ
IP_Lapl8FLT	システムデータタイプ
IP_Lapl4FLTAbs	システムデータタイプ
IP_Lapl8FLTAbs	システムデータタイプ
IP_LineFLT	システムデータタイプ
IP_LineFLTAbs	システムデータタイプ
IP_MinFLT	システムデータタイプ
IP_MinFLT4	システムデータタイプ
IP_MinFLT8	システムデータタイプ
IP_MaxFLT	システムデータタイプ
IP_MaxFLT4	システムデータタイプ
IP_MaxFLT8	システムデータタイプ
IP_LineMinFLT	システムデータタイプ
IP_LineMaxFLT	システムデータタイプ
IP_RankFLT	システムデータタイプ
IP_Rank4FLT	システムデータタイプ
IP_Rank8FLT	システムデータタイプ
IP_MedFLT	システムデータタイプ
IP_Med4FLT	システムデータタイプ
IP_Med8FLT	システムデータタイプ
IP_Label4	符号なし８ビット
IP_Label8	符号なし８ビット
IP_Label4withAreaFLT	符号なし８ビット
IP_Label8withAreaFLT	符号なし８ビット
IP_Label4withAreaFLTSort	符号なし８ビット
IP_Label8withAreaFLTSort	符号なし８ビット
IP_TrsPipelineFLT	２値
IP_BinMatchFLT	システムデータタイプ

4.4 映像入力

4.4.1 映像入力の概要

ＳｏｆｔＶＰは、映像入力にＵＳＢカメラ、動画ファイル（ＡＶＩファイル）、ビットマップファイルからの映像キャプチャをサポートします。

ＳｏｆｔＶＰでサポートする映像入力は、１カメラのフルフレーム取込でGetCameraによる単一映像入力だけをサポートし、連続映像入力、プリフェッチ映像入力、２カメラ以上の同時入力はサポートしていません。また、ビデオポートの選択はできません。

○　サポート
×　未サポート

表4-4-1　映像入力コマンドのサポート状況

No	関数名	機能	サポート	備考
1	ActiveVideoPort	ビデオポート選択	×	ダミー関数
2	SetVideoFrame	映像入力画面設定	×	ダミー関数
3	SelectCamera	カメラ番号とカメラタイプの設定	×	ダミー関数
4	GetCamera	カメラ映像入力	○	
5	SetVFDelay	映像画面の遅延サイズ設定	×	ダミー関数
6	SetTriggerMode	カメラ映像入力トリガモードの設定	×	ダミー関数
7	SetStrobeMode	カメラ映像入力ストロボの設定	×	ダミー関数
8	SetPartialMode	カメラ映像入力パーシャルモードの設定	×	ダミー関数
9	SetVideoLUTMode	ビデオ入力ＬＵＴの設定	×	ダミー関数
10	GetSceneCounter	連続入力シーン番号カウンタ取得	×	ダミー関数
11	SetSceneCounter	連続入力シーン番号カウンタ設定	×	ダミー関数
12	SetCameraData	カメラデータの設定	×	ダミー関数
13	GetCameraData	カメラデータの取得	×	ダミー関数
14	LoadCameraFile	カメラデータファイルのロード	×	ダミー関数
15	CaptureContinuous	連続映像入力モードの設定	×	ダミー関数
16	StopCaptureContinuous	連続キャプチャ停止	×	ダミー関数
17	GetCameraReqScene	連続入力シーン番号指定付きカメラ映像入力	×	SoftVPのGetCameraと同じ動作
18	GetCameraResScene	連続入力シーン番号出力付きカメラ映像入力	×	SoftVPのGetCameraと同じ動作

4.4.2 IPSCamからの映像入力

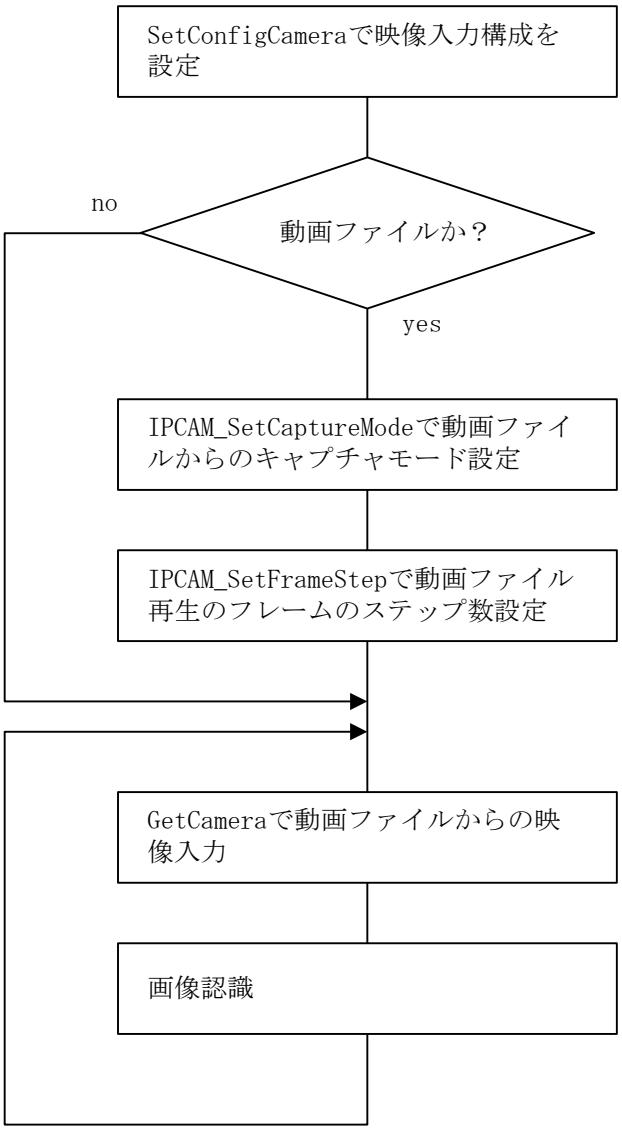
SetConfigCameraで映像入力構成を設定することで、「IPSCam.exe」アプリケーションが起動され、これによりGetCameraでの映像入力ができます。

表4-4-2　構成制御コマンド(映像入力系)

No	関数名	機能	備考
1	SetConfigCamera	映像入力構成制御設定	
2	ExitIPCamera	カメラ映像入力プログラム終了	
3	ReadConfigCamera	映像入力構成制御テーブル読み出し	
4	IPCAM_SetCaptureMode	動画ファイルからのキャプチャモード設定	動画ファイル、ビットマップファイルキャプチャのみ対応
5	IPCAM_GetCaptureMode	動画ファイルからのキャプチャモード取得	
6	IPCAM_SetFrameStep	フレームのステップ数設定	
7	IPCAM_GetFrameStep	フレームのステップ数取得	
8	IPCAM_SetFrameNo	動画ファイルの指定フレームへのシーク	
9	IPCAM_GetFrameNo	動画ファイルのカレントフレーム番号取得	

4.4.3 動画ファイルからの映像入力

以下に映像入力手順を示します。動画ファイルからの映像入力の場合、画像認識時間がフレームレートよりも遅い場合で映像入力の間引き処理が必要な場合は、IPCAM_SetCaptureModeで動画ファイルからのキャプチャモードを「IPCAM_CaptureStep」に設定し、「IPCAM_SetFrameStep」で動画ファイル再生のフレームのステップ数で調整してください。キャプチャモードの「IPCAM_CaptureSeek」設定や「IPCAM_SetFrameNo」でシークすると一定時間間隔でのキャプチャが安定しない場合があります。



通常は、「IPCAM_CaptureStep」を指定して下さい。なお、設定しない場合デフォルトで「IPCAM_CaptureStep」になります。

画像認識時間がフレームレートよりも遅い場合、で間引き処理が必要な場合、ステップ数に１より大きい値を設定して間引き処理を実施してください。なお、設定しない場合デフォルトで「１」になります。

4.5 映像出力

4.5.1 映像出力の概要

ＳｏｆｔＶＰは、Ｗｉｎｄｏｗｓ上の画面への画像メモリ表示をサポートします。ＳｏｆｔＶＰでサポートする映像表示コマンドを下表に示します。なお、カメラライブ映像表示はできません。

○　サポート
×　未サポート

表4-5-1　表示コマンドのサポート状況

No	関数名	機能	サポート	備考
1	SetConfigDisp	表示画面の構成制御設定	○	SetConfigViewと同じ動作
2	SetDispPalette	表示パレットの設定	○	パレットは共通
3	SelectDisp	映像表示選択	×	ダミー関数
4	DispCamera	カメラ映像表	×	ダミー関数
5	DisableDisp	カメラ映像表示終了	×	ダミー関数
6	DispImg	画像メモリ表示	○	
7	NoDisp	表示中止	○	
8	DispOverlap	画像メモリのオーバレイ表示	○	
9	DisableOverlap	画像メモリのオーバレイ表示終了	○	
10	CombineRGB CombineRGBEx	コンポーネントRGB画面から16bitRGB画面の合成	○	

4.5.2 IPSViewへの映像表示

SetConfigDisp、SetConfigViewで画像メモリ表示構成を設定することで「IPSView.exe」アプリケーションが起動され、DispImgを実行することで指定した画像メモリを表示、DispOverlap実行で指定した画像のオーバーレイ表示ができます。

表4-5-2　構成制御(映像表示系)コマンド

No	関数名	機能	備考
1	SetConfigView	画像メモリ表示構成制御設定	
2	ExitIPView	画像メモリ表示プログラム終了	

第5章　画像処理コマンドの概要

画像処理は、機能ごとに下記のように分類されます。

（１）アフィン変換
（２）２値化
（３）濃度変換
（４）画像間算術演算
（５）画像間論理演算
（６）２値画像形状変換
（７）コンボリューション
（８）ランクフィルタ
（９）ラベリング
（１０）ヒストグラム
（１１）画像メモリアクセス
（１２）２値パイプラインフィルタ
（１３）パイプライン制御（ＳｏｆｔＶＰでは使用できません）
（１４）２値マッチングフィルタ
（１５）正規化相関
（１６）グラフィックス
（１７）線分化
（１８）穴埋め
（１９）ＶＰ－９１０Ａ互換
（２０）直線抽出
（２１）イメージキャリパ
（２２）エッジファインダ

5.1 画像処理コマンドの概要

画像処理は、処理対象画像データの画像メモリからの読み出し、画像処理プロセッサによる演算、処理結果画像の画像メモリへの書き込みの順番で行われます。
処理対象画像は、ソース画像またはソース画面と呼びます。
処理結果画像は、デスティネーション画像またはデスティネーション画面と呼びます。

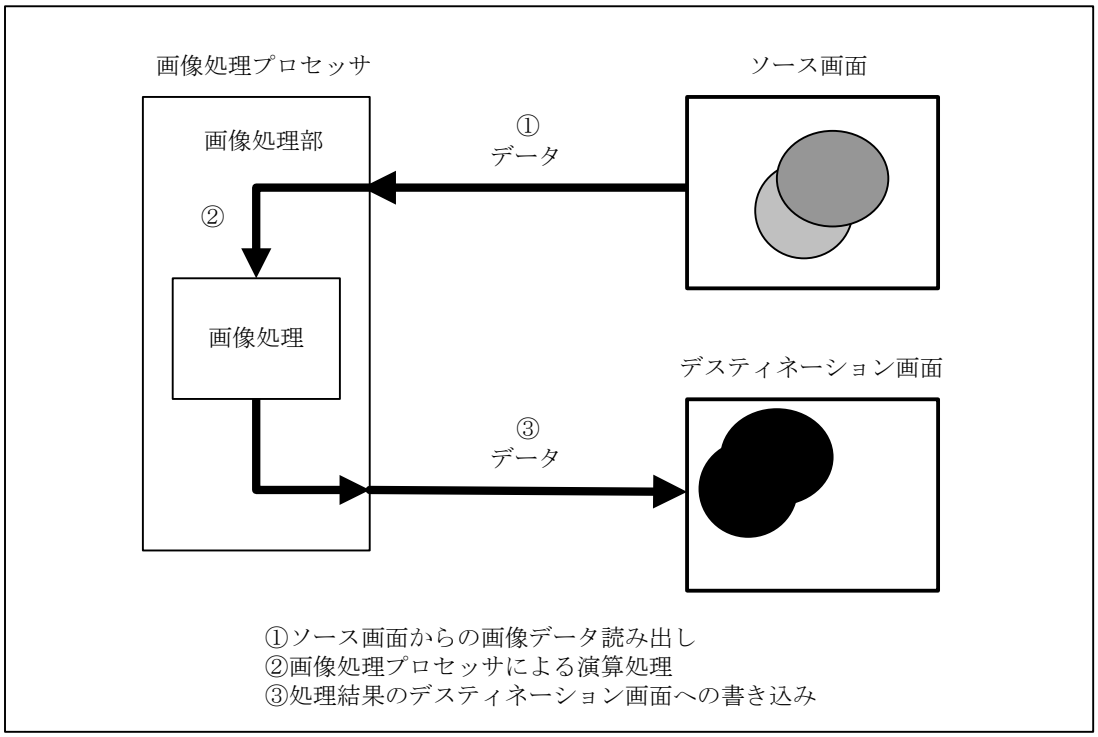
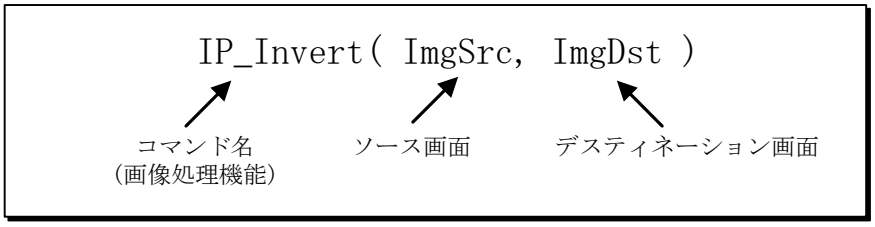


図5-1-1　画像処理の概要

本ライブラリでは、画像処理の機能をＣ言語のサブルーチン形式で提供します。画像処理のプログラムでは、コマンド（画像処理機能）の選択、ソース画面とデスティネーション画面の選択によって、画像処理機能を実行します。



画像処理の手順と各機能との関係は次のとおりです。ただし、これは一般的な場合の例であり、アプリケーションごとに、若干前後することがあります。

表5-1-1　画像処理の手順と各機能の対応

画像処理の手順	システム制御	画像メモリ領域管理	画像メモリ制御	映像入力表示制御	画像処理	パターン作成
開始 ↓ イニシャライズ（初期化）	○					
↓ 処理画像領域の設定		○				
↓ 画像データ入力				○		
↓ 画像間演算・微分等の画像前処理					○	
↓ 面積等の特徴量を抽出					○	
↓ 画像メモリの内容を読み出し／書き込み			○			
↓ 処理結果を表示する際の文字等の書き込み／重ね合せ表示				○		○
↓ 終了						

5.2 アフィン変換

アフィン変換とは、拡大／縮小、移動、回転などの幾何学変換のことです。本ライブラリでは、下の式の２次元のアフィン変換をハード処理とソフト処理によりサポートします。２～８倍及び１／２～１／８倍の整数倍の拡大／縮小と移動はハード処理で実現し、それ以外はソフト処理で実現します。

$$X = ax + by + c$$
$$Y = dx + ey + f$$

x，y	変換前の座標
X，Y	変換後の座標
a，b，c，d，e，f	任意定数

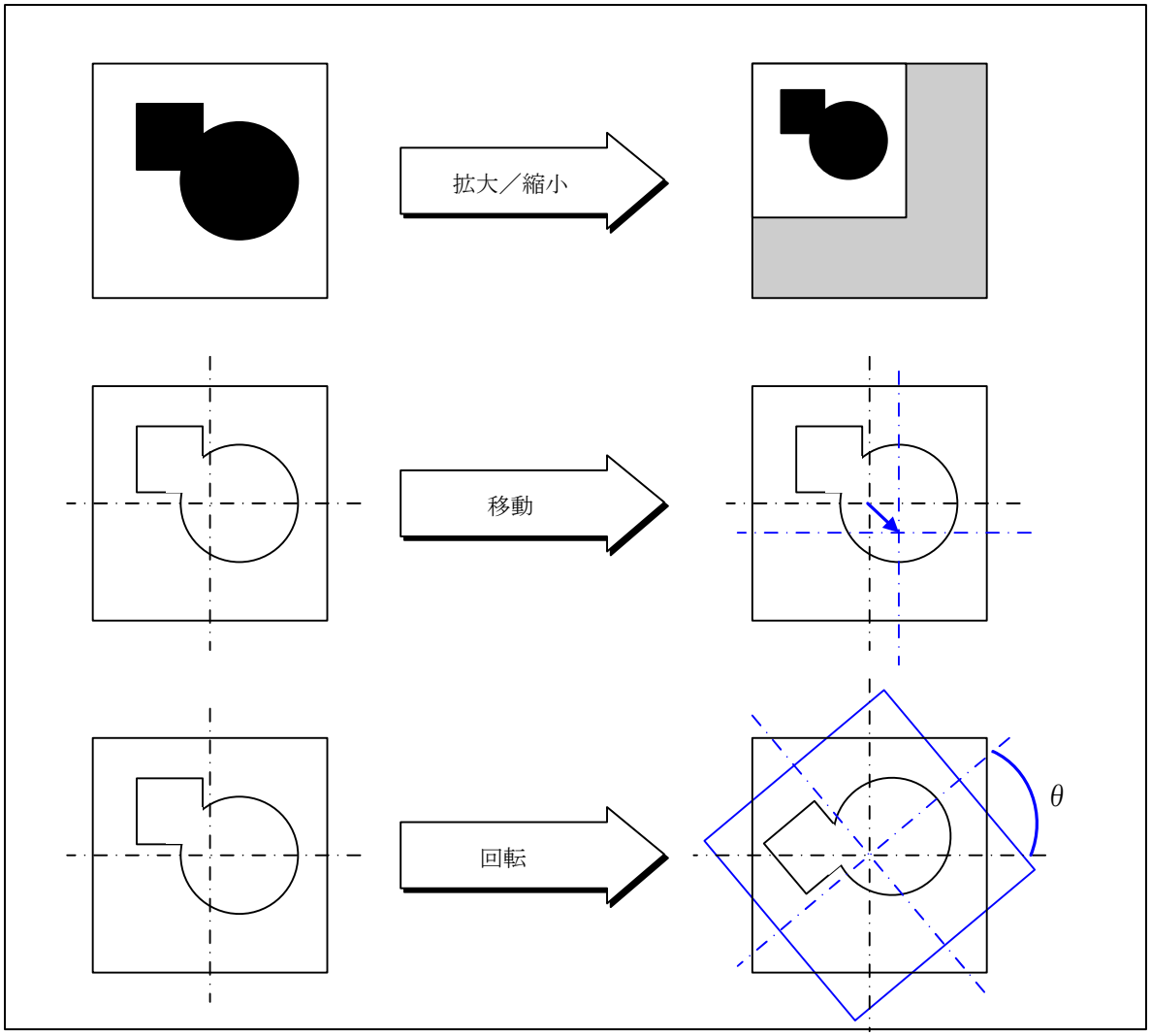


図5-2-1　アフィン変換

5.3 2値化

２値化とは、濃淡画像（２５６階調）を黒（濃度０）と白（濃度２５５）の２階調の画像に変換することです。本ライブラリコマンドでは、任意のしきい値以上の濃度値を白、それ以外の濃度値を黒とする通常の２値化と、任意の濃度値の範囲内と範囲外で白か黒にする範囲・反転付２値化をサポートします。

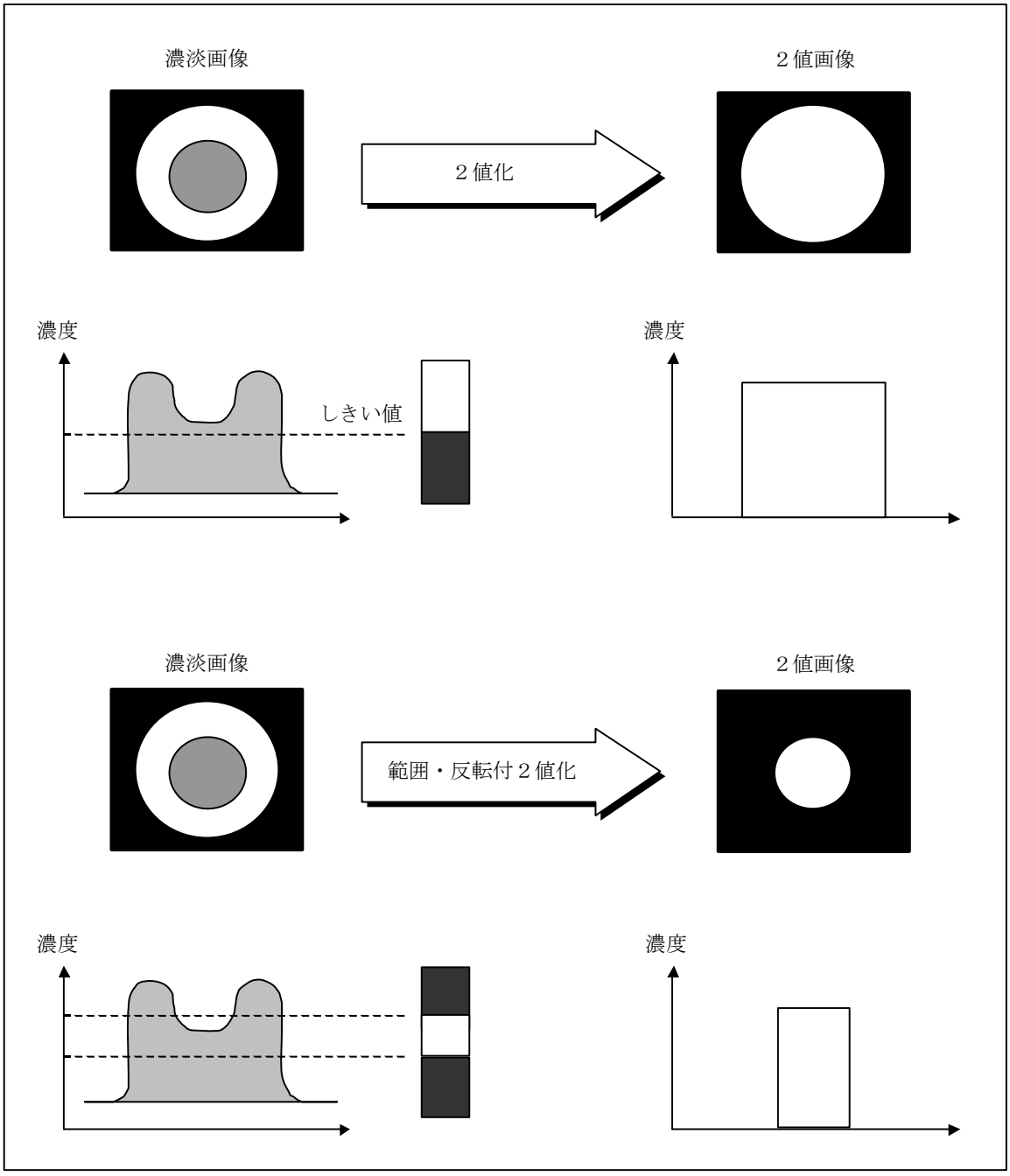


図5-3-1　2値化

5.4 濃度変換

濃度変換とは、濃度（輝度・明るさ）を変換する処理です。
濃度変換のパターンを換えることにより、暗い部分の濃度を高くしたり、逆に明るい部分の濃度を低くしたりする画質改善が可能です。画質改善以外にも、等濃度縞（ストライプ）のデータを使用すれば、曲面の曲がり、くぼみなどを見つけることができます。さらに、この考えを拡大していけば、３値化、４値化という処理が行えます。濃度変換のパターンは、自由に設定可能です。

