CONTENTS

ビジュアルヘルプ – 画像処理（2）

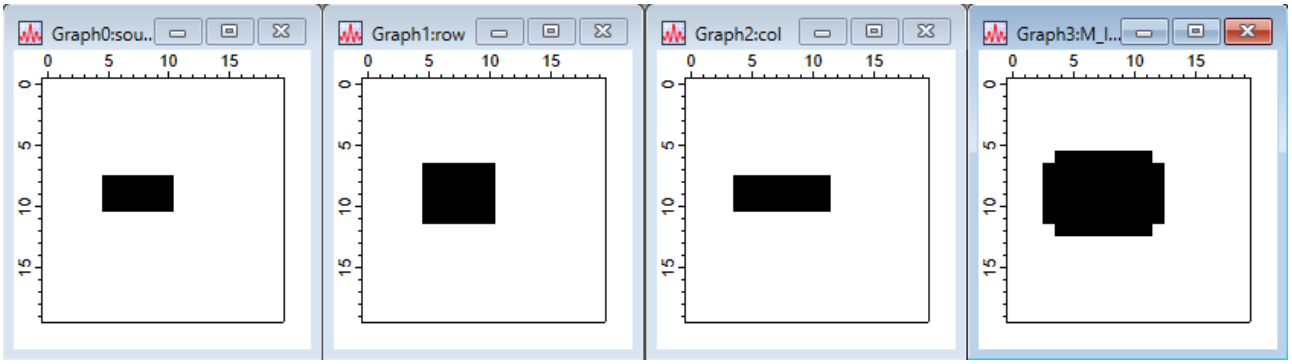
本ドキュメントの操作は、ほとんどの場合、Image Processing Tutorial エクスペリメントを使って行います。

形態学的コマンド

形態学的コマンドは、画像内の領域の形状と境界に影響を与えるツールです。
膨張（dilation）と収縮（erosion）から始まる典型的な形態学的コマンドは、画像と構造要素を伴います。
構造要素は通常、画像よりもはるかに小さいサイズです。
膨張は、構造要素をその原点を中心に反転させ、畳み込みマスクと同様の方法で使うことで行われます。
これは次の例で確認できます。

```
Make/B/U/N=(20,20) source=0
source[5,10][8,10]=255                    // ソースは塗りつぶされた長方形
NewImage source
Imagemorphology /E=2 BinaryDilation source    // 1×3 構造要素で膨張
Duplicate M_ImageMorph row
NewImage row                              // 膨張の結果を表示
Imagemorphology /E=3 BinaryDilation source    // 3×1 列による膨張を実行
Duplicate M_ImageMorph col
NewImage col                              // 列による拡張を表示
Imagemorphology /E=5 BinaryDilation source    // 円による膨張
NewImage M_ImageMorph                     // 円による膨張を表示
```