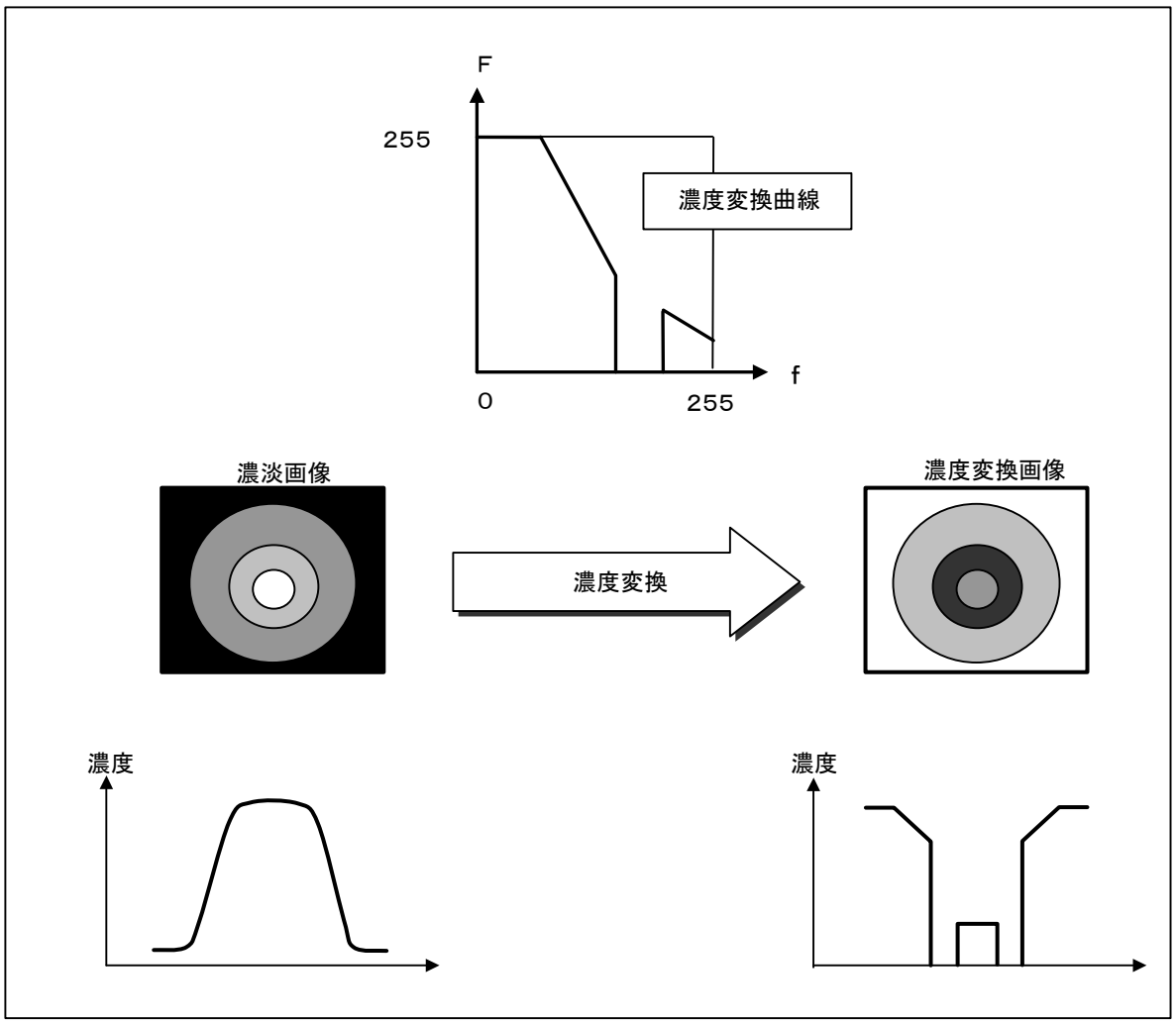
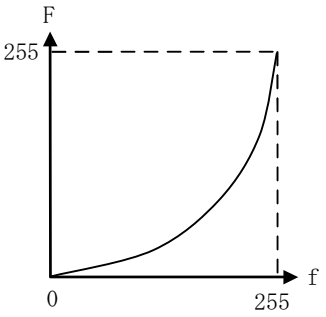
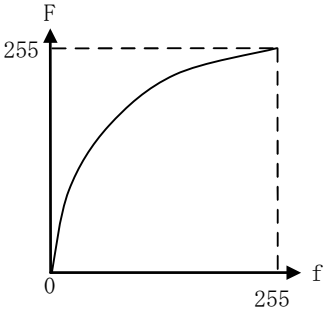
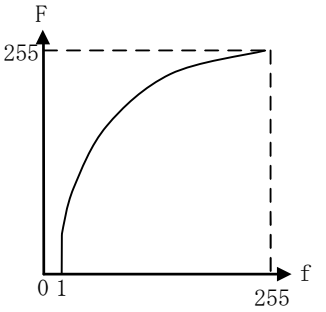
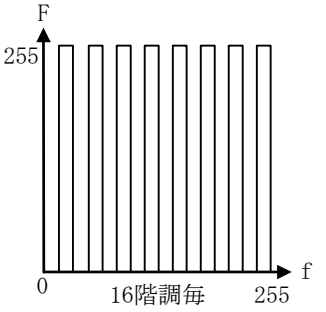
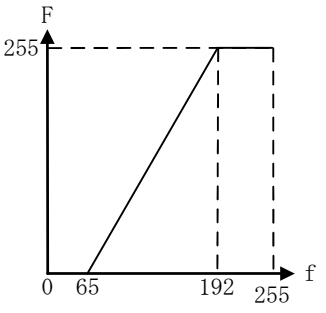


図5-4-1　濃度変換

表5-4-1　濃度変換詳細

種類	濃度変換曲線	詳細	効果
γ補正①		0≦f≦255において $F = af^{1.2}$ （但し $a = 255^{-0.2}$ ） f ： 処理対象画像濃度 F ： 処理結果画像濃度	低濃度部の強調
γ補正②		0≦f≦255において $F = af^{1/1.2}$ （但し $a = 255^{-0.2/1.2}$ ） f ： 処理対象画像濃度 F ： 処理結果画像濃度	高濃度部の強調
log変換		1≦f≦255において F = a*log(f) （但し a = 255/log(255)） f = 0 において F = 0 f ： 処理対象画像濃度 F ： 処理結果画像濃度	１以下を０とし高濃度部を強調
等濃度縞		0≦f≦255において 16階調毎にF=0 or F=255 f ： 処理対象画像濃度 F ： 処理結果画像濃度	１６階調単位に黒と白に変換
強調		0≦f≦255において f = 0 ～ 64 において F = 0 f = 65 ～ 191 において F = 255/127*(f-65) f = 192 ～ 255において F = 255 f ： 処理対象画像濃度 F ： 処理結果画像濃度	低濃度部と高濃度部の強調 （コントラスト改善）

5.5 画像間算術演算

画像間演算とは、２画面の画像で演算を行い演算後別の１画面に合成することです。画像間演算には加算、減算、乗算などの画像間算術演算と論理和、論理積などの画像間論理演算があります。

画像間算術演算の用途としては、基準パターンとの減算による差画像から欠陥を検出したり、何度も同一画像を取込み加算することでランダムノイズ（ホワイトノイズ）を除去するといったことがあります。また、リアルタイムに２画面の画像の最大値をとることで、明るい物体を次から次へと１画面の画像に合成することができます。これは、移動物体の軌跡や移動量を検出するために有効です。

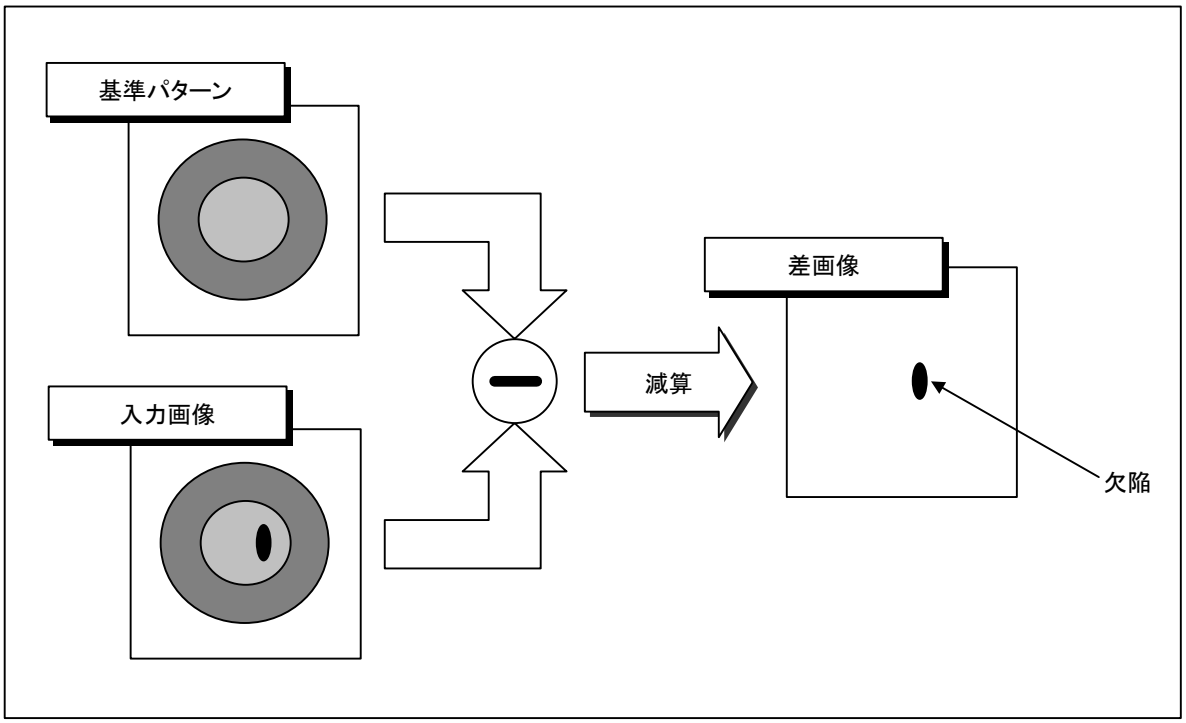


図5-5-1　画像間算術演算

5.6 画像間論理演算

２画面間で１画素毎に論理和、論理積などの論理演算を行い結果を別の画像に格納します。
２値画像での基準パターンと検査パターンとのチェック（ＸＯＲ演算）、画像の合成（ＯＲ演算）、キズの検出に有効です。
また、画像に任意のパターンでマスクをかける場合にも任意パターンとターゲット画像のＡＮＤ演算を行います。

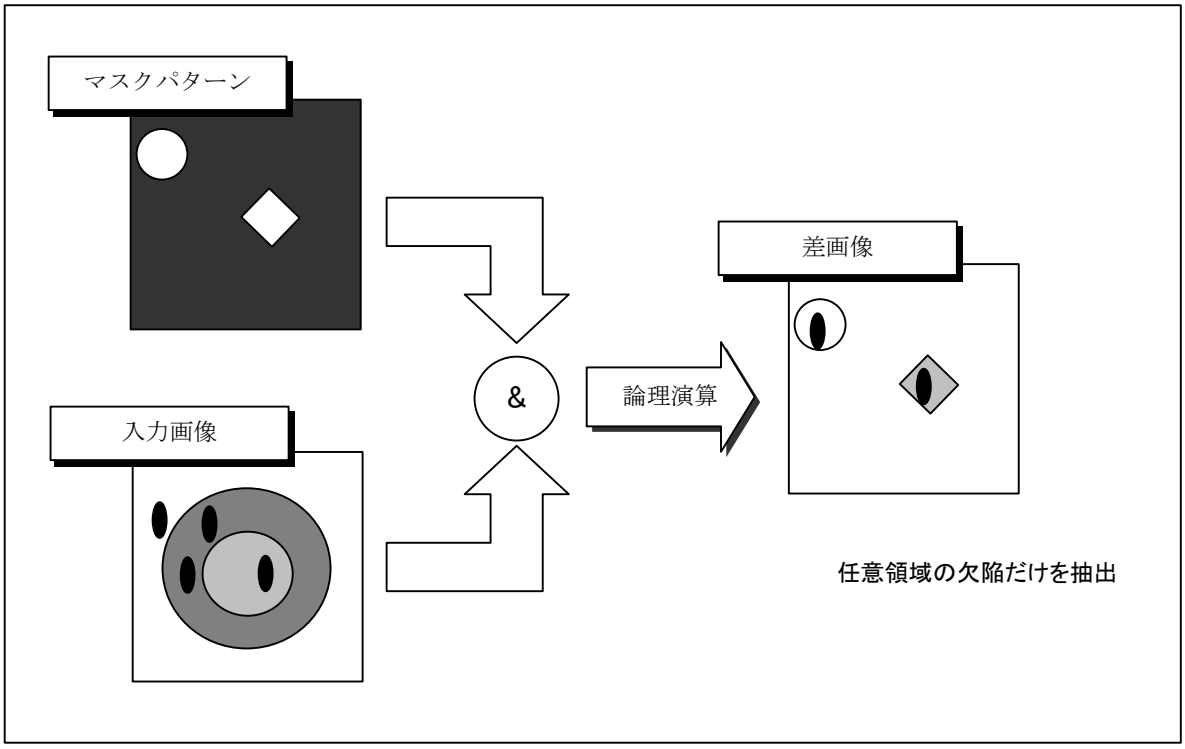


図5-6-1　画像間論理演算

5.7 2値画像形状変換

２値画像に対してノイズ除去、輪郭抽出、膨張、収縮、細線化、縮退化といった処理を行うことができます。これらの処理は、目的に応じて４連結と８連結を選択できます。

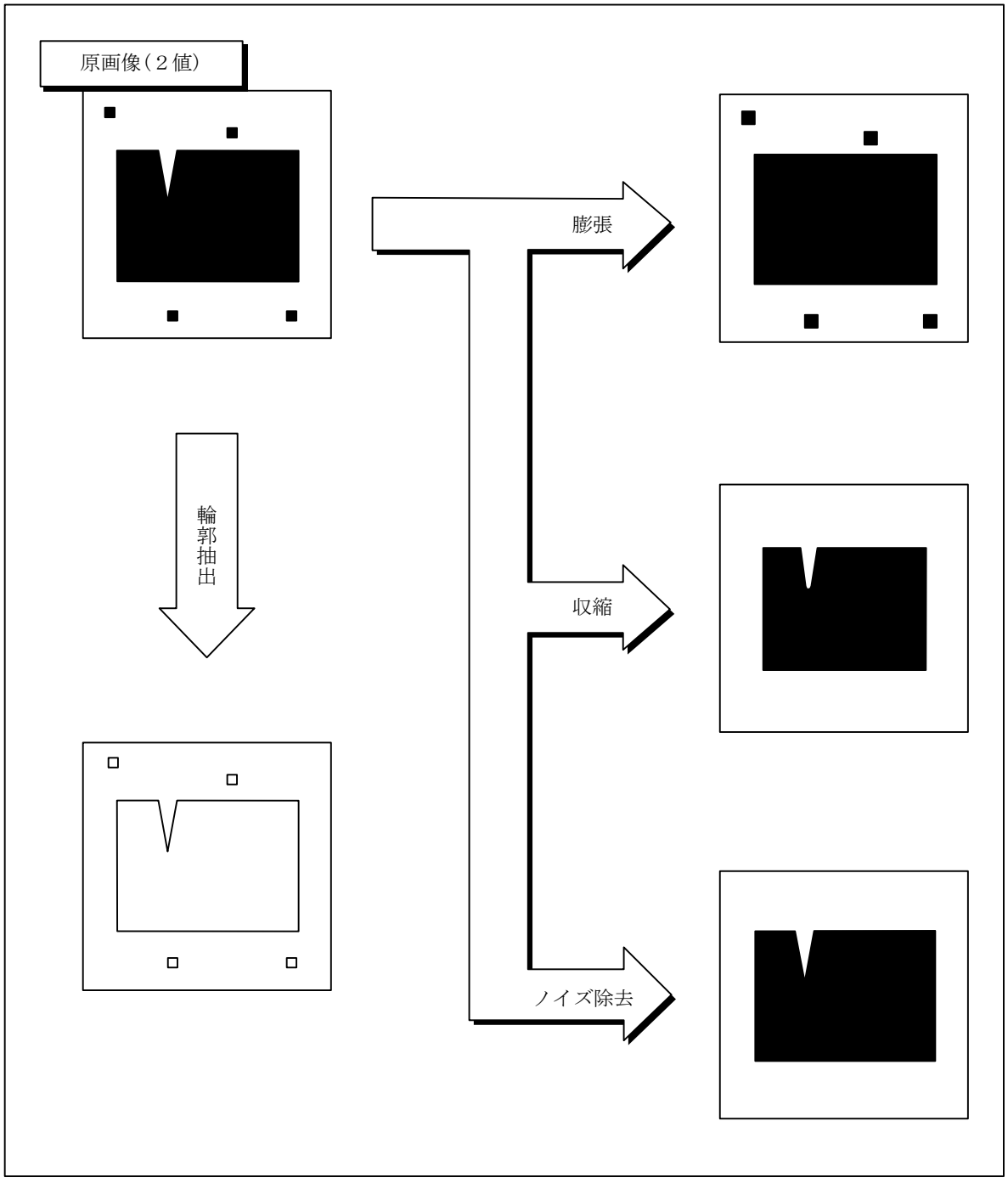


図5-7-1　２値画像形状変換

5.8 コンボリューション

コンボリューションは、近傍領域の積和演算です。下図に示すように、ソース画面の指定領域内の全画素に対して、注目画素を中心とした３×３近傍の局所領域で与えられた荷重係数との積和演算を行います。

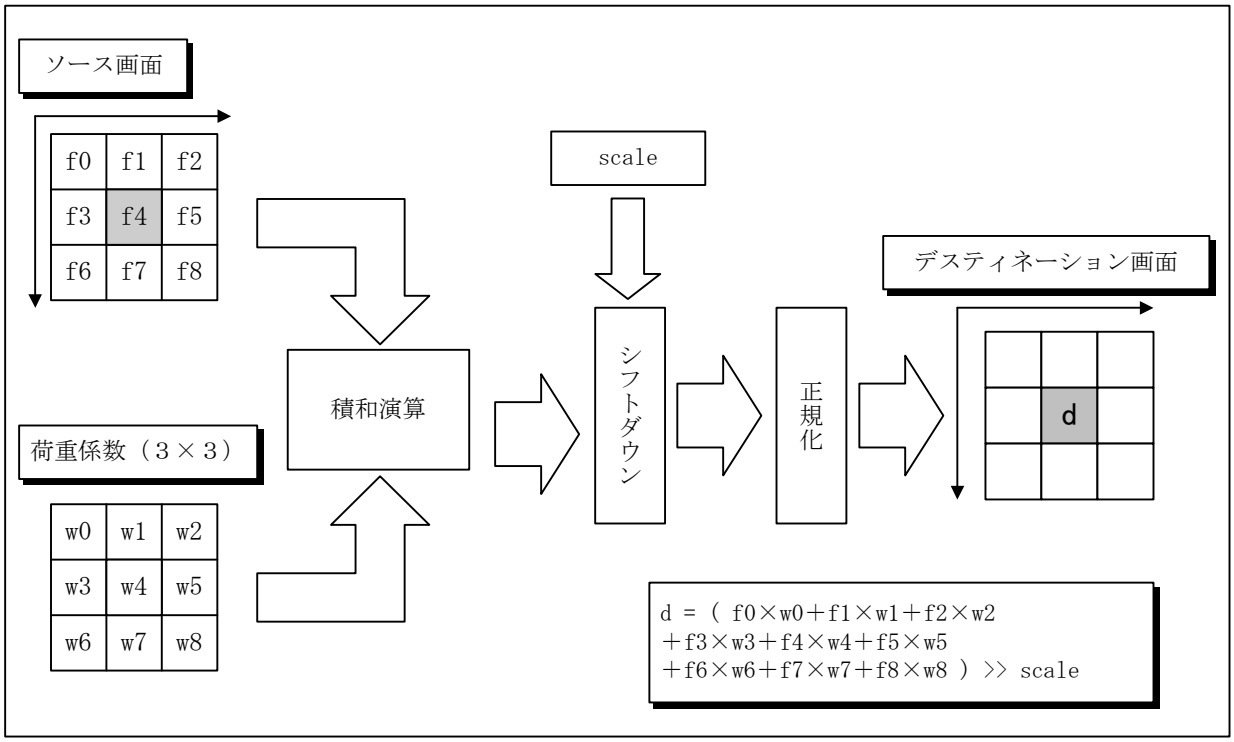


図5-8-1　コンボリューション

この演算は下表に示す荷重係数の設定により、濃淡画像での平滑化や輪郭強調などを行うことができます。

表5-8-1　コンボリューション詳細

種類･名称		荷重係数	scale	効果
平滑化	－	1 1 1 1 1 1 1 1 1	3	平滑化。画像の輪郭を滑らかにする。
１次微分 グラディエント	微分	0 0 0 0 1 -1 0 0 0	0	輪郭強調（X方向）
		0 0 0 0 1 0 0 -1 0	0	輪郭強調（Y方向）