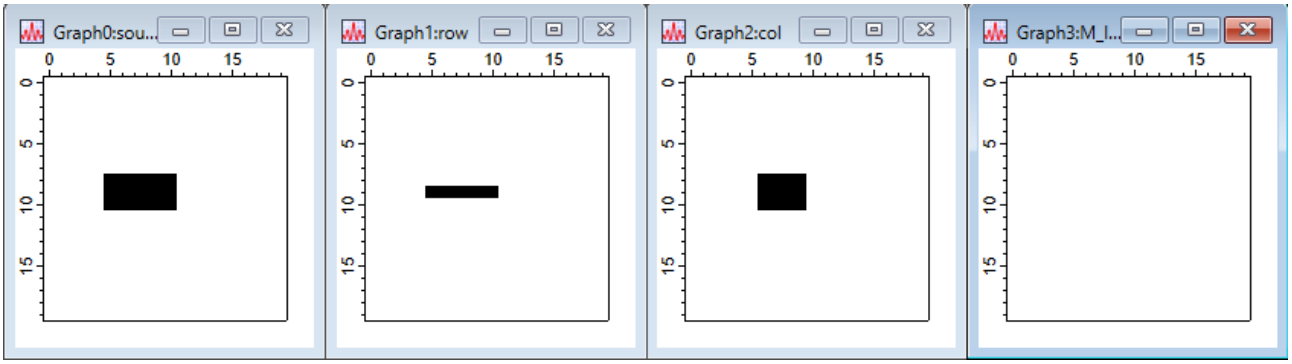
ソース　行　列　円



収縮の結果は、構造要素がその量だけ平行移動されてもなおその集合内に含まれるようなピクセル x, y の集合です。

```
Make/B/U/N=(20,20) source=0
source[5,10][8,10]=255                    // ソースは塗りつぶされた長方形
NewImage source
Imagemorphology/E=2 BinaryErosion source      // 1x3 構造要素で収縮
Duplicate M_ImageMorph row
NewImage row                              // 収縮の結果を表示
Imagemorphology /E=3 BinaryErosion source     // 3x1 列による収縮を実行
Duplicate M_ImageMorph col
NewImage col                              // 列による収縮を表示
Imagemorphology /E=5 BinaryErosion source     // 円による収縮
NewImage M_ImageMorph                     // 円による収縮を表示
```

ソース　　行　　列　　円

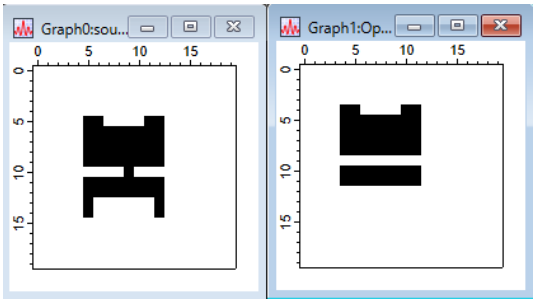


まず、円による収縮がすべてのソースピクセルを消去したことに注目します。
この結果が生じるのは、円構造要素が 5x5 の「円」であり、円がソース内に完全に収まるような x、y 方向のオフセットが存在しないためです。
行画像と列画像は、主に一方向に収縮が生じていることを示しています。
繰り返しますが、1x3 の構造要素（行の場合）がソースピクセル上を滑りながら収縮を生成する様子を想像してみてください。

次の形態学的演算のペアは、開操作と閉操作です。
機能的には、開操作はソース画像をある構造要素（例えば E）で収縮し、その結果を同じ構造要素 E で再び膨張させることに相当します。
一般的に開操作は平滑化効果を持ち、次の例に示すように小さな（細い）突起を除去します。

```
Make/B/U/N=(20,20) source=0
source[5,12][5,14] = 255
source[6,11][13,14] = 0
source[5,8][10,10] = 0
source[10,12][10,10] = 0
source[7,10][5,5] = 0
NewImage source
ImageMorphology /E=1 opening source     // 2x2 構造要素を使った開操作
Duplicate M_ImageMorph OpenE1
NewImage OpenE1
```

ソース　　Open 2x2



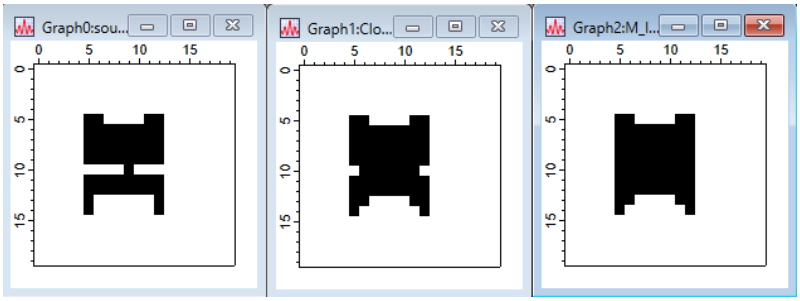
2x2 構造要素は上部と下部の領域を結ぶ細い接続部と、下部の２つの突起を除去しました。
一方、上部の２つの突起は十分に大きかったため、2x2 構造要素を生き延びました。

閉操作は、ソース画像の膨張処理に続いて、同じ構造要素を用いた収縮処理を行うことに相当します。

```
Make/B/U/N=(20,20) source=0
source[5,12][5,14] = 255
source[6,11][13,14] = 0
source[5,8][10,10] = 0
source[10,12][10,10] = 0
```

```
source[7,10][5,5] = 0
NewImage source
ImageMorphology /E=4 closing source     // 3x3 構造要素を使った閉操作
Duplicate M_ImageMorph CloseE4
NewImage CloseE4
ImageMorphology /E=5 closing source     // 5x5 構造要素を使った閉操作
NewImage M_ImageMorph
```

ソース **Close 3x3** **Close 5x5**

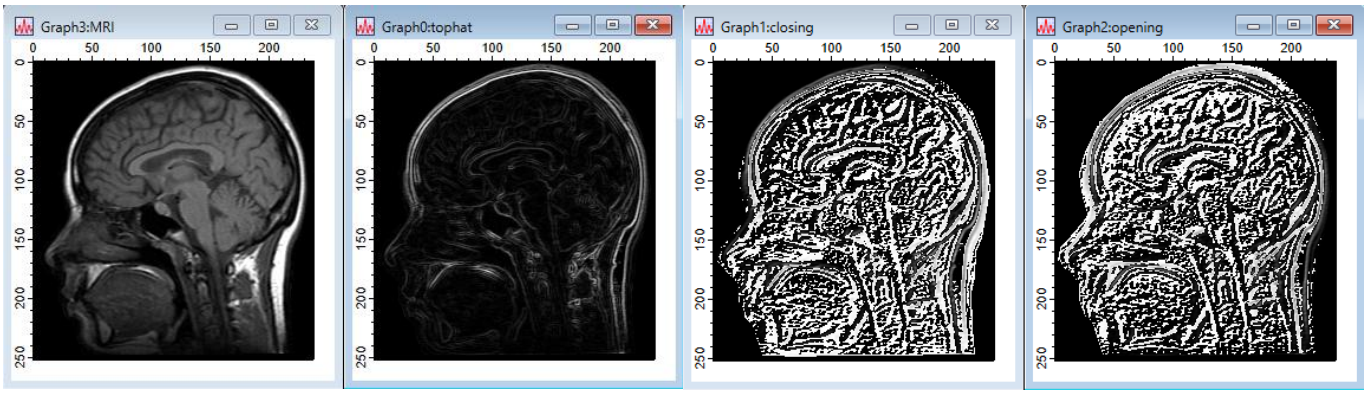


中央の画像は、3×3 構造要素を使った閉操作に対応します。
この要素は上部と下部の領域間の隙間を埋めるには十分な大きさに見えますが、上部と下部の突起間の隙間を埋めるには十分ではありません。
右側の画像は 5×5 の「円形」構造要素で作成されました。

トップハット形態学的演算には様々な解釈があります。
Igor 式トップハットは、収縮画像と膨張画像の差分を計算します。
その他の解釈には、画像自体とそのクロージングまたはオープニングの差分を計算する方法が含まれます。
次の例では、これらのバリエーションの一部を説明します。

```
Duplicate root:images:mri source
ImageMorphology /E=1 tophat source      // 2x2 構造要素を使った閉操作
Duplicate M_ImageMorph tophat
NewImage tophat
ImageMorphology /E=1 closing source     // 3x3 構造要素を使った閉操作
Duplicate M_ImageMorph closing
closing-=source
NewImage closing
ImageMorphology /E=1 opening source     // 3x3 構造要素を使った開操作
Duplicate M_ImageMorph opening
opening=source-opening
NewImage opening
```

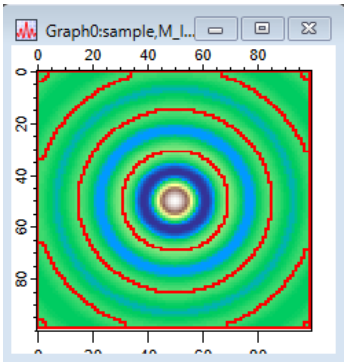
ソース **トップハット** **閉操作** **開操作**



４つの画像からわかるように、組み込みのトップハット処理は画像内の領域の境界（輪郭）を強調しますが、開操作または閉操作を施したトップハットは小さなグレースケールの変動を強調します。

次に示すように、流域コマンドは流域地域の境界を特定します。

```
Make/O/N=(100,100) sample
sample=sinc(sqrt((x-50)^2+(y-50)^2)/2.5)        // 同心円のように見える
ImageTransform/O convert2Gray sample
NewImage sample
ModifyImage sample ctab= {*,*,Terrain,0}        // 識別しやすくするために色を追加
ImageMorphology /N/L watershed sample
AppendImage M_ImageMorph
ModifyImage M_ImageMorph explicit=1, eval={0,65000,0,0}
```