種類･名称		荷重係数	scale	効果
１次微分 グラディエント	Roberts	0 0 0 0 1 0 0 0 -1	0	輪郭強調（X方向）
		0 0 0 0 0 1 0 -1 0	0	輪郭強調（Y方向）
	Sobel	-1 0 1 -2 0 2 -1 0 1	0	輪郭強調（X方向）
		-1 -2 -1 0 0 0 1 2 1	0	輪郭強調（Y方向）
２次微分 ラプラシアン	４連結	0 -1 0 -1 4 -1 0 -1 0	0	輪郭強調
	８連結	-1 -1 -1 -1 8 -1 -1 -1 -1	0	輪郭強調

5.9 ランクフィルタ

ミニマムフィルタ、マックスフィルタ、メディアンフィルタなどをランクフィルタと呼びます。ランクフィルタ処理は、ソース画面の指定領域内の全画素に対して、注目画素を中心とした３×３近傍の局所領域内の画素データをソート（順に並べる）し、任意の順番の画素データを抽出します。また、３×３近傍の局所領域内で任意のカーネルマスクでマスク処理を行い、有効となった画素データ内でソートすることができます。

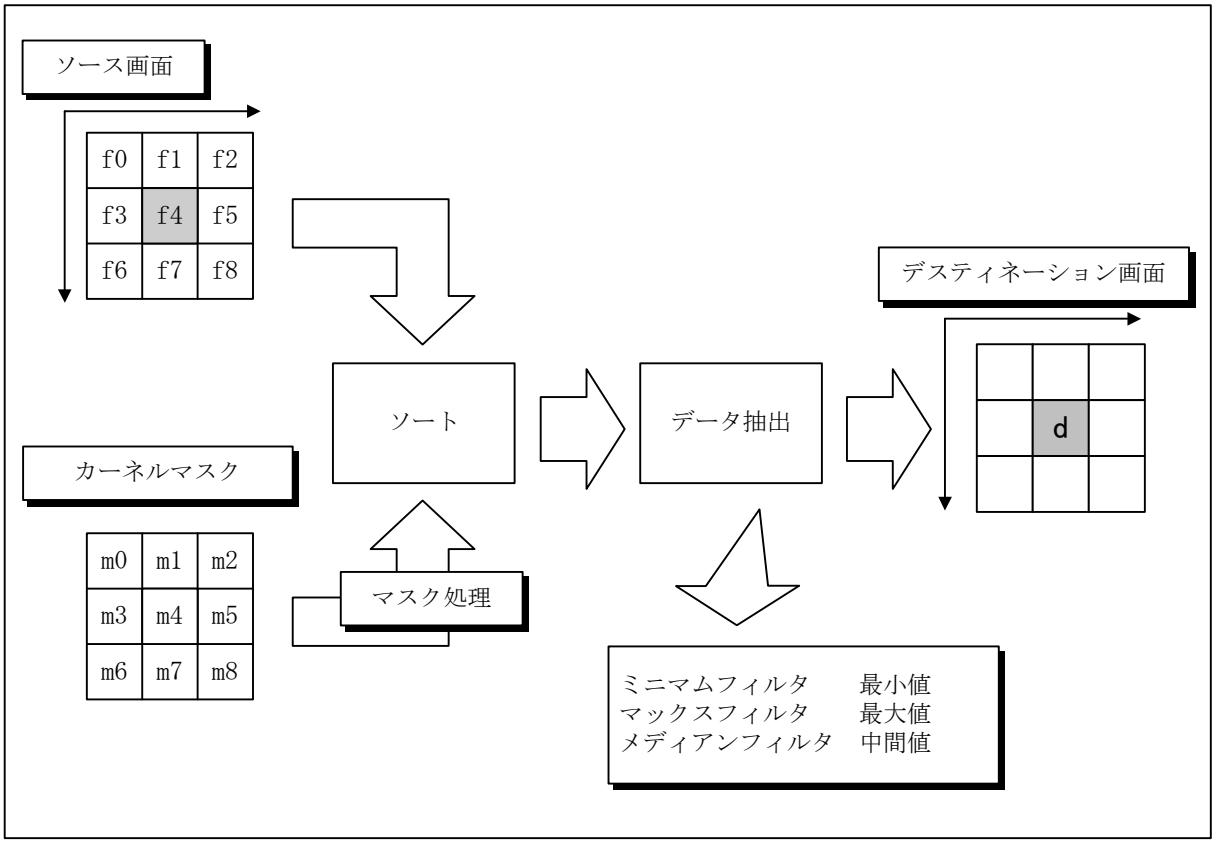


図5-9-1　ランクフィルタ

5.10 ラベリング

ラベリング処理では、２値画像に対して連結している物体ごとに番号を付けます。ラベリング処理の種類には、すべての物体をラベリングするものと、一定範囲内の面積物体にのみラベリングするものがあり、それぞれにオプションとして白をラベリングするか、黒をラベリングするかが選択できます。

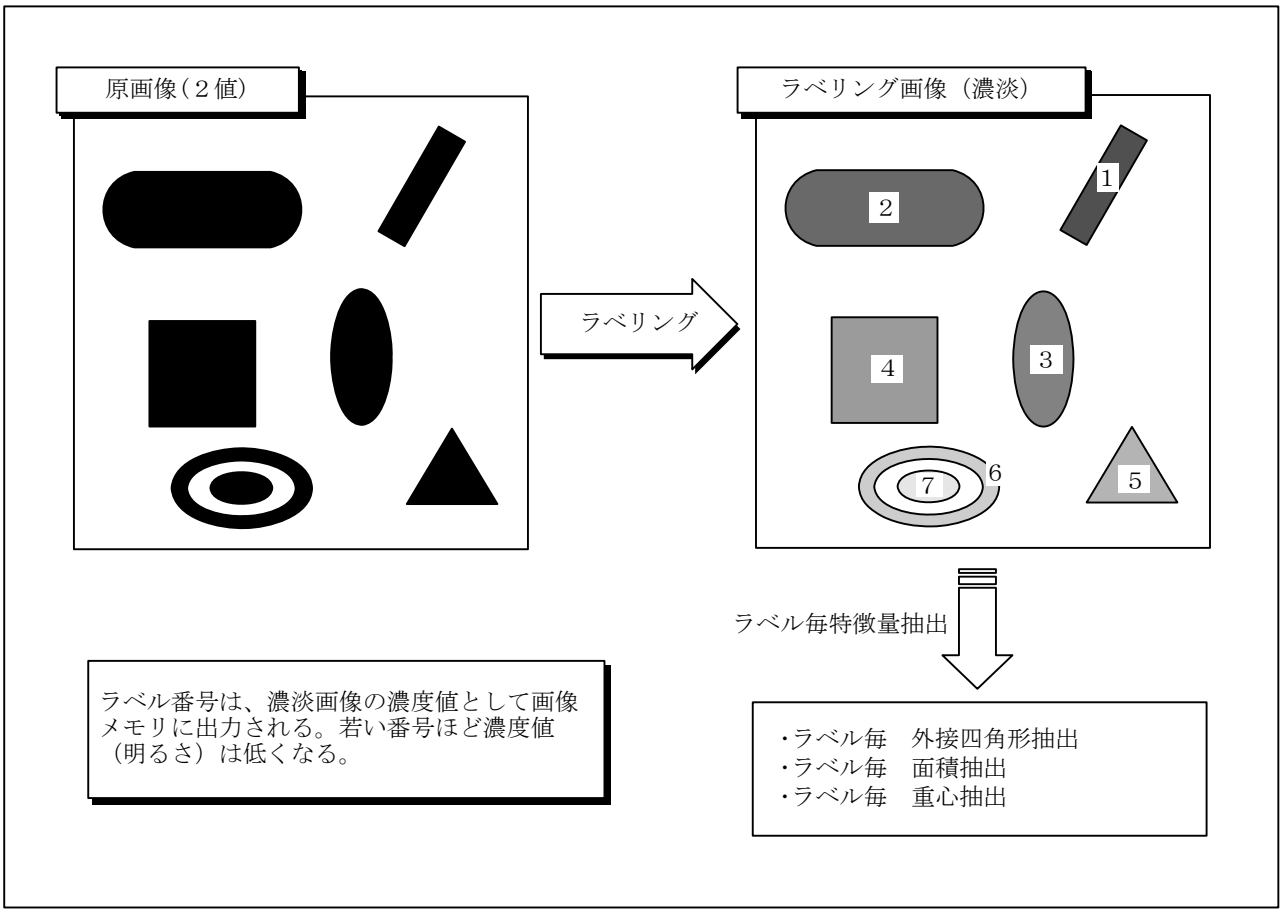


図5-10-1　ラベリング概要

ラベリング処理は、仮ラベル付け、合流対検出、真ラベル付けの３段階で行われます。

次にそれぞれの処理について説明します。
原画像図ａに対してラベリング処理を起動すると、画面左上から仮ラベル付けが始まります。ラスタスキャンにて番号付けを行うため、図ｂに示すように同じ物体を異なるラベルとして認識します。この段階では物体Ａに対して１、２、３の３つの仮ラベル番号が、物体Ｂに対しては４、５の２つの仮ラベル番号が割り当てられます。そこで次の段階として合流対検出を行います。合流対検出とは同じ物体に対して複数の仮ラベル番号が付いていないかを調べる処理であり、図ｂに対しては図ｃに示すように、１、２、３は同じ、４、５は同じという結果が得られます。この結果をもとに真ラベル付けを行うことにより、最終的に図ｄに示す結果が得られます。
なお、各々の処理には、
・仮ラベル付け番号：最大　１０２２番まで
・合流対　　：最大　１０２２件まで
・真ラベル　　：最大　２５５個まで
との制約があり、この値を超えるとオーバーフローとなり、正しい結果が得られません。

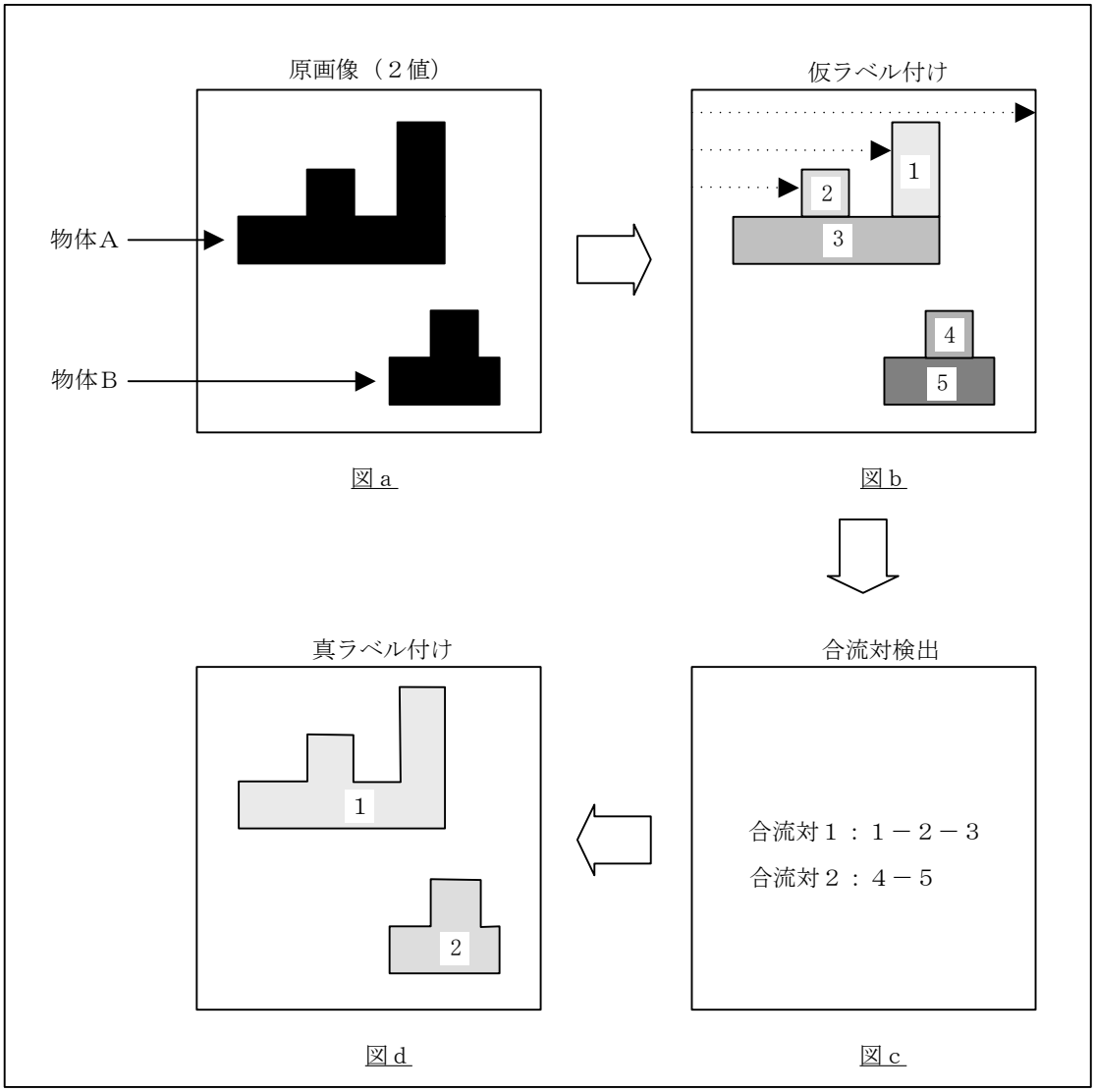


図5-10-2　ラベリング処理

5.11 ヒストグラム

ヒストグラム処理では、２値画像と濃淡画像に対し、最大／最小濃度、濃度頻度分布抽出等の統計処理が行えます。

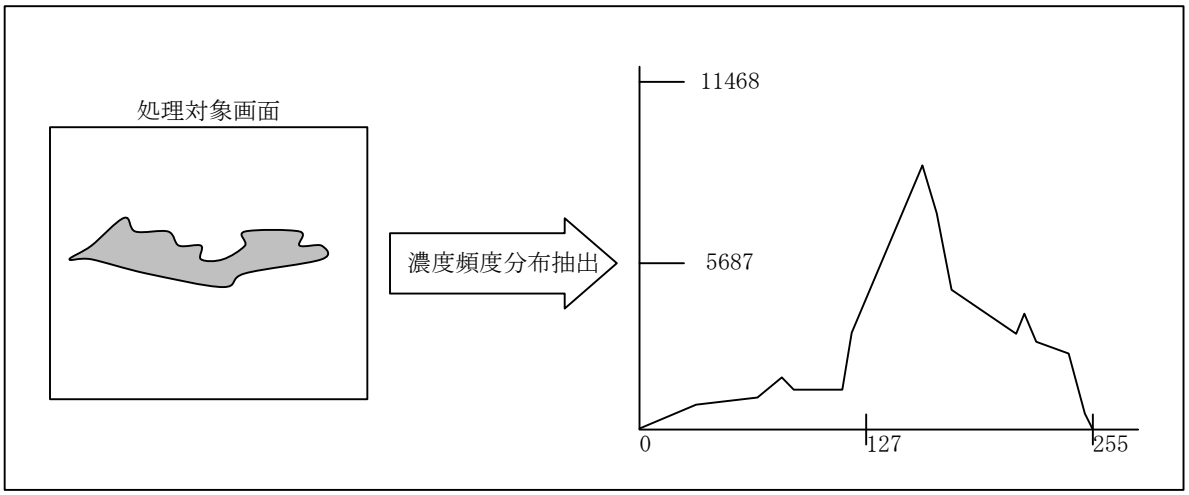


図5-11-1　濃度頻度分布抽出

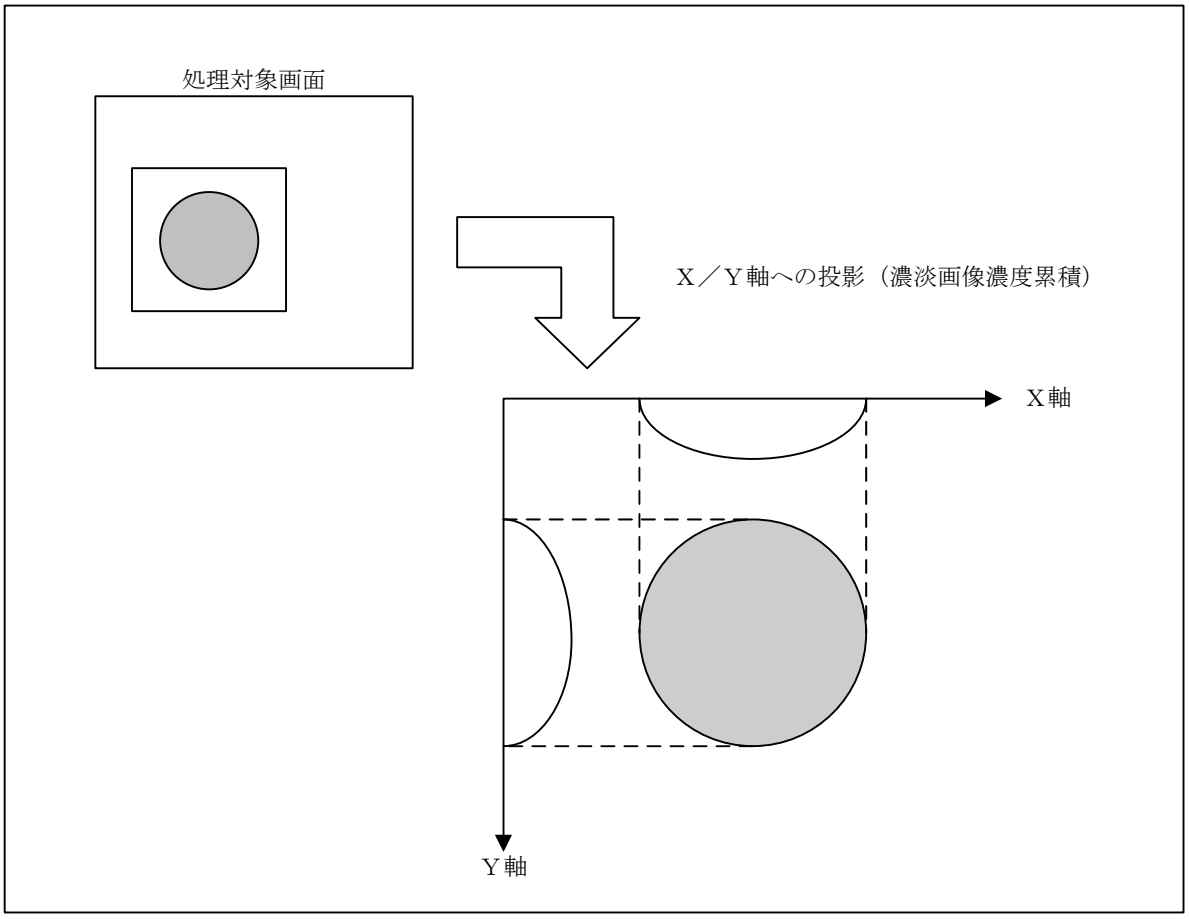


図5-11-2　X／Y軸への投影

ヒストグラム処理での座標系は、ウィンドウ相対であり、ウィンドウの始点が座標０になります。
よって、処理結果の座標はウィンドウ始点を基準とした座標であり、処理結果のテーブルはウィンドウ内の有効データがテーブルの先頭から格納されます。

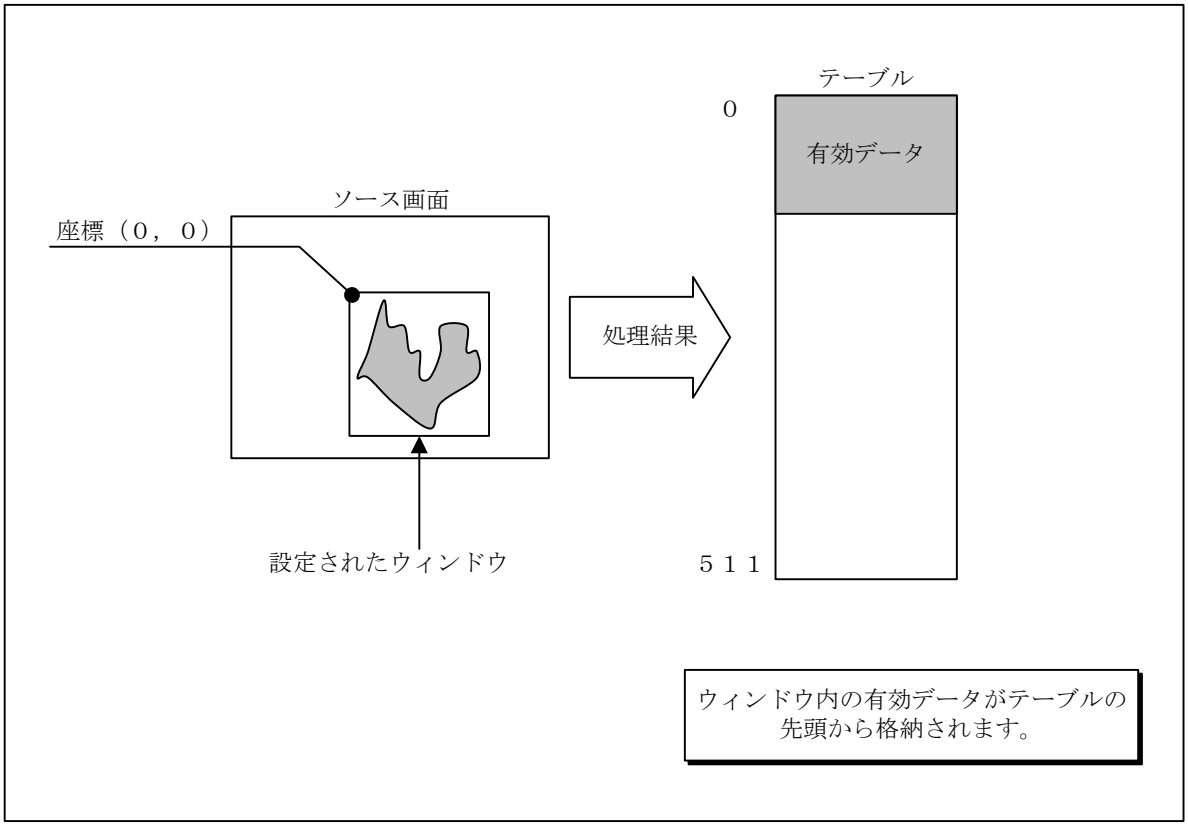


図5-11-3　処理結果テーブル

5.12 画像メモリアクセス

画像メモリアクセスコマンドでは、画像メモリのデータを１画素単位または、ブロック単位でアクセスすることができます。

5.12.1 画像メモリアクセス手順フロー

以下に、画像メモリアクセス手順フローを示します。

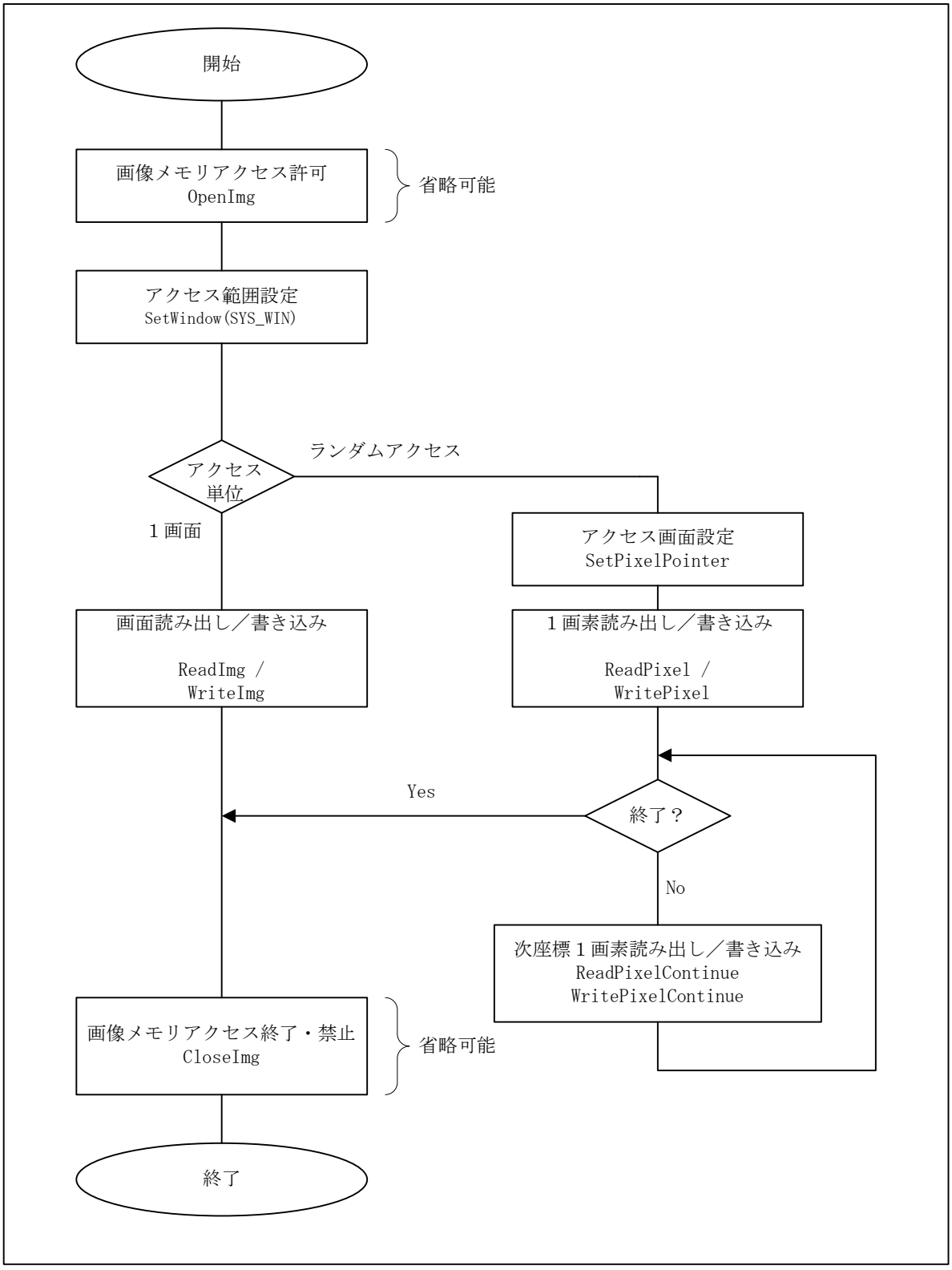


図5-12-1　画像メモリアクセス手順

5.12.2 ウィンドウの設定

画像メモリアクセスコマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はSYS_WINです。ReadImgコマンドはSYS_WINで設定された矩形領域の画素データをローカルメモリにブロック単位に読込み、WriteImgコマンドはローカルメモリ上のデータをSYS_WINで設定された矩形領域の画像メモリにブロック単位に書込みます。
また、画像メモリ制御コマンドでの座標系はウィンドウ相対であり、ウィンドウの始点が座標０となります。

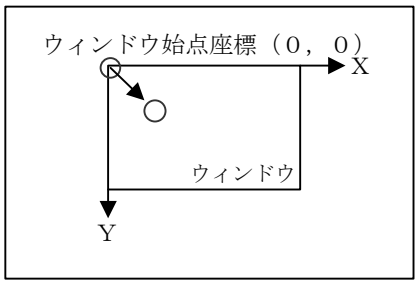


図5-12-2　ウィンドウ座標系

5.12.3 画面データタイプ

画像メモリアクセスを行う画面の画面データタイプは符号なし８ビット、符号付８ビット、２値です。なお、２値データ０は画素データ「０」、２値データ１は画素データ「２５５」です。

5.12.4 画像メモリ直接アクセス

本ライブラリでは、画像メモリ直接アクセスコマンドにより画像メモリの先頭アドレスを取得し、画像メモリを直接アクセスすることができます。
実際の処理では、OpenImgコマンドはパラメータで指定した画面の画面サイズでメモリを確保し、その領域に指定画面の画像データを格納します。OpenImgコマンドが返すアドレスはこの領域の先頭アドレスです。CloseImgDirectコマンドは、OpenImgDirectコマンドで確保したメモリの画像データをOpenImgDirectコマンドで指定した画面の画像メモリに書き戻してから、確保メモリを解放します。そのため、CloseImgDirectコマンドの実行で画像データの変更が反映されます。OpenImgDirectコマンド実行後は必ずCloseImgDirectコマンドを実行して下さい。
以下に、画像メモリ直接アクセス手順フローを示します。

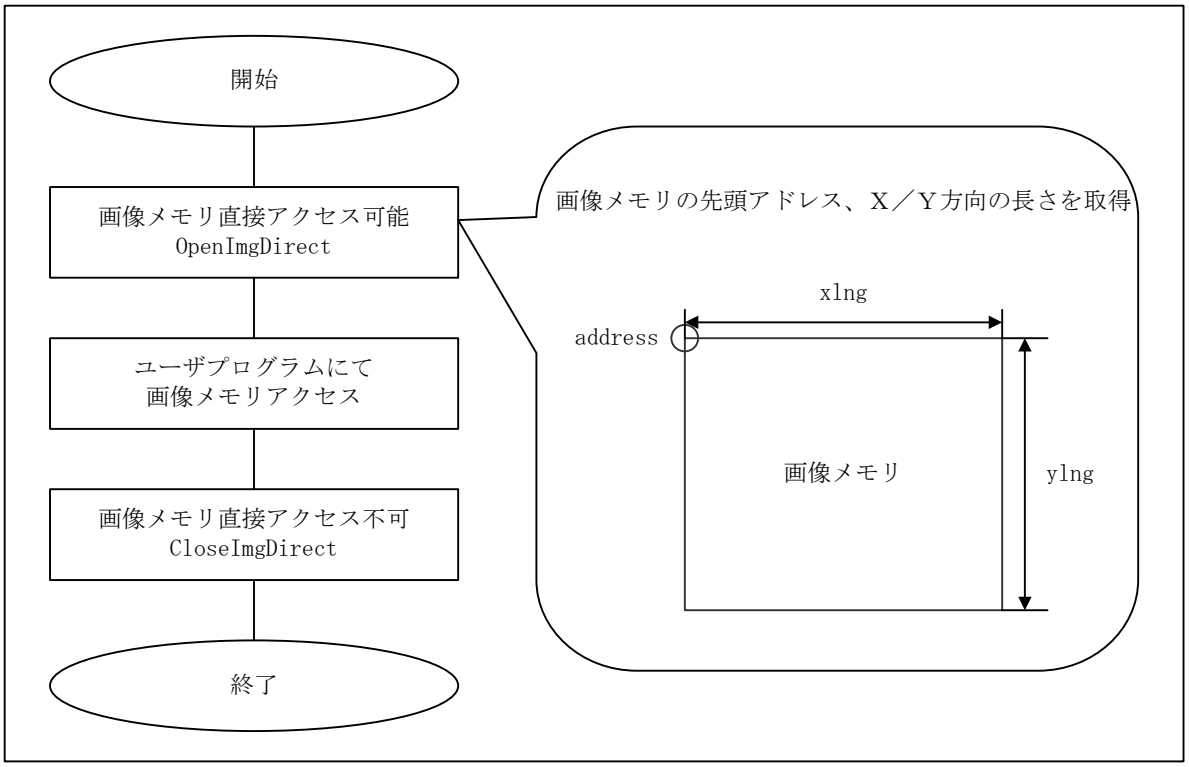


図5-12-3　画像メモリ直接アクセス手順

5.13 2値パイプラインフィルタ

２値パイプラインフィルタコマンドは、２値画像に対し、３×３カーネルを用いてノイズ除去、膨張、収縮、輪郭抽出の４つの画像処理を組み合わせて、最大８段までのフィルタ処理のパイプライン処理を行います。また、８段パイプラインを４段×２の構成にして、４段目の出力結果を論理演算して出力することもできます。

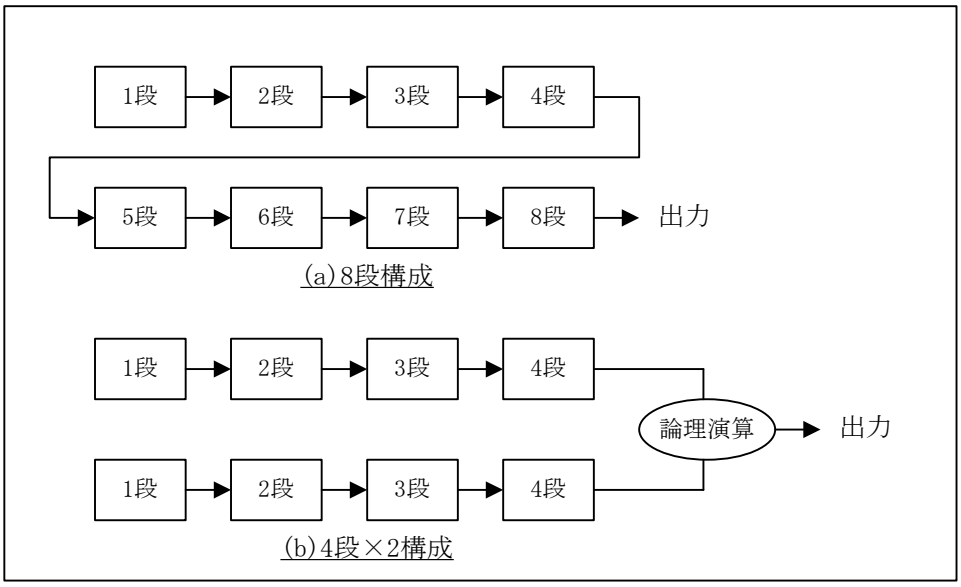
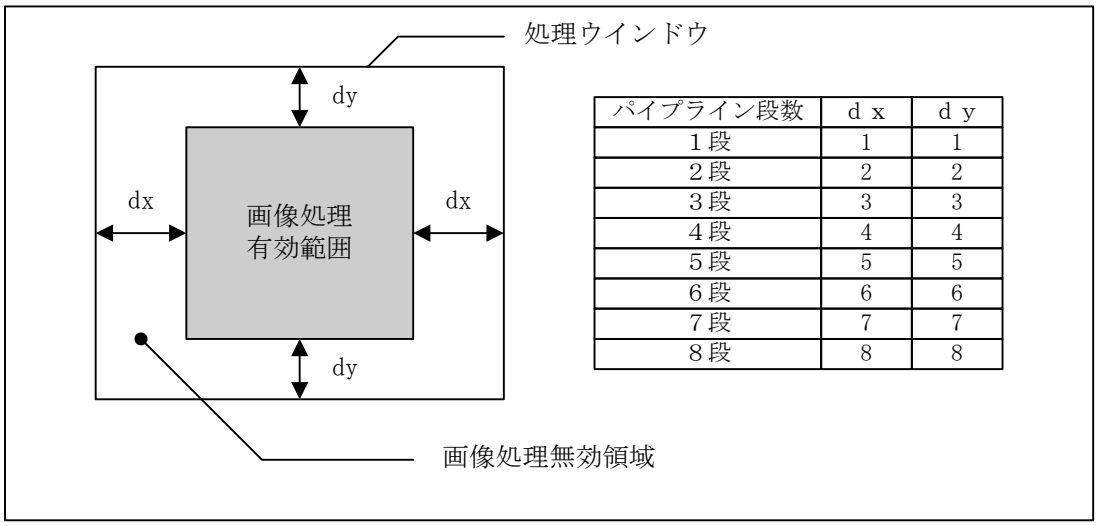


図5-13-1　２値パイプラインフィルタ

5.13.1 2値パイプラインフィルタ処理領域

３×３カーネルを用いて画像処理を行う為、画像処理対象エリアのエッジ部分（周辺）に画像処理結果無効の範囲が生じます。また、この画像処理無効範囲は、パイプラインの段数によって決まります。
段数による無効範囲を以下に示します。



パイプライン段数	ｄｘ	ｄｙ
１段	1	1
２段	2	2
３段	3	3
４段	4	4
５段	5	5
６段	6	6
７段	7	7
８段	8	8

図5-13-2　処理領域

5.14 パイプライン制御

ＳｏｆｔＶＰではパイプライン制御は動作しません。

『EnablePipeline』および『DisablePipeline』はダミー関数（処理なし）として実装しています。このため、ＳｏｆｔＶＰで『EnablePipeline』を行う場合は注意が必要になります。

パイプラインモードが有効の場合、ＶＰ－Ａｘシリーズの画像認識ライブラリでは各ウィンドウのＡＮＤ領域を一度に処理しますが、ＳｏｆｔＶＰではコマンドを順次実行するため、コマンドの実行前に各ウィンドウを設定する必要があります。

以下にパイプラインモード有効時の回避例を示します。

パイプライン処理

```
EnablePipeline();
IP_Add(ImgSrc0, ImgSrc1, ImgID);
IP_Binarize(ImgID, ImgDst, thr);
DisablePipeline();
```

回避例

```
EnablePipeline();
IP_Add(ImgSrc0, ImgSrc1, ImgID);
#ifdef SOFTVP
  SetWindow(SRC0_WIN,dsx,dsy,dex,dey);
#endif
IP_Binarize(ImgID, ImgDst, thr);
DisablePipeline();
```

・『IP_Add』の結果はいったんImgIDのデスティネーション画面ウィンドウの領域に出力されます。次の『IP_Binarize』の入力画面がImgIDになりますので、ソース画面ウィンドウをデスティネーション画面ウィンドウと同じに設定する必要があります。

・ＳｏｆｔＶＰでは不定画面は発生しません。

・ＶＰ－Ａｘシリーズのパイプライン処理については、ユーザーズマニュアルの『パイプライン制御』を参照してください。

5.15 2値マッチングフィルタ

２値マッチングフィルタコマンドは、２値画像に対してテンプレートでマッチングフィルタ処理を行い、その処理結果を出力します。

入力画像に対し、９×９画素のテンプレートでＡＮＤ／ＸＮＯＲのマッチングフィルタ処理を行います。処理結果は、”１”となる画素の総数となります。

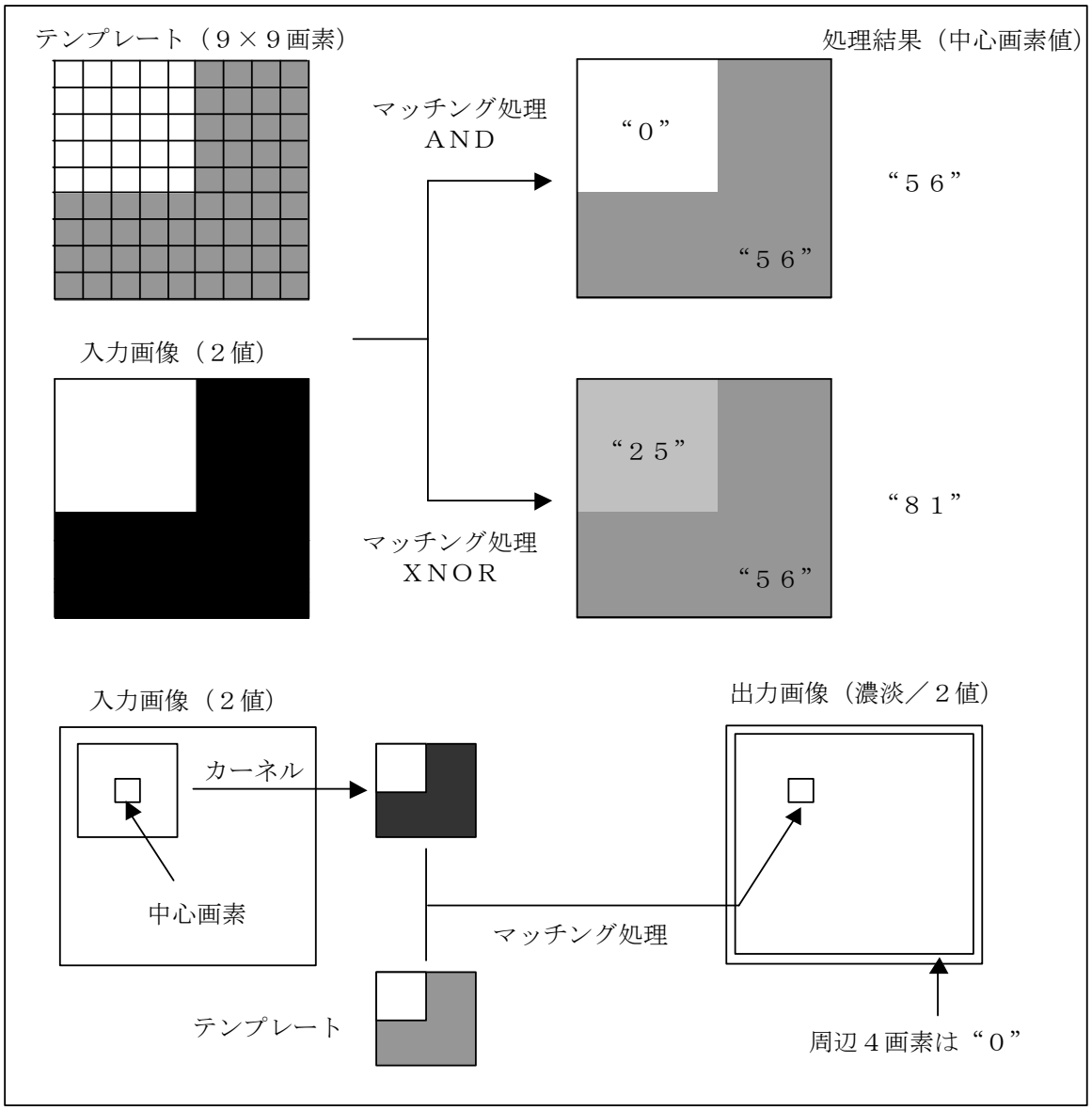


図5-15-1　２値マッチングフィルタ処理

２値マッチングフィルタ処理は、９×９テンプレートを用いて画像処理を行うため、画像処理対象領域の周辺４画素に画像処理結果無効の領域が生じます。この画像処理無効領域は”０”となります。

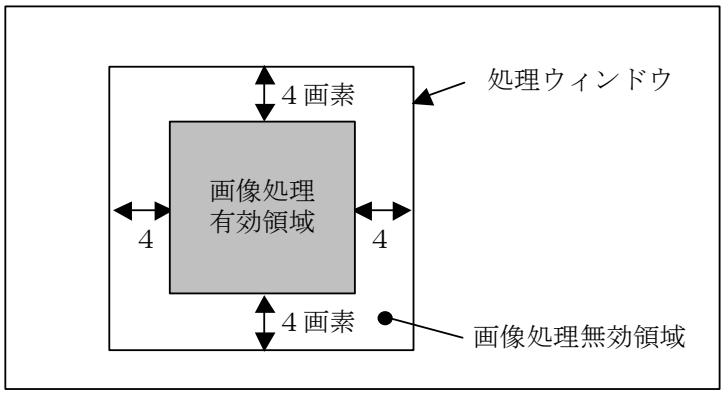


図5-15-2　２値マッチングフィルタ領域

5.16 正規化相関

正規化相関処理とは、濃淡画像によるパターンマッチングのことであり、ユーザの登録した濃淡テンプレートを、処理対象の画面の中から探し当てる（座標、相関値）ことができます。

正規化相関では、テンプレートと対象画面間で下記演算処理を行います。相関値（ｒ）は０～１の値を取り、濃淡テンプレート（ｇ）とサーチ対象画面（ｆ）が完全に一致すると１となります。