流域コマンドにおいて /L フラグを省略すると、アルゴリズムが 8-connectivity ではなく 4-connectivity に従うため、誤った流域境界線が生じる可能性があることに注意してください。

画像の解析

画像処理と画像解析の区別はかなり微妙です。
純粋な解析操作は次の通りです：ImageStats、ラインプロファイル、ヒストグラム、HSL セグメンテーション、粒子解析。

ImageStats コマンド

ウェーブ全体の統計情報は、標準の WaveStats コマンドで取得できます。
ImageStats コマンドは特に 2D と 3D ウェーブに対応しています。
このコマンドでは、標準的な ROI ウェーブを使って完全に任意の ROI 領域を定義できます（「ROI での作業」のセクションを参照）。
特別なフラグ /M=1 を指定すると、ROI 領域内の最小値、最大値、平均値のみを知りたい場合に、高次モーメントの評価に必要な追加計算時間を省略して処理を高速化できます。
このコマンドは、ユーザー定義の適応アルゴリズムで動作するよう設計されています。

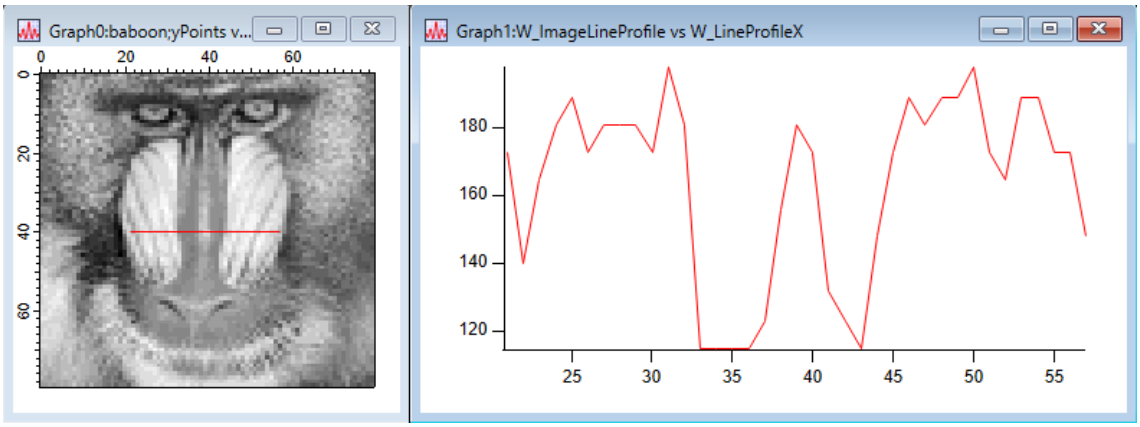
ImageStats は /P フラグを使って、3D ウェーブの特定の平面に対して処理を実行できます。

ImageLineProfile コマンド

ImageLineProfile コマンドは、任意の数の線分から成るパスに沿って画像をサンプリングします。
このコマンドを使うには、まず一対のウェーブを使ってパスの記述を作成する必要があります。
例を挙げます。

```
NewImage root:images:baboon                    // プロファイル対象の画像を表示
// 直線経路を表すウェーブのペアを作成
Make/O/N=2 xPoints={21,57}, yPoints={40,40}
AppendToGraph/T yPoints vs xPoints             // 画像上にパスを表示
```

```
// プロファイルを計算
ImageLineProfile xwave=xPoints, ywave=yPoints, srcwave=root:images:baboon
Display W_ImageLineProfile vs W_LineProfileX          // プロファイルを表示
```



任意の数のポイントで構成されるより複雑なパスを作成できます。
この場合、コマンドによって生成される W_LineProfileX と W_LineProfileY ウェーブを活用し、プロファイルを 3D パスプロットとして描画することをお勧めします（ヘルプ Path Plots を参照）。
より詳細な例については、Image Processing Tutorial エクスペリメントも参照してください。