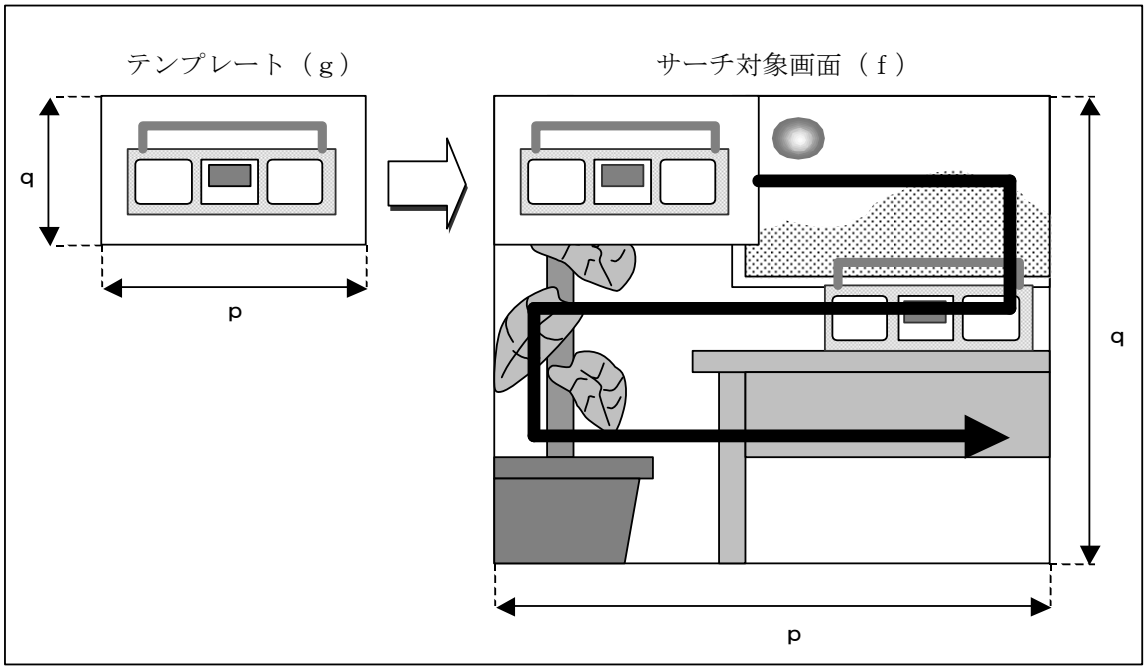


図5-16-1　正規化相関の概要

$$r^2 = \frac{\{ N \sum_{u}^{p} \sum_{v}^{q} f(u,v)g(u,v) - \sum_{u}^{p} \sum_{v}^{q} f(u,v) \times \sum_{u}^{p} \sum_{v}^{q} g(u,v) \}^2}{[N \sum_{u}^{p} \sum_{v}^{q} f(u,v)^2 - \{ \sum_{u}^{p} \sum_{v}^{q} f(u,v) \}^2] [N \sum_{u}^{p} \sum_{v}^{q} g(u,v)^2 - \{ \sum_{u}^{p} \sum_{v}^{q} g(u,v) \}^2]}$$

上記の式のうち、以下の項の計算はハードウェアによって局所並列的に実行されます。
相関値は、これらの結果を用いてソフトウェアで計算することにより求められます。

(1) $\sum_{u}^{p} \sum_{v}^{q} f(u,v)$　　(2) $\sum_{u}^{p} \sum_{v}^{q} g(u,v)$　　(3) $\sum_{u}^{p} \sum_{v}^{q} f(u,v)^2$　　(4) $\sum_{u}^{p} \sum_{v}^{q} g(u,v)^2$

(5) $\sum_{u}^{p} \sum_{v}^{q} f(u,v)\, g(u,v)$

ｒ：相関値
ｕ：Ｘ座標
ｖ：Ｙ座標
Ｎ：テンプレートの有効画素数

5.16.1 正規化相関実行手順

正規化相関処理の処理手順は、

①テンプレート登録
②サーチモード指定（サーチ個数、サーチ方向やサーチ精度等）
③正規化相関実行

の３段階です。

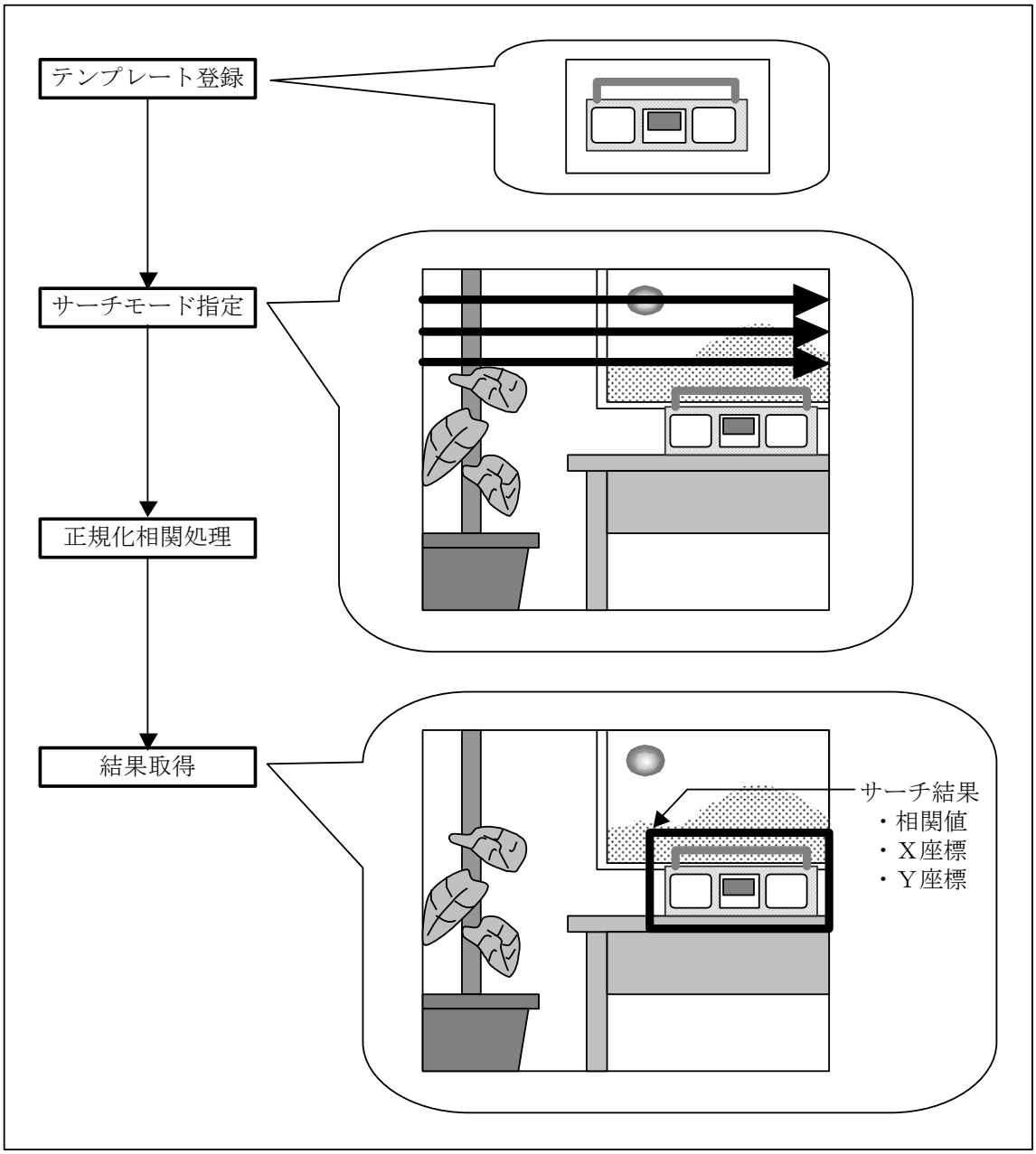


図5-16-2　正規化相関実行手順

以下に正規化相関の実行手順フローを示します。

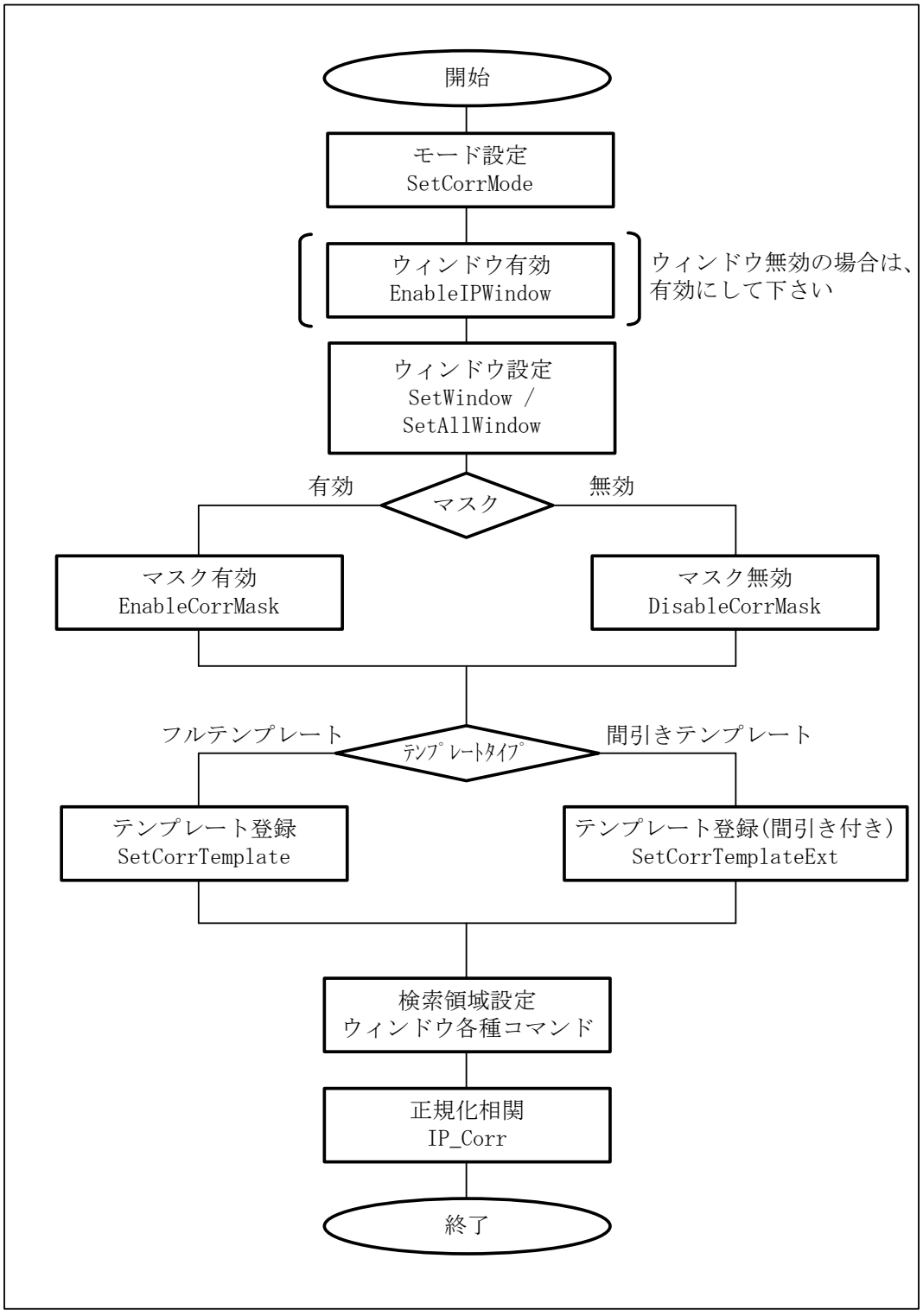


図5-16-3　正規化相関実行手順フロー

5.16.2 テンプレートタイプ

正規化相関のテンプレートには、２種類のタイプがあります。

（1）フルテンプレート

SetCorrTemplateコマンドで登録されたテンプレート。フルテンプレートは、IP_Corr実行時に、処理対象画面内全画素を処理する（フルサーチ）ので、高精度な処理が可能です。

（2）間引きテンプレート

SetCorrTemplateExtコマンドで登録されたテンプレートです。間引きテンプレートは、Ｘ／Ｙ方向の間引き率を登録でき、この間引き率により、テンプレートデータおよび処理対象画面を間引いて演算処理するので、精度はフルテンプレートに対して落ちるが演算回数が減るため、高速処理が可能です。

処理対象画像に応じて、これら２つのテンプレートタイプを評価し使用します。

5.16.3 正規化相関マスク

正規化相関マスクは、テンプレート内で濃度０の画素を処理対象外にします。よって、矩型でない任意の形をテンプレートとしたい場合に有効です。矩型のテンプレート内で必要物体部以外を０で塗りつぶしておき、IP_Corrを実行する前に、EnableCorrMaskを実行すれば、０画素部分は処理されないため、任意形のテンプレートで処理したことと同じになります。
正規化相関マスクはEnableCorrMaskコマンドで設定を行います。

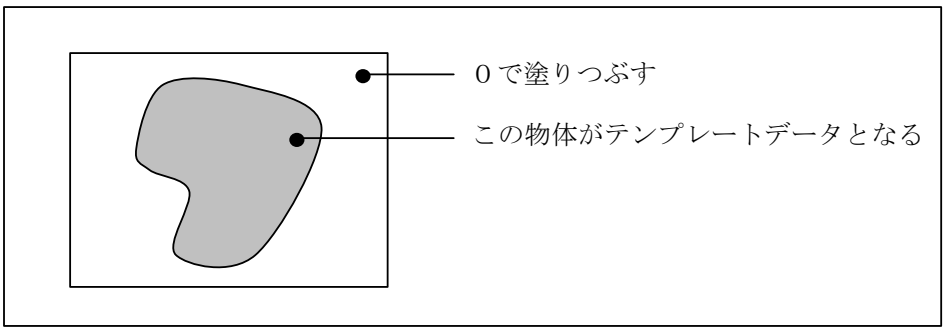


図5-16-4　正規化相関マスク

5.16.4 相関演算途中打ち切りによるサーチの高速化

正規化相関の演算として画像処理プロセッサによりテンプレートカーネル内の積和演算を行っていますが、その他にもう一つ差分累積演算を行う演算器を持っています。これは、テンプレートカーネル内の画像と差分累積演算を行い累積誤差が指定しきい値よりも大きくなったとき、正規化相関の積和演算の処理を途中で打ち切るためのものです。正規化相関の積和演算の処理を途中で打ち切ることにより、正規化相関サーチの高速化を図ることができます。
サーチ対象画像がテンプレート画像とあきらかに異なる場合、相関演算実行中に累積誤差が非常に大きくなると考えられます。そこで、この累積誤差があるしきい値を超えた時点で、ミスマッチと判断し、途中で演算を中止させます。途中で演算を中止することにより処理の高速化を図ることができますが、照明の変動による許容値が低くなります。
※ＳｏｆｔＶＰでは、機能的な互換性はありますが、本機能による高速化はできません。

（1）しきい値の設定

しきい値とモードの設定はSetCorrBreakThrコマンドで行います。
実際に演算を途中で打ち切るための評価値である累積誤差しきい値は、

累積誤差しきい値＝　ｔｈｒ　×　テンプレート画素数

で計算されます。
また、ＳＳＤＡ法による自動しきい値変更モードもサポートしています。これは、常に最小をしきい値として、演算の打ち切りを行います。この場合、照明変動には弱くなります。

（2）相関演算打ち切りの設定

相関演算途中打切りを行う場合は、IP_Corrコマンドを実行する前にEnableCorrBreakコマンド、DisableCorrBreakコマンドで相関演算途中打ち切りモードを設定します。画像処理コマンドシステムの初期設定は、相関演算途中打ち切り無効になっています。

5.17 グラフィックス

グラフィックス処理では、文字列、図形描画等を行うことができます。本コマンドでの座標系は画像メモリアクセス（SYS_WIN）ウィンドウ相対であり、ウィンドウ始点が座標０となります。
以下にグラフィックス処理の実行手順フローを示します。

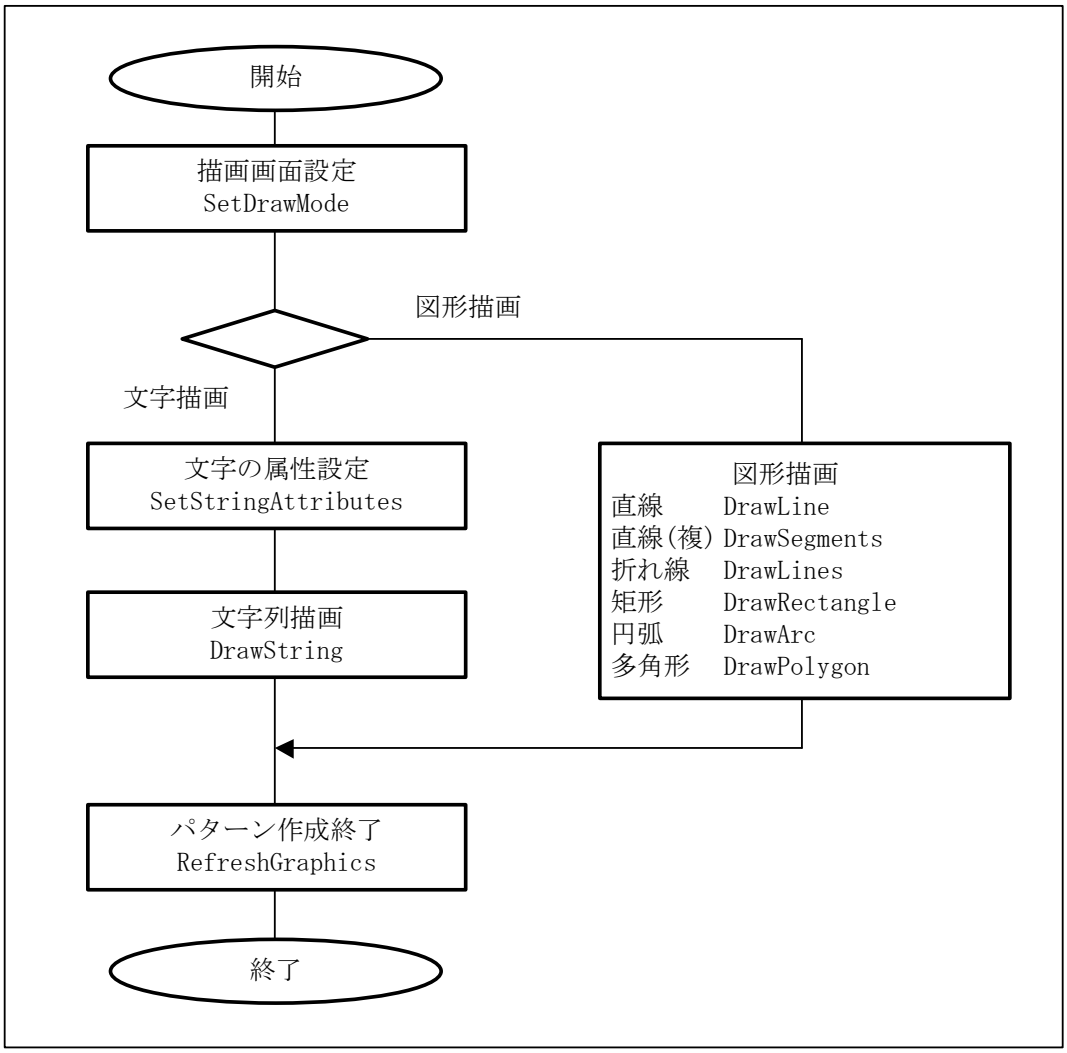


図5-17-1　グラフィックス処理の実行手順

5.18 線分化

画像処理コマンドでは線分化コマンドを用意しています。線分化は、２値画像の物体に対して線分列外周座標を抽出し、線分列外周座標から特徴量の抽出を行います。

線分列外周座標の抽出は、ExtractPolylineコマンドで行います。ExtractPolylineコマンドでは、指定画面の探索開始座標(sx,sy)からラスタースキャンで指定した濃度の探索対象物体の外周探索開始座標を取得し、取得した外周探索開始座標から時計回りに探索対象物体の外周を探索して、線分列外周座標を取得します。

対象物体の特徴量は、PolyArea、PolyPerm、PolyGrav, PolyFeatureコマンドで抽出します。

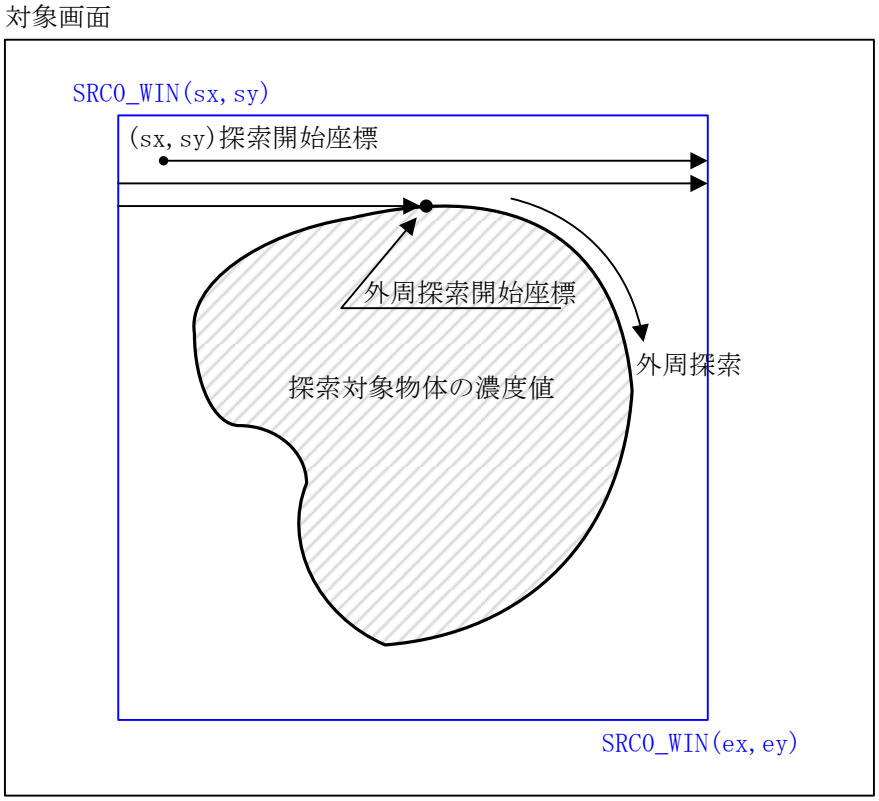


図5-18-1　線分化処理

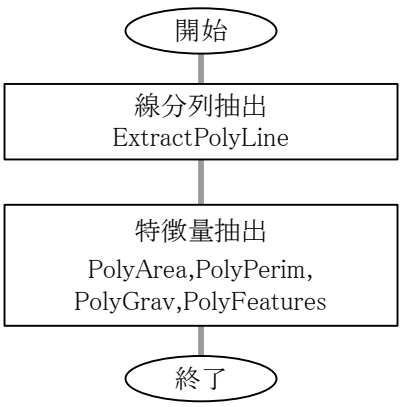


図5-18-2　線分化処理フロー

5.19 2値画像の穴埋め

画像処理コマンドでは穴埋めコマンドを用意しています。穴埋めは、２値画像の物体に対してラベリング処理を行い、指定した面積しきい値の条件を満たすオブジェクト（穴）を塗りつぶす処理を行います。

下の図は、対象画面中の穴埋め対象物が”白”オブジェクト、穴が”黒”オブジェクトの例です。
穴埋め処理は、”黒”オブジェクトに対してラベリング処理を行い、面積が最小面積以上最大面積以下のものを塗りつぶします。
尚、穴埋め処理はラベリング処理を用いているため、２５５個以上の穴埋めはできません。

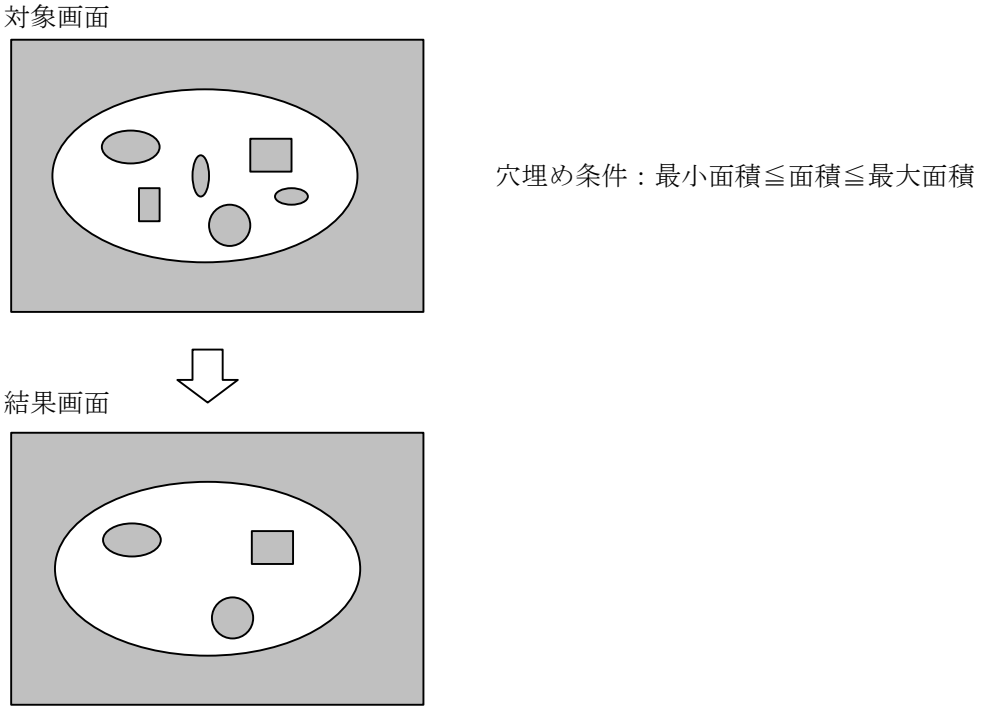


図5-19-1　２値画像穴埋め処理

5.20 正規化相関（VP－910A互換）

5.20.1 テンプレートデータ領域管理コマンド

正規化相関サーチを行う際、テンプレート特徴量データ領域確保コマンドにより、必ずテンプレート特徴量データ領域を確保しなければなりません。

（1）テンプレートデータについて

トレーニングによりテンプレートが登録されると正規化相関サーチに必要なデータはテンプレート番号（テンプレートＩＤ）で管理されます。
正規化相関サーチのテンプレートのデータは、テンプレート特徴量と画像が必要です。トレーニングを行うと指定された画像からテンプレート特徴量を計算し、テンプレート特徴量データ領域にセーブされます。指定された画像は、そのまま画像メモリにあり、その画面番号とウィンドウのデータだけがテンプレート特徴量データ領域にセーブされます。つまり、テンプレート特徴量データと画像データは別々の領域にあるということで、特に画像データは誤って別の画像に書換えられる可能性があります。テンプレートデータ内の特徴量データと画像メモリの内容が異なると正常にサーチできなくなるのでテンプレートに指定する画面の管理には十分注意が必要です。

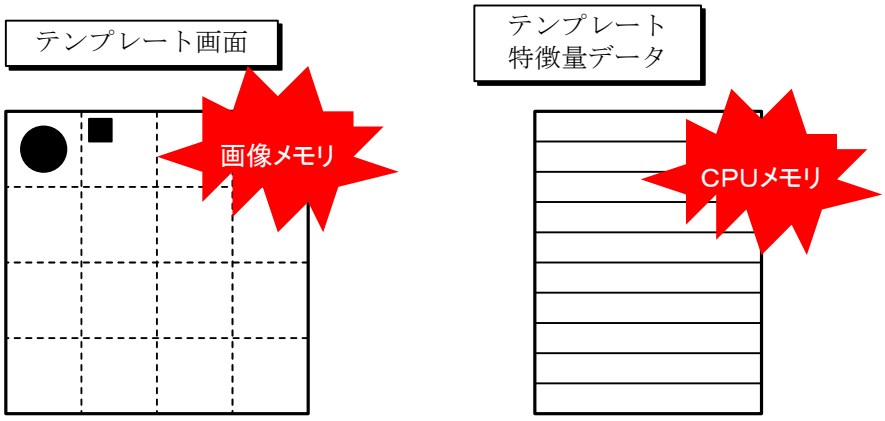


図5-20-1　テンプレートデータ

5.20.2 セットアップとトレーニング

正規化相関サーチは、セットアップ、トレーニング、サーチという３つのステップで実行されます。ここでは、セットアップとトレーニングのコマンドについて説明します。

（1） セットアップと画像の切出し

正規化相関サーチのセットアップとは、サーチを行う際のテンプレートを準備する操作のことです。
テンプレートとして登録したい画像を入力画像から切出したり、目的とする画像を画像描画コマンドで描画したりしてテンプレートを用意します。
正規化相関サーチでは、セットアップコマンドとして特別に用意しているものはありません。画像転送、画像縮小コマンドで画像をテンプレート画面として確保した画面に画像を切出し（転送）して下さい。また、画像描画コマンドを使用して目的の図形の画像をテンプレート画面に直接描画するか、別の画面に描画したものを画像転送、画像縮小コマンドでその画像を切出して下さい。

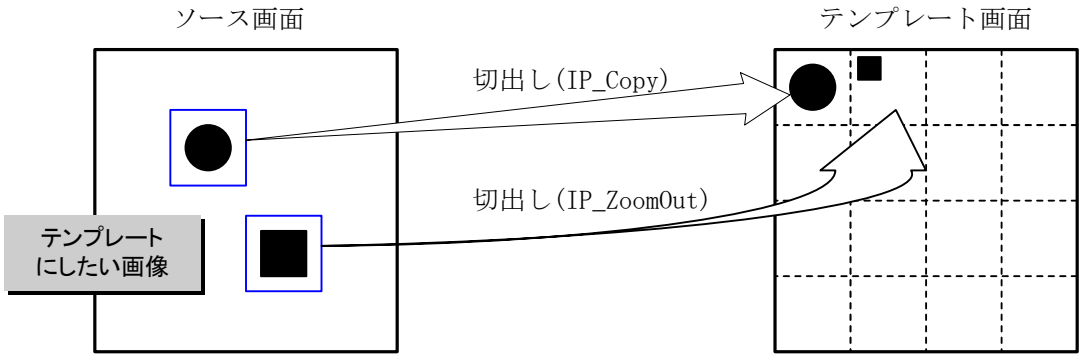


図5-20-2　画像の切出し

(2) テンプレートの情報量と処理時間

正規化相関サーチは、画像処理プロセッサとオンボードＣＰＵにより行っています。画像処理プロセッサではテンプレートカーネル内の積和演算を行います。オンボードＣＰＵでは、積和演算の結果から相関値を計算し、テンプレートカーネルを移動します。

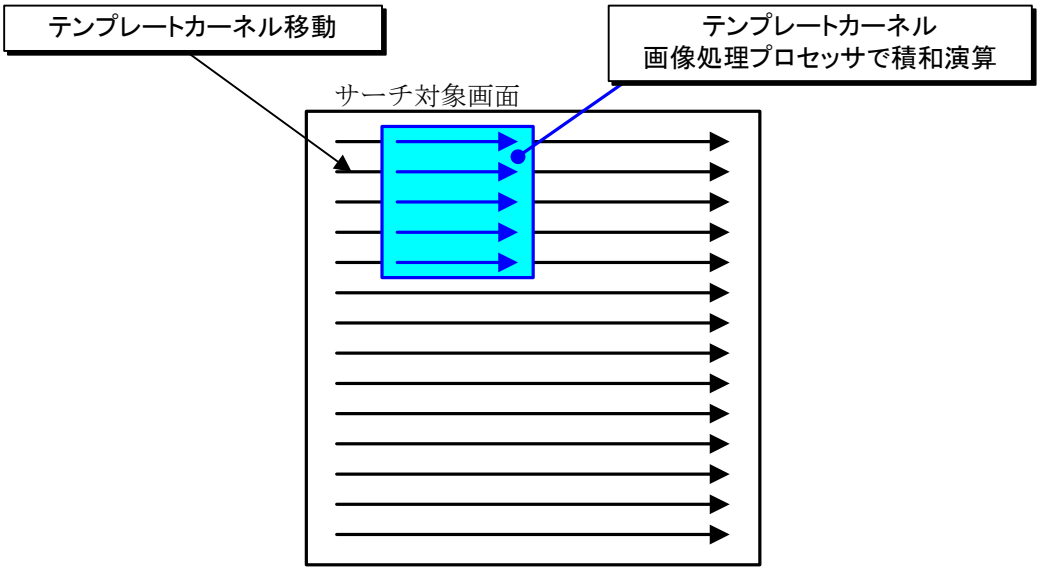


図5-20-3　正規化相関処理

画像処理プロセッサは、テンプレートカーネル内の積和演算を約７ｎＳ／画素で実行します。そしてオンボードＣＰＵでテンプレートカーネルを移動しながら、それぞれのポイントでの相関値を求め、最大の相関値とその座標を求めるわけです。ですから、サーチ時間はハードのオーバヘッド等を無視して単純に計算すると

サーチ時間 ＝ ７ｎＳ × テンプレートの大きさ × カーネルの移動回数

の式で計算できます。例えば、テンプレートの大きさが１２８×１２８画素で５１２×４８０のサーチ対象画面をサーチするとすると

7e-9 ×（128 × 128）×（512 × 480）= 28.1秒

２８秒もかかることになります。
処理時間を縮めるにはテンプレートの大きさを小さくするかカーネルの移動回数を減らせばいいことになります。
画像処理コマンドでは、テンプレートに関しては、テンプレートの大きさを小さくする代わりに情報量を減らすことで等価的にテンプレートの大きさを小さくできるようにしています。情報量を減らすということは、サーチのときのカーネル内の積和演算のデータのサンプリングをハード的に間引いて実行するということです。また、カーネルの移動も間引いて実行できるようになっています。
上記の例で、テンプレート情報量を縦８画素、横８画素間引き、カーネル移動を縦８画素、横８画素間引くと

7e-9 ×（128/8 × 128/8）×（512/8 × 480/8）= 0.007秒

２８秒の処理が４０９６分の１の７ミリ秒（実際には、レジスタアクセスのオーバヘッドで１０ｍＳ程度）でできることになります。
また、このときのテンプレートの情報量は

128/8 × 128/8 = 256

になります。

このように、実際の処理は情報量を減らして高速化を図ります。

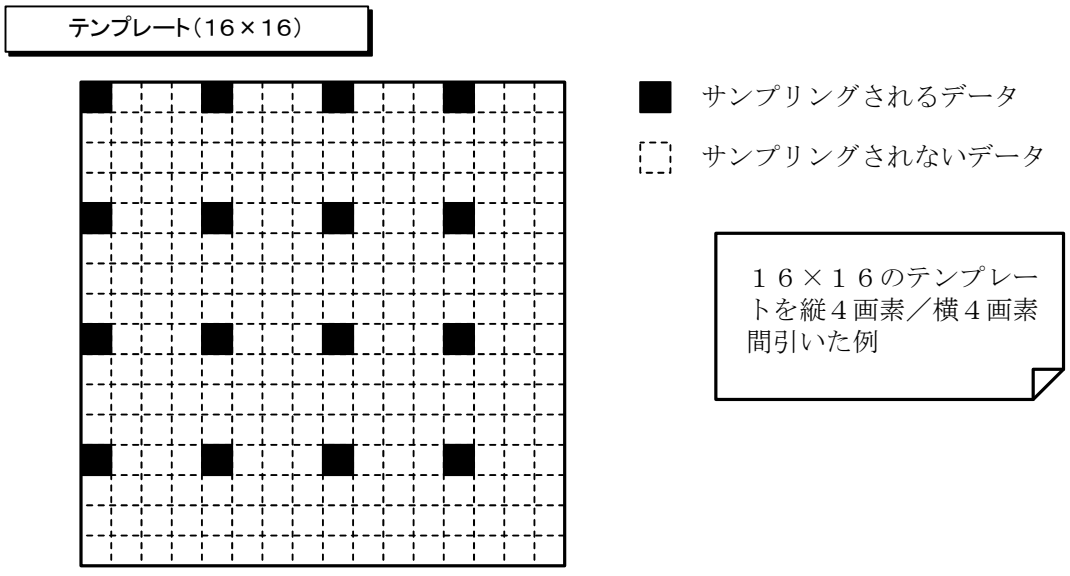


図5-20-4　テンプレートの間引き

(3) テンプレートマスクの設定

テンプレート画像は矩形領域しか設定できません。しかし、テンプレート画像として矩形以外の形の領域を設定したい場合があります。そこで、テンプレート画像にはマスク（不感帯）領域を設定できる機能があります。テンプレート画像のマスク設定された領域は、ハード処理される積和演算で計算の対象から除外されます。

設定したいマスク領域のテンプレート画像の値を「０」にすることでその領域がマスク領域になります。

通常は、円等の簡単な図形でマスクしますが、複雑なマスクも設定可能です。

下図にマスクの設定例を示します。

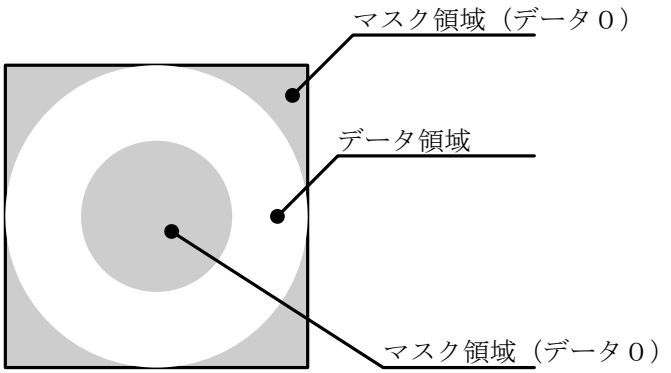


図5-20-5　マスクの設定例

次にマスクの設定手順を説明します。まず、テンプレートを切出し、テンプレートのマスクしたい領域に「０」を書込みます。

次にEnableCorrMaskコマンドでマスクモードを有効にします。

そして、SetCorrTemplateまたは、SetCorrTemplateExtコマンドでトレーニング（テンプレートの登録）を行います。

マスクを解除する場合は、DisableCorrMaskコマンドでマスクモードを解除して下さい。

マスクなしのテンプレートに戻したい場合は、テンプレート画像をもう一度切出し、トレーニングを行って下さい。

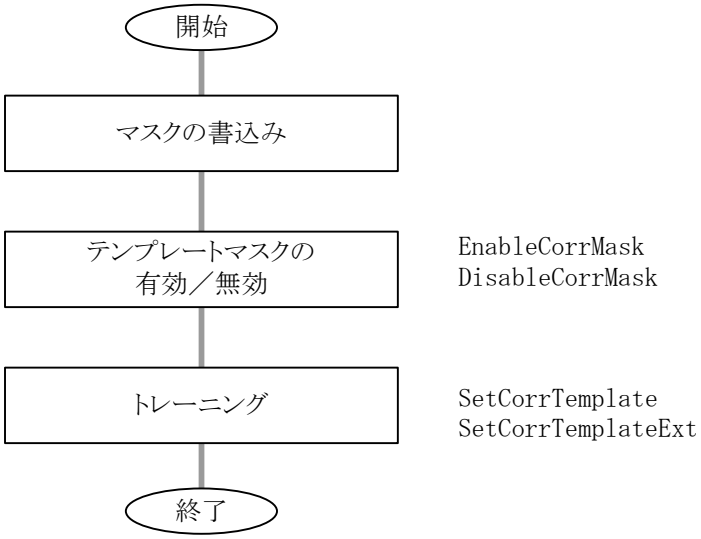


図5-20-6　テンプレートマスクの設定フロー

5.20.3 正規化相関サーチ

正規化相関サーチは、セットアップ、トレーニング、サーチという３つのステップで実行されます。ここでは、サーチのコマンドについて説明します。

（1） 高速サーチの方法

サーチコマンドには、vpxIP_CorrStep , vpxIP_CorrPoint , vpxIP_CorrPrecise コマンドがあります。
サーチコマンドは、テンプレートカーネルを移動しながら相関値を演算し最大の相関値と座標を見つけ出す処理を行っています。
最も単純なサーチでは、前項で説明しているように処理時間がユーザの要求に応えられるものではありません。そのため、テンプレートカーネルの間引きとサーチの間引きを行うわけです。前項ではテンプレートカーネルの間引きについて説明しましたが、ここではテンプレートカーネルを移動する時の間引きについてと高速化の方法を説明します。

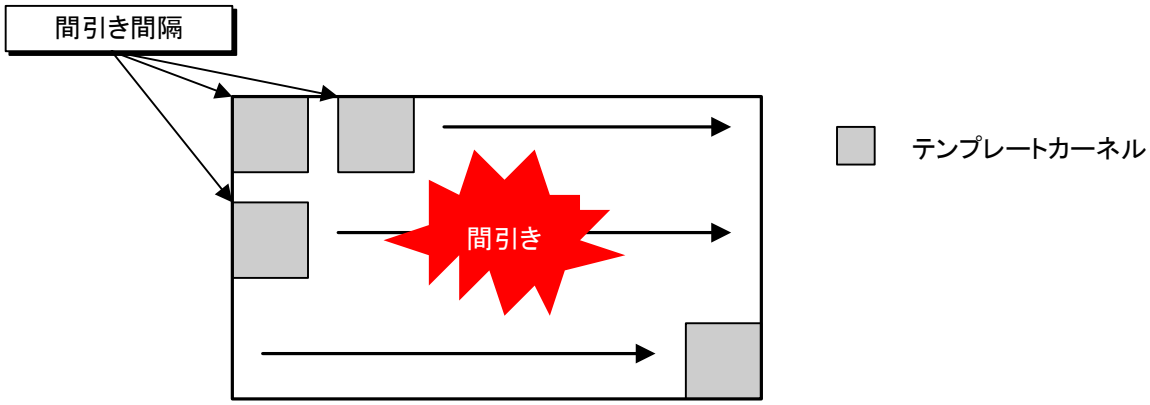


図5-20-7　vpxIP_CorrStepコマンド

vpxIP_CorrPointコマンドは、入力された座標の近傍領域でサーチを行います。通常は、vpxIP_CorrStep コマンドで間引いたぶんの補間をするためのサーチとして使用します。

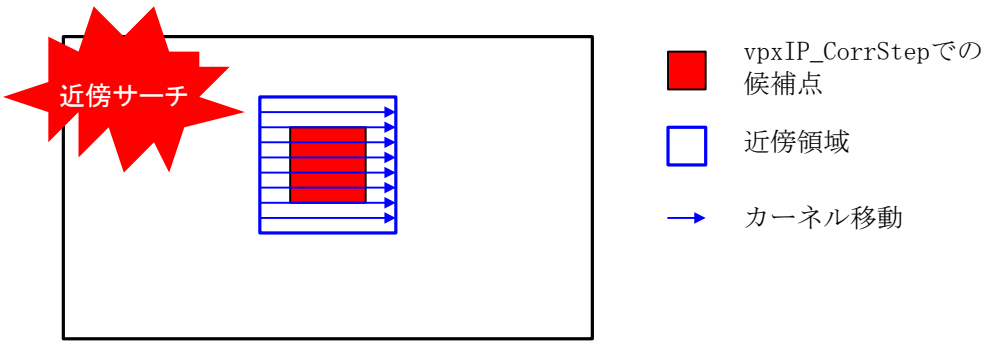


図5-20-8　vpxIP_CorrPointコマンド

vpxIP_CorrPrecise コマンドは、入力された座標の近傍領域の相関値からサブピクセルの位置を計算します。
通常は、vpxIP_CorrPoint コマンドで得られた最終位置座標を vpxIP_CorrPrecise コマンドに入力します。