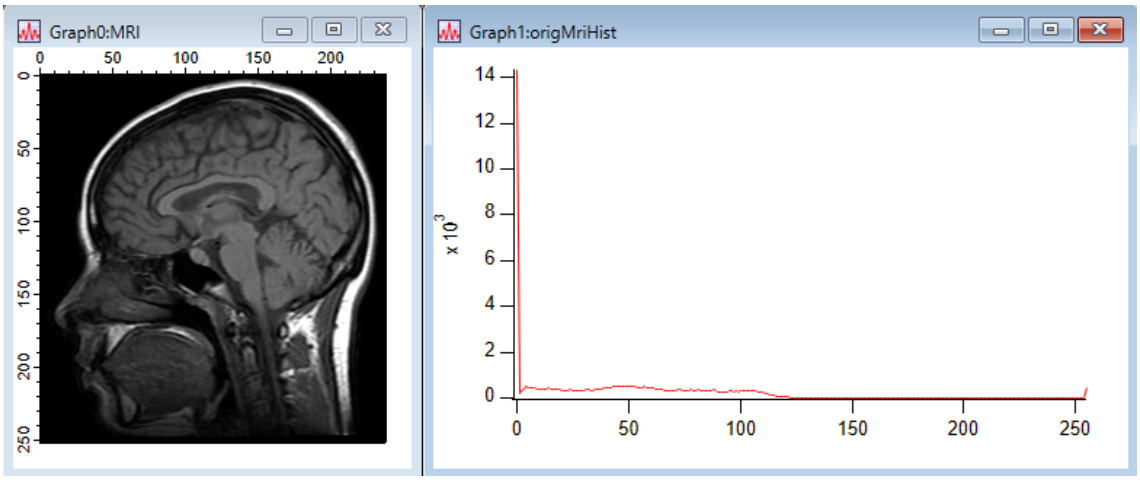
注記： 3 レイヤーを超える 3D ウェーブを扱う場合、ImageLineProfile/P=plane を使うことで、プロファイルを計算する平面を指定できます。

ウェーブから連続した配列データ（行または列）を抽出するためにラインプロファイルを使う場合、
ImageTransform の getRow または getCol を使ってデータを抽出する方が効率的です（速度が向上します）。

ヒストグラム

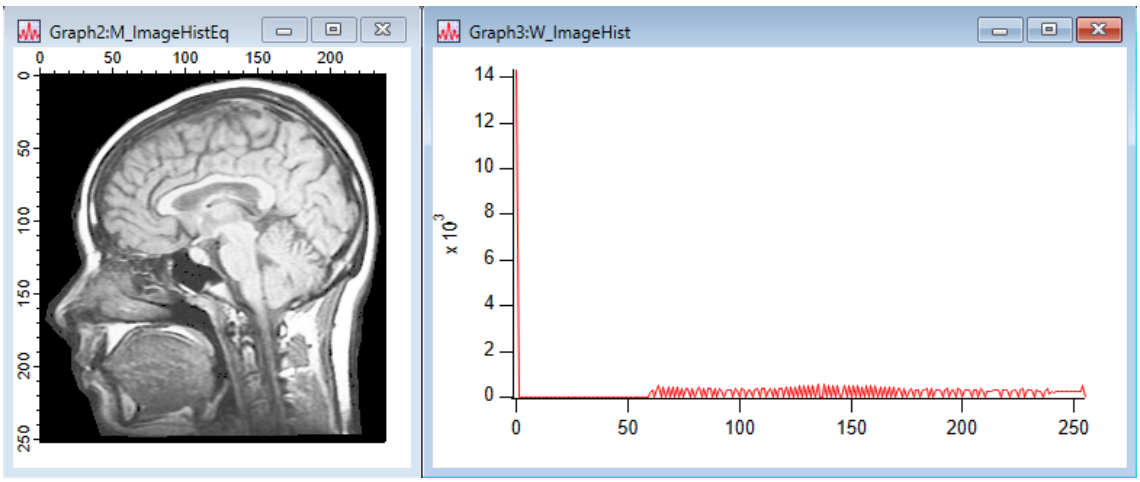
ヒストグラムは画像解析において非常に重要なツールです。
例えば、画像内の背景を自動検出する最も単純な手法は、ヒストグラムを計算し、最も高い頻度で出現するピクセル値を選択することです。
ヒストグラムは閾値値の決定や画像コントラストの強化においても極めて重要です。
次に画像ヒストグラムを使ったいくつかの例を示します。

```
NewImage root:images:mri
ImageHistogram root:images:mri
Duplicate W_ImageHist origMriHist
Display /W=(201.6,45.2,411,223.4) origMriHist
```