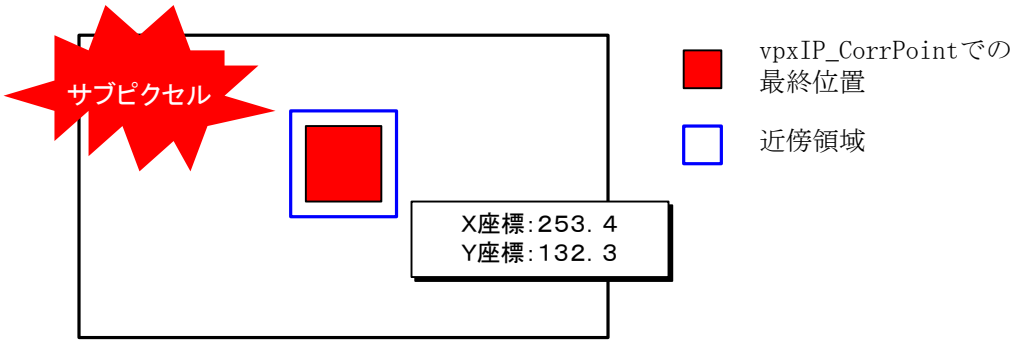


図5-20-9　vpxIP_CorrPreciseコマンド

（2） 並列演算カーネルによるサーチの高速化

画像処理プロセッサは、正規化相関演算用の積和演算器を１６組内蔵しています。その並列演算カーネルを使用して１つのテンプレートカーネルに対して１６組（４×４）の積和演算を並列に実行することができます。
並列演算カーネルを使用すると処理時間は原理的に並列演算カーネルを使わない時とくらべ１／１６になります。
また、原理的に４×４カーネルの間隔は通常は１画素ですが、テンプレートカーネルの間引きを行うと４×４カーネルの間隔はテンプレートカーネルの間引き間隔になります。
※ＳｏｆｔＶＰでは、機能的な互換性はありますが、本機能による高速化はできません。

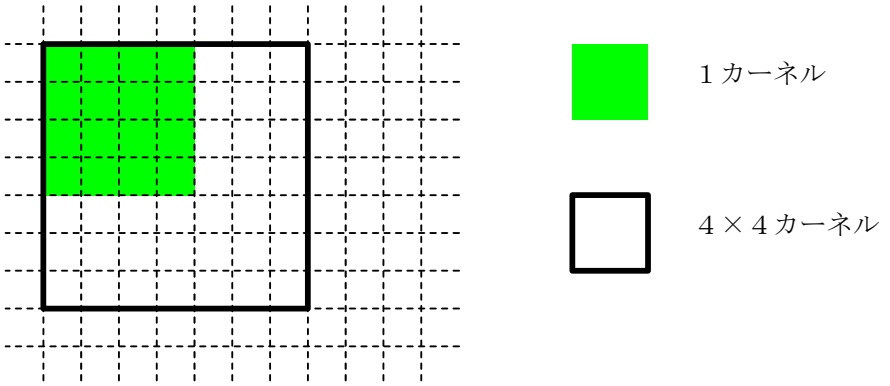


図5-20-10　４×４カーネル

サーチの高速化には上記の並列演算カーネルを使用する方法とテンプレートとサーチ移動の間引きによる方法がありますが、両方の方法を使用することで画像処理ボードの最大の性能を引き出すことができます。　しかし、並列演算カーネルとテンプレートとサーチ移動の間引きを同時に使用するには以下に示す条件を満足しなければなりません。

① サーチ移動の間引き間隔がテンプレートカーネルの間引き間隔と同じ
　または、その整数分の１
　または、その２倍

①の条件は、テンプレートカーネルの間引きを行うと４×４カーネルの間隔はテンプレートカーネルの間引き間隔になることに起因しています。
vpxIP_CorrStep ， vpxIP_CorrPoint コマンドは、上記①の条件のとき自動的に並列演算カーネルモードで処理を実行します。

（3） 相関演算途中打切りによるサーチの高速化

正規化相関の演算として画像処理プロセッサによりテンプレートカーネル内の積和演算を行っていますが、その他にもう一つ差分累積演算を行う演算器を持っています。これは、テンプレートカーネル内の画像と差分累積演算を行い累積誤差が指定閾値よりも大きくなったとき、正規化相関の積和演算の処理を途中で打切るためのものです。正規化相関の積和演算の処理を途中で打切ることにより、正規化相関サーチの高速化を図ることができます。
サーチ対象画像がテンプレート画像とあきらかに異なる場合、相関演算実行中に累積誤差が非常に大きくなると考えられます。そこで、この累積誤差がある閾値を超えた時点で、ミスマッチと判断し、途中で演算を中止します。
途中で演算を中止することにより処理の高速化を図ることができますが、照明の変動による許容値が低くなるので、注意して下さい。

※ＳｏｆｔＶＰでは、機能的な互換性はありますが、本機能による高速化はできません。

● 閾値の設定

閾値とモードの設定は vpxSetCorrBreak コマンドで行います。
実際に演算を途中で打ち切るための評価値である累積誤差閾値は、
累積誤差閾値　＝　ｔｈｒ × テンプレート画素数
で計算されます。
また、ＳＳＤＡ法による自動閾値変更モードもサポートしています。これは、常に最小を閾値として、演算の打ち切りを行います。この場合、照明変動にはかなり弱くなりますので十分に注意して下さい。

● 相関演算打切りの設定

相関演算途中打切りを行う場合は、vpxIP_CorrStep , vpxIP_CorrPoint コマンドを実行する前に vpxEnableCorrBreak , vpxDisableCorrBreak コマンドで相関演算途中打切りモードを設定して下さい。
画像処理コマンドシステムの初期設定は、相関演算途中打切り無効になっています。

（4） サーチ除外領域設定とマッチング候補点

vpxIP_CorrStep コマンドでは、テンプレートカーネルを移動しながら相関値の計算を行い、マッチングポイントを探します。通常は、間引いてサーチを行うので、探し出したポイントは正確なマッチングポイントではありません。その正確ではないマッチングポイントをマッチング候補点と呼びます。
vpxIP_CorrStep コマンドでは、探し出すマッチング候補点の個数を任意に設定できるようになっています。しかし、複数のマッチング候補点をサーチすると、近傍領域にその候補点が集中します。
そこで、マッチング候補点がその近傍領域に集中しないようにサーチ除外領域を設定します。サーチ除外領域として設定した距離以内にマッチング候補点が２つ以上存在しないようにポイントを探します。
サーチ除外領域を設定／解除は、vpxSetSearchDistance コマンドで行います。

5.21 直線抽出

画像処理コマンドではハフ変換による直線・矩形抽出コマンドを用意しています。以下にその使用方法を説明します。

5.21.1 ハフ変換による直線抽出

画像処理コマンドでは、ハフ変換を用いて離散的な座標データから直線を抽出します。ハフ変換の演算はオンボードＣＰＵで高速に処理します。
また、直線抽出コマンドでは下図のようにρ－θで示される直線が抽出されます。

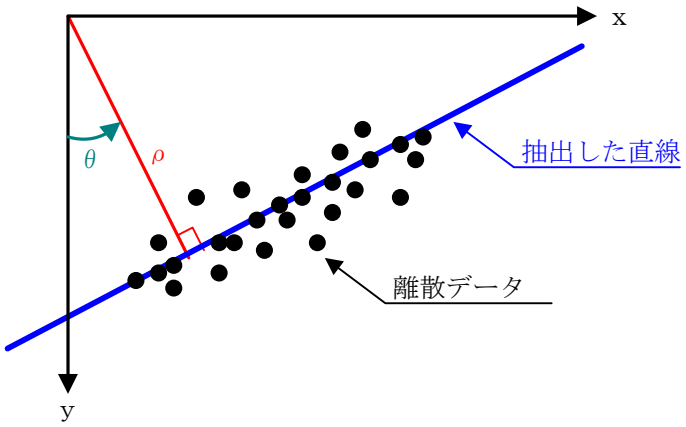


図5-21-1　離散データからの直線抽出

離散データからの直線抽出では、まず、上の図に示すような離散データの座標を２値化した画像からIP_ProjectBORegionX ， IP_ProjectBORegionY コマンドなどの座標抽出を行い抽出します。
次に、その座標データ群からハフ変換によりρ－θで示される直線を抽出します。

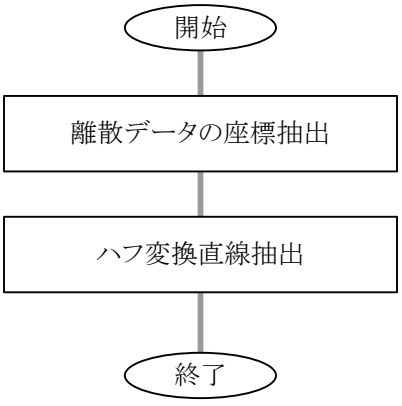


図5-21-2　離散データからの直線抽出フロー

5.21.2 ハフ変換直線からの矩形算出

画像処理コマンドでは、ハフ変換で抽出されたρ－θで示される４つの直線からそれぞれの交点座標を算出し、矩形の頂点座標や角度等を算出することができます。

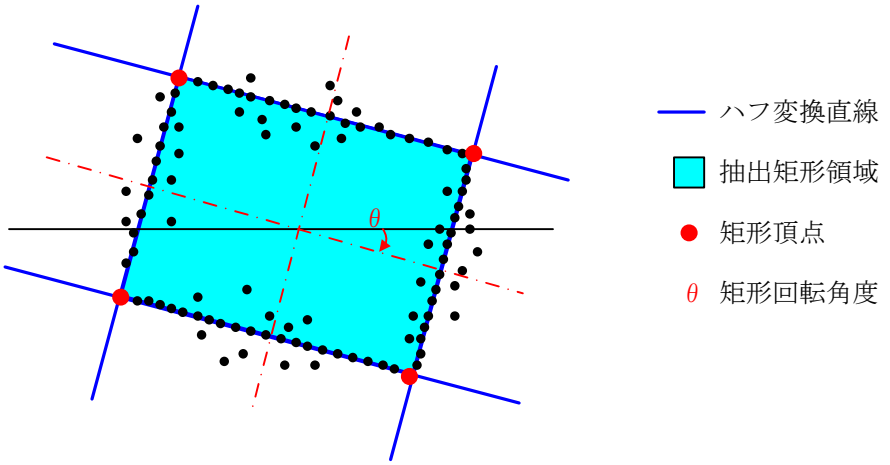


図5-21-3　ハフ変換直線からの矩形算出

GetCrossPoint,GetRectPoint,GetRectCenter,GetAnglePoint4,GetAnglePoint2コマンドにより、ハフ変換で抽出されたρ－θで示される２つまたは４つの直線からそれぞれの交点座標の算出、矩形の頂点座標や角度等の算出を行うことができます。

5.22 イメージキャリパ

画像処理コマンドではキャリパコマンドを用意しています。キャリパとはノギスのことで、画像処理により対象物のエッジを検出し、その対象物のエッジで平行なペアをサーチし、２つのエッジ間の距離やその中心位置を計測することができます。
以下にその使用方法を説明します。

5.22.1 イメージキャリパによる寸法計測

以下にイメージキャリパコマンドによる寸法計測のフローを示します。例は、集積回路（ＩＣ）パッケージのリード幅やリード間隔を求める場合です。

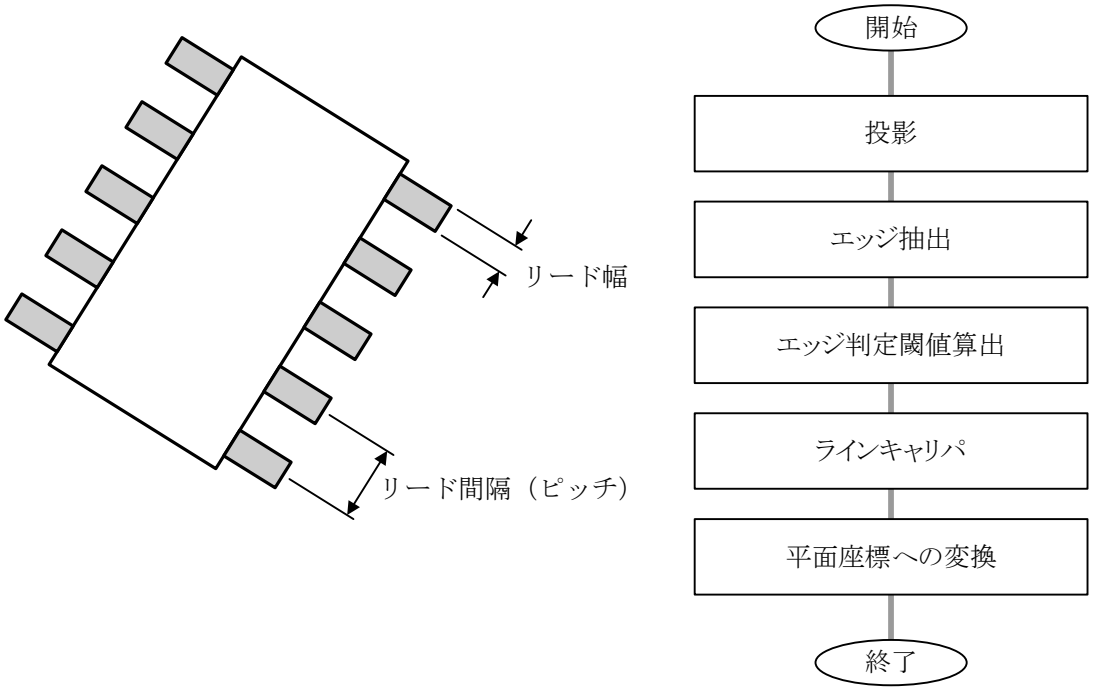


図5-22-1　寸法計測フロー

まず、ProjectLineコマンドにより、ＩＣのリードに沿って投影を行い、平面のデータをラインのデータに変換します。次にLineEdgeFilter,LineEdgeFilterExtコマンドによりエッジ抽出を行います。エッジデータからGetCaliperScoreコマンドでエッジ判定閾値を算出し、LineCaliperコマンドでエッジ判定閾値に従い、アップエッジとダウンエッジのペアをサーチします。そして、CaliperLPtoSPコマンドによりラインデータを平面のデータに変換して処理が完了します。そのデータからリード幅とリード間隔を計算します。

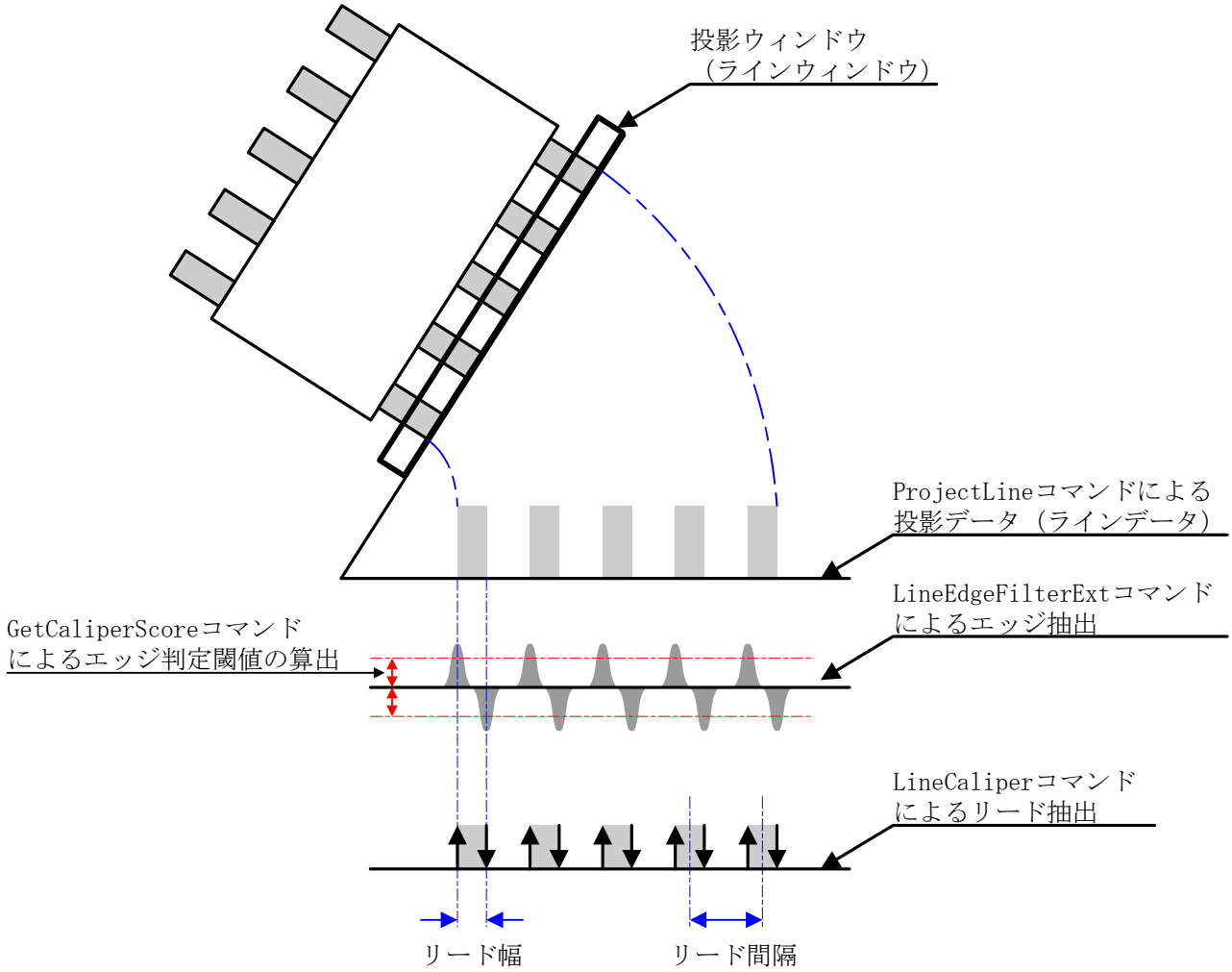


図5-22-2　寸法計測方法

5.22.2 ラインウィンドウについて

ProjectLineコマンドで投影を行う場合やCaliperLPtoSPコマンドによりラインデータを平面のデータに変換する場合、投影を行う領域を指定する必要があります。その場合の領域は、ラインウィンドウにより指定します。通常の画像処理コマンドのウィンドウは、傾斜した矩形領域を設定することはできませんが、ProjectLineコマンドでは、ラインウィンドウにより傾斜した矩形領域を設定し、傾斜した領域を投影することができます。

以下に、ラインウィンドウの構造体を示します。

```
typedef struct {
    short     sx;       // 開始点Ｘ座標
    short     sy;       // 開始点Ｙ座標
    short     ex;       // 終了点Ｘ座標
    short     ey;       // 終了点Ｙ座標
    short   leng;       // 投影幅
    short    opt;       // オプション（未使用）
} LINEWINDOW;
```

次に、ラインウィンドウの例を挙げながら構造体の各メンバーについて説明します。

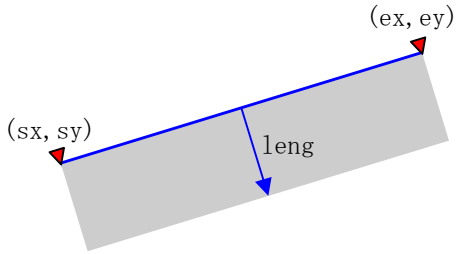


図5-22-3　ラインウィンドウ

ラインウィンドウでは、(sx, sy)と(ex, ey)で結ばれる直線で投影領域を指定し、lengで投影する画素数を指定します。(sx, sy)と(ex, ey)で結ばれる直線上のポイントが投影の開始ポイントになり、直線上のポイントに沿って(sx, sy)と(ex, ey)で結ばれる直線と垂直な方向にlengで指定される画素数分画素データの濃度累積が行われなす。

また、lengは(sx, sy)と(ex, ey)で結ばれる直線の状態と符号により濃度累積する方向が変わります。

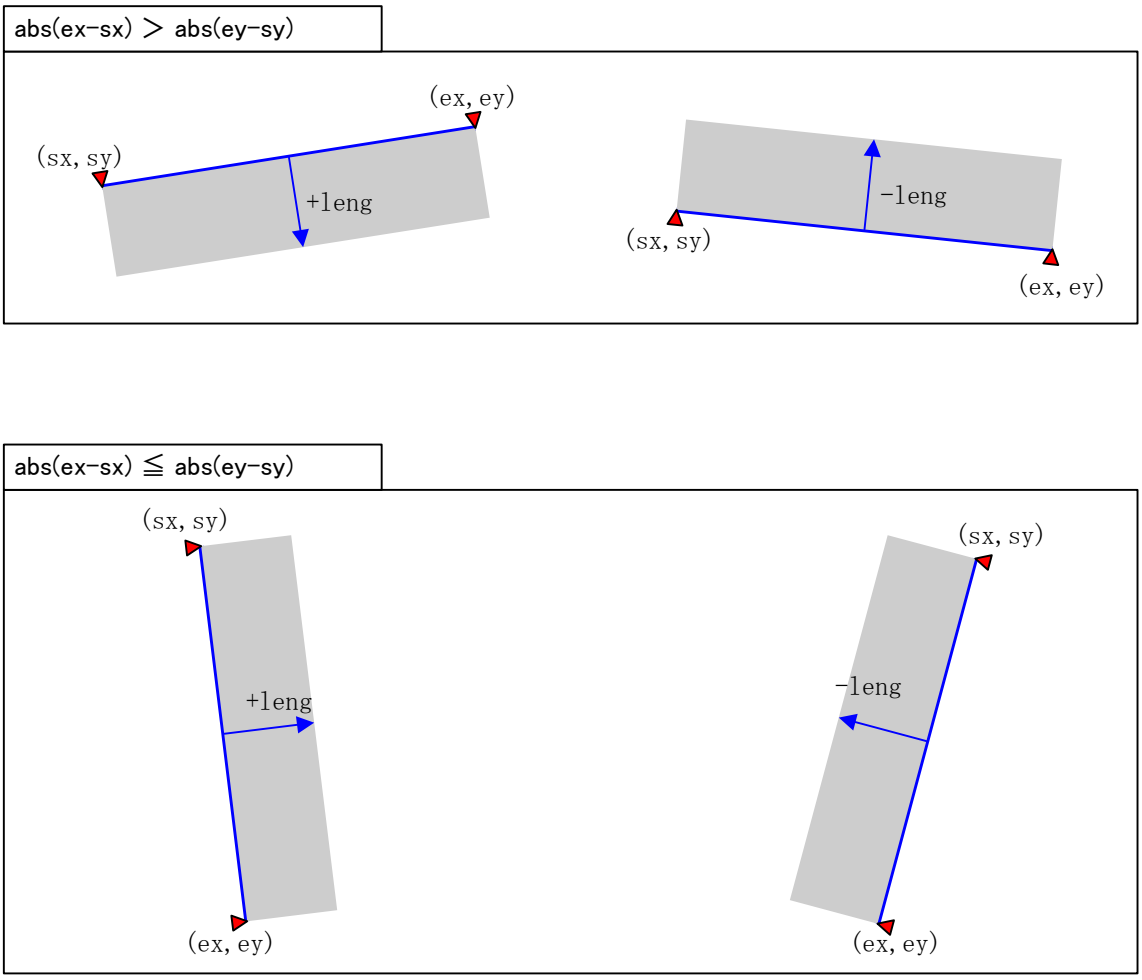


図5-22-4　lengの符号

5.23 エッジファインダ

画像処理コマンドではエッジファインダコマンドを用意しています。イメージキャリパでは画像処理により対象物のエッジを検出し、その対象物のエッジで平行なペアをサーチするため、エッジのペアがないパターンでは、エッジを抽出することはできません。そこで、エッジファインダーではその対象物に含まれるあらゆるエッジを解析し、位置、ポラリティ、レベルといった情報を抽出します。

5.23.1 ラインエッジファインダのエッジ抽出

以下にラインエッジファインダコマンドによるエッジ抽出のフローを示します。処理手順は基本的にイメージキャリパコマンドと同じですので、そちらも参照して下さい。

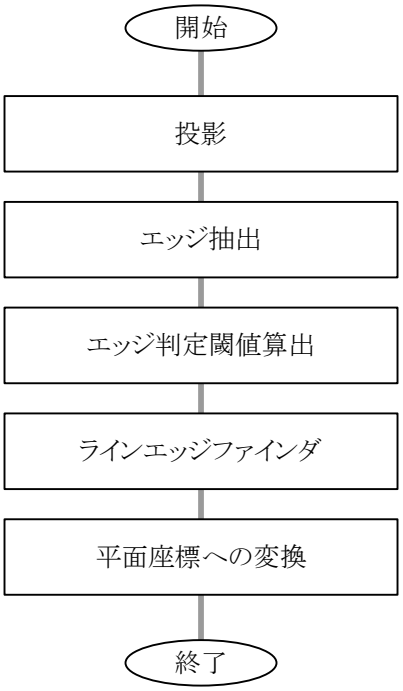


図5-23-1　エッジ抽出フロー

5.24 RGBLUT変換

ＲＧＢＬＵＴ変換は、２４ビットのＲＧＢカラー画像を濃度変換テーブル(ＬＵＴ)により、８ビットの画像に変換する処理です。濃度変換のパターンを換えることにより、ＲＧＢカラー画像から特定カラーを抽出したり、色相、彩度、明度を抽出する等、様々な処理が可能です。

画像処理コマンドでは、ＲＧＢＬＵＴ変換情報をＲＧＢＬＵＴオブジェクトに格納し、ＲＧＢＬＵＴハンドルで管理します。あらかじめ、ＲＧＢＬＵＴオブジェクトにＲＧＢＬＵＴ変換情報を設定し、コマンドを実行することによりＲＧＢＬＵＴ変換を行います。ＲＧＢＬＵＴオブジェクトの構成とＲＧＢＬＵＴ変換の手順を以下に示します。

5.24.1 RGBLUT変換手順

ＲＧＢＬＵＴ変換するには、まず、CreateRGBLUTコマンドでＲＧＢＬＵＴオブジェクトを生成します。次に、OpenRGBLUTコマンドでＬＵＴをアクセス可能にしてから、ＬＵＴに変換データを設定します。ＬＵＴへは、直接、変換データを書き込むか、あらかじめ用意されているマクロ(5.24.3 参照)で設定します。ＬＵＴの設定が終了したら、CloseRGBLUTコマンドでＬＵＴのアクセスを終了します。ＲＧＢＬＵＴ変換は、ConvertRGBLUTまたはIP_ConvertRGBLUTExコマンドを実行します。最後にDeleteRGBLUTコマンドでＲＧＢＬＵＴオブジェクトを削除して処理を終了します。

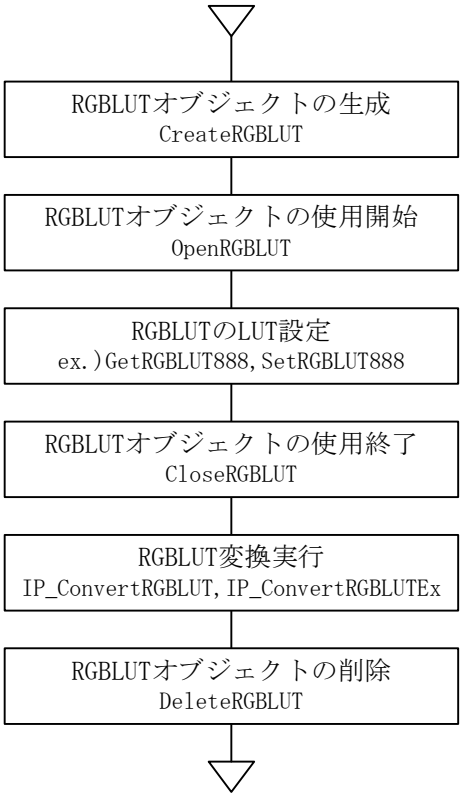


図5-24-1 ＲＧＢＬＵＴ変換手順

5.24.2 RGBLUTオブジェクト

ＲＧＢＬＵＴオブジェクトはCreateRGBLUTコマンドでオンボードＣＰＵ上に生成され、ＲＧＢＬＵＴハンドルで管理します。ＲＧＢＬＵＴオブジェクトの構成を以下に示します。

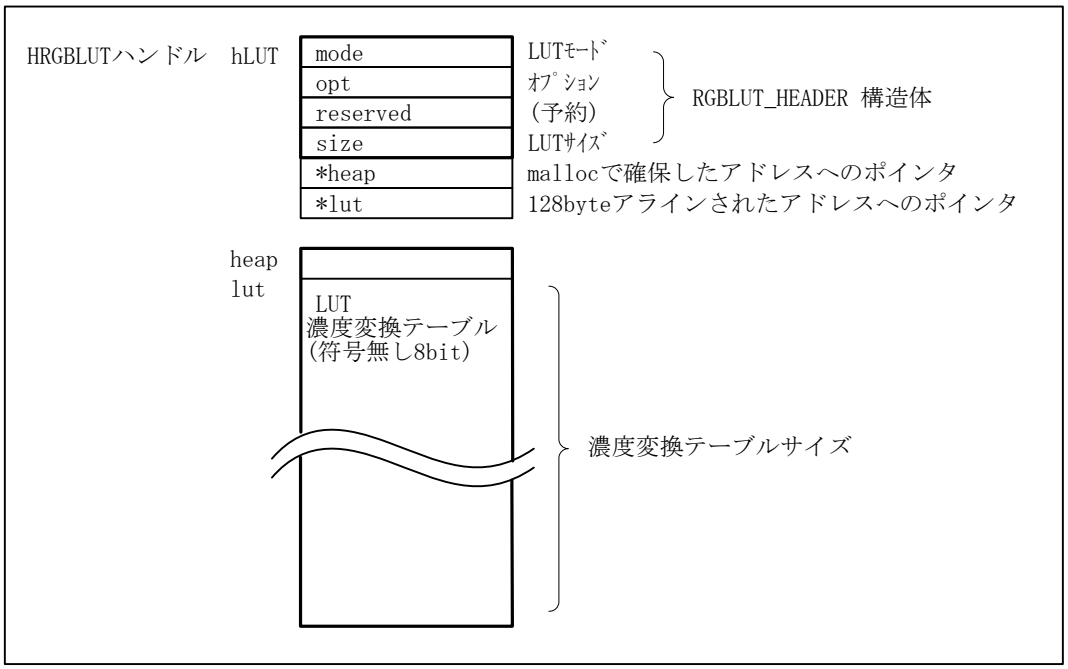


図5-24-2　ＲＧＢＬＵＴオブジェクト構成

ＬＵＴモード(mode)は以下の通りで、濃度変換処理と濃度変換テーブル(ＬＵＴ)の大きさを決定します。例えば、RGBLUT888は、Ｒデータが０～２５５(８bit)、Ｇデータが０～２５５(８bit)、Ｂデータが０～２５５(８bit)のＲＧＢカラー画像を濃淡画像０～２５５(８bit)に変換するモードで、CreateRGBLUTコマンド実行時に16,777,216byteの変換テーブルを確保します。

表5-24-1　ＬＵＴモード

ＬＵＴモード	定数値	内　容
RGBLUT888	0	R:8bit / G:8bit / B:8bit (16,777,216byte)
RGBLUT777	1	R:7bit / G:7bit / B:7bit (2,097,152byte)
RGBLUT666	2	R:6bit / G:6bit / B:6bit (262,144byte)
RGBLUT887	3	R:8bit / G:8bit / B:7bit (8,388,608byte)
RGBLUT878	4	R:8bit / G:7bit / B:8bit (8,388,608byte)
RGBLUT788	5	R:7bit / G:8bit / B:8bit (8,388,608byte)
RGBLUT787	6	R:7bit / G:8bit / B:7bit (4,194,304byte)
RGBLUT2CH88	7	R:8bit / G:8bit (65,536byte)
RGBLUT565	8	R:5bit / G:6bit / B:5bit (65,536byte)

5.24.3 濃度変換テーブル

濃度変換テーブル(ＬＵＴ)は、OpenRGBLUTコマンドを実行して、ＬＵＴサイズと先頭アドレスを取得し、直接アクセスすることが可能です。ＬＵＴへは、直接データを書き込むか、あらかじめ「ipxdef.h」に定義されている以下のマクロを用いて設定します。ここで、lutはＬＵＴの先頭アドレス、ｒ、ｇ、ｂはそれぞれＲＧＢ画像のＲデータ、Ｇデータ、Ｂデータ、ｄは変換データです。マクロ定義から分かるように、ｒ、ｇ、ｂ値がＬＵＴのアドレスを示すパラメータになります。
ＬＵＴの設定が終了したら、必ず、CloseRGBLUTコマンドでＬＵＴのアクセスを終了してください。

ＬＵＴへのデータ書き込みマクロ

```
#define SetRGBLUT888(lut,r,g,b,d) lut[r << 16 | g << 8 | b]= (d)
#define SetRGBLUT777(lut,r,g,b,d) lut[(r & 0xfe) << 13 | (g & 0xfe) << 6 | b >> 1]= (d)
#define SetRGBLUT666(lut,r,g,b,d) lut[(r & 0xfc) << 10 | (g & 0xfc) << 4 | b >> 2]= (d)
#define SetRGBLUT887(lut,r,g,b,d) lut[r << 15 | g << 7 | b >> 1]= (d)
#define SetRGBLUT878(lut,r,g,b,d) lut[r << 15 | (g & 0xfe) << 7 | b]= (d)
#define SetRGBLUT788(lut,r,g,b,d) lut[(r & 0xfe) << 15 | g << 8 | b]= (d)
#define SetRGBLUT787(lut,r,g,b,d) lut[(r & 0xfe) << 14 | g << 7 | b >> 1]= (d)
#define SetRGBLUT2CH88(lut,r,g,d) lut[r << 8 | g]= (d)
#define SetRGBLUT565(lut,r,g,b,d) lut[(r & 0xf8) << 8  | (g & 0xfc) << 5 | b >> 3]= (d)
```

ＬＵＴからのデータ読み出しマクロ

```
#define GetRGBLUT888(lut,r,g,b)   lut[r << 16 | g << 8 | b]
#define GetRGBLUT777(lut,r,g,b)   lut[(r & 0xfe) << 13 | (g & 0xfe) << 6 | b >> 1]
#define GetRGBLUT666(lut,r,g,b)   lut[(r & 0xfc) << 10 | (g & 0xfc) << 4 | b >> 2]
#define GetRGBLUT887(lut,r,g,b)   lut[r << 15 | g << 7 | b >> 1]
#define GetRGBLUT878(lut,r,g,b)   lut[r << 15 | (g & 0xfe) << 7 | b]
#define GetRGBLUT788(lut,r,g,b)   lut[(r & 0xfe) << 15 | g << 8 | b]
#define GetRGBLUT787(lut,r,g,b)   lut[(r & 0xfe) << 14 | g << 7 | b >> 1]
#define GetRGBLUT2CH88(lut,r,g)   lut[r << 8 | g]
#define GetRGBLUT565(lut,r,g,b)   lut[(r & 0xf8) <<  8 | (g & 0xfc) << 5 | b >> 3]
```

5.24.4 RGBLUT濃度変換

ＲＧＢ画像のＬＵＴ変換は、IP_ConvertRGBLUTコマンド、または、IP_ConvertRGBLUTExコマンドで実行します。以下にＬＵＴモードに対する演算内容を示します。
ここで、ImgSrc(R)、ImgSrc(G)、ImgSrc(B)は、それぞれソース画像となるＲＧＢ画像のＲ、Ｇ、Ｂデータ、ImgDstはデスティネーション画像に設定されるデータです。

表5-24-2　ＬＵＴモードと演算内容

LUTモード	演算内容
RGBLUT888	ImgDst = LUT[ImgSrc(R) << 16 | ImgSrc(G) << 8 | ImgSrc(B)]
RGBLUT777	ImgDst = LUT[(ImgSrc(R) & 0xFE) << 13 | (ImgSrc(G) & 0xFE) << 6 | ImgSrc(B) >> 1]
RGBLUT666	ImgDst = LUT[(ImgSrc(R) & 0xFC) << 10 | (ImgSrc(G) & 0xFC) << 4 | ImgSrc(B) >> 2]
RGBLUT887	ImgDst = LUT[ImgSrc(R) << 15 | ImgSrc(G) << 7 | ImgSrc(B) >> 1]
RGBLUT878	ImgDst = LUT[ImgSrc(R) << 15 | (ImgSrc(G) & 0xFE) << 7 | ImgSrc(B)]
RGBLUT788	ImgDst = LUT[(ImgSrc(R) & 0xFE) << 15 | ImgSrc(G) << 8 | ImgSrc(B)]
RGBLUT787	ImgDst = LUT[(ImgSrc(R) & 0xFE) << 14 | ImgSrc(G) << 7 | ImgSrc(B) >> 1]
RGBLUT2CH88	ImgDst = LUT[ImgSrc(R) << 8 | ImgSrc(G)]
RGBLUT565	ImgDst = LUT[(ImgSrc(R) & 0xF8) << 8 | (ImgSrc(G) & 0xFC << 5) | (ImgSrc(B) << 3)]

5.24.5 RGBカラー抽出(色相・彩度・明度)

ＲＧＢＬＵＴ変換は、ＬＵＴのパターン設定により様々な変換をすることができます。
色の性質を色合い／色調(色相)、あざやかさ(彩度)、明るさ(明度)の３つの要素に分ける表現方法がありますが、ＲＧＢＬＵＴ変換を用いて、ＲＧＢカラー画像を色相(Hue)、彩度(Saturation)、明度(Intensity)の画像に変換することが可能です。以下に色相、彩度、明度の関係を示した図と画像変換の演算式を示します。

・色相 (Hue)
　色を円周上に配列していると考え、色あい／色調を角度で表わしたものです。
　Hの範囲は 0～359 です。
・彩度 (Saturation)
　色のあざやかさを示します。Ｓの範囲は 0 ～255 です。
・明度 (Intensity)
　色の明るさを示します。Iの範囲は 0 ～255 です。

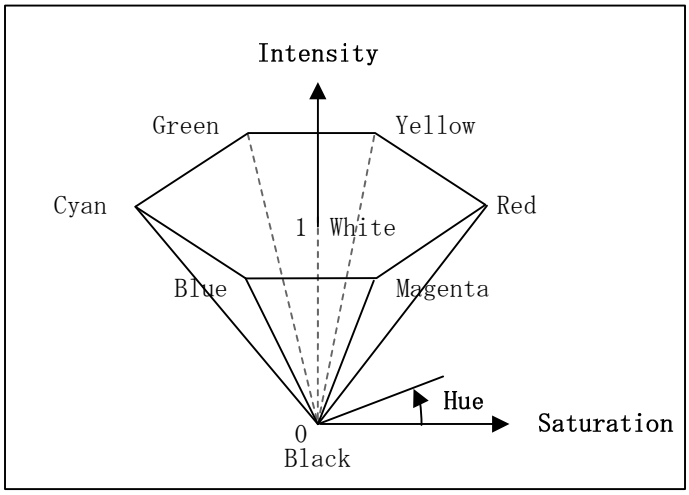


図5-24-3　色相、彩度、明度の関係

演算式	I= max(R,G,B) I= 0: S= 0: H= 0(不定) I≠ 0: i= min(R,G,B) S=(I- i)* 255/ I R= I: H=(G- B)* 60 /(I- i) G= I: H=(B- R)* 60 /(I- i)+ 120 B= I: H=(R- G)* 60 /(I- i)+ 240 (H < 0 の時は、H = H + 360 とする) H：Hue, S：Saturation, I:Intensity

画像メモリには、０ ～２５５ までしかデータを格納することができません。そこで、０～３６０度までのＨｕｅの値は０．７１ 倍するなどしてＬＵＴに設定してください。

以下に、ＲＧＢ画像から色相(Hue)画像へのＲＧＢＬＵＴ変換例を示します。

```
/* 画面確保 */
ImgRGB = AllocRGBImg( IMG_FS_512H_512V );
ImgID  = AllocImg( IMG_FS_512H_512V );

.....................................

/* RGBLUTオブジェクト作成 */
mode = RGBLUT888;
hLUT = CreateRGBLUT( mode , 0 );

/*********************************************
  ＲＧＢ画像から色相(Hue)画像へ変換
*********************************************/
/* RGBLUTのLUTアクセス可能 */
mode = 0;
OpenRGBLUT( hLUT, &size, &lut, mode );

/* RGBLUTのLUT設定[色相(Hue)] */
for( r = 0; r < 256; r++ ){                          r,g,b値(0～255)の組合せで
   for( g = 0; g < 256; g++ ){                       H(Hue)値が決定されます
      for( b = 0; b < 256; b++ ){
         I = max( max( b, g ), max( g, r ) );
         i = min( min( b, g ), min( g, r ) );
         if( I == 0 ){
            H = 0;
         }else{
            val = 60.0f / ( I - i );
            if( I == r ){
               H = ( g - b ) * val;
            }else if( I== g ){
               H = ( b - r ) * val + 120;
            }else{
               H = ( r - g ) * val + 240;
            }
            if( H < 0 ) H += 360;
            H *= 0.71f;
         }
         SetRGBLUT888( lut, r, g, b, H );            r,g,b値のときのH(Hue)値
      }                                              をLUTへ設定します
   }
}

/* RGBLUTのLUTアクセス禁止 */
CloseRGBLUT( hLUT, lut );

/* RGBLUT変換 */
IP_ConvertRGBLUT( ImgRGB, ImgID , 0 , hLUT);

.....................................

/* RGBLUTオブジェクト削除 */
DeleteRGBLUT( hLUT );

/* 画面解放 */
FreeImg( ImgID );
FreeImg( ImgRGB );
```

付録A コンパイラの設定

1. Visual Studio でのコンパイルとリンク

画像認識コマンドを使用する場合、ＳｏｆｔＶＰで用意しているインクルードファイルをＣのソースプログラムからインクルードすることと、画像認識コマンドのＤＬＬインポートライブラリをリンクする必要があります。

Microsoft Visual Studio 2005 Visual C/C++(VisualC)でプログラムをコンパイル、リンクする場合、下記の要領でインクルードファイルのディレクトリパスの設定とライブラリのプロジェクトへの追加を行って下さい。以下の”プロジェクト名”とは、ユーザが開発するプロジェクトの名称です。

1.1 インクルードファイルのディレクトリパスの設定

ユーザアプリケーションをコンパイルする場合、Visual Cでインクルードファイルのディレクトリパスを設定します。

［プロジェクト］メニューから［”プロジェクト名”のプロパティ］を選択します。［構成プロパティ］から［Ｃ／Ｃ＋＋］を選択後、[全般］を選択すると”追加のインクルードファイル”という項目が表示されます。ここに、インストールしたリモートコマンドライブラリのインクルードファイルディレクトリを入力します。インストール時にディレクトリの変更がなければ

C:¥SoftVP¥SDK¥VPSeries¥VC¥inc

と入力します。

1.2 ライブラリのプロジェクトへの追加

画像認識コマンドのＤＬＬインポートライブラリをリンクする場合、VisualC上でライブラリをプロジェクトに追加することでビルド時にリンクされます。

［プロジェクト］メニューから［”プロジェクト名”のプロパティ］を選択します。［構成プロパティ］から［リンカ］を選択後、［全般］を選択すると、”追加のライブラリディレクトリ”という項目が表示されます。ここに、リモートコマンドのＤＬＬインポートライブラリをインストールしたディレクトリを入力します。インストール時にディレクトリの変更がなければ

C:¥SoftVP¥SDK¥VPSeries¥VC¥lib

と入力します。

また、［プロジェクト］メニューから［”プロジェクト名”のプロパティ］を選択し、［構成プロパティ］から［リンカ］を選択後、［入力］を選択すると、”追加の依存ファイル”という項目が表示されます。ここに、以下のリモートコマンドのＤＬＬインポートライブラリを入力します。ここに、以下の画像認識コマンドのＤＬＬインポートライブラリ”ＩＰＳＣＭＤＳ．ＬＩＢ”と”ＩＰＳＣＴＬＳ．ＬＩＢ”を入力します。

表3-2-1 VisualCのプロパティ設定

設定項目	設定値
インクルードファイルのパス	[インストールフォルダ]¥SDK¥VPSeries¥VC¥inc 設定例： C:¥SoftVP¥SDK¥VPSeries¥VC¥inc
ライブラリファイルのパス	[インストールフォルダ]¥SDK¥VPSeries¥VC¥lib 設定例： C:¥SoftVP¥SDK¥VPSeries¥VC¥lib
ＤＬＬインポートライブラリ	IPSCMDS.LIB IPSCTLS.LIB

VPシリーズ互換 SoftVP
ユーザーズマニュアル（第三版）

開発元

マクセルシステムテック株式会社

設計部　〒992-0021 山形県米沢市花沢3091-6

営業部　〒244-0801 神奈川県横浜市戸塚区信濃町549-2三宅ビル

技術サポート窓口　URL http://www.systemtech.maxell.co.jp/
mail : vp-support@maxell.co.jp