ヒストグラムから明らかなように、画像はかなり暗く、背景はほぼゼロである可能性が高いです。
125 を超えるピクセルの数が少ないことから、この画像はヒストグラム均等化に適した候補であると考えられます。

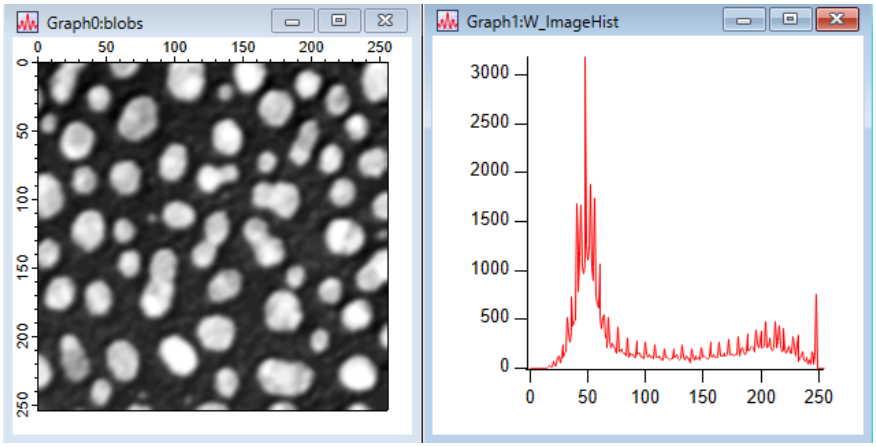
```
ImageHistModification root:images:mri
ImageHistogram M_ImageHistEq
NewImage M_ImageHistEq
AutoPositionWindow/E/M=1 /R=Graph0 $WinName(0,1)
Display /W=(201.6,45.2,411,223.4) W_ImageHist
```



２つのヒストグラムを比較すると、２つの特徴がすぐにわかります。
まず、暗い背景は 1 レベル（0）のみであるため変化がありません。
次に、主に 0 から 120 の値の範囲にあった画像の残りの部分が、57 から 255 の範囲に伸張されています。

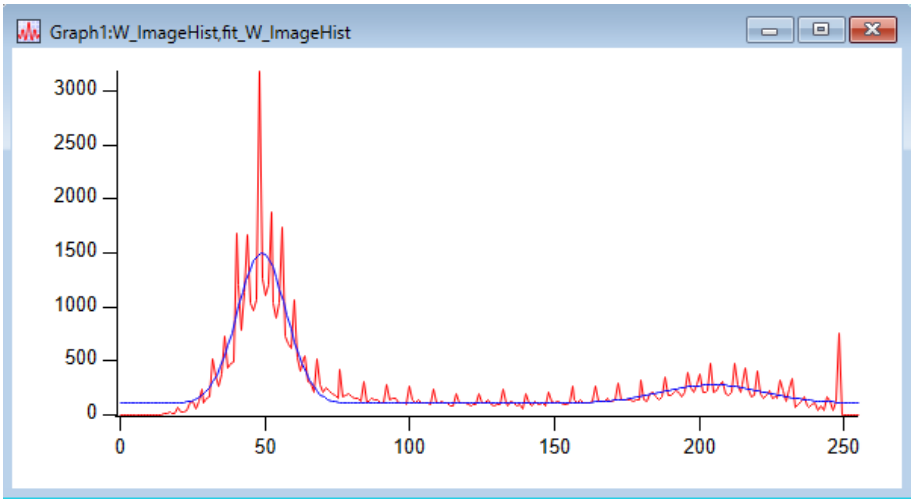
次の例は、ヒストグラム情報を使って閾値を決定する方法を示しています。

```
NewImage root:images:blobs
ImageHistogram root:images:blobs
Display /W=(201.6,45.2,411,223.4) W_ImageHist
```



結果のヒストグラムは明らかに二峰性です。
単なる興味本位で、これを２つのガウス分布にフィッティングさせてみます。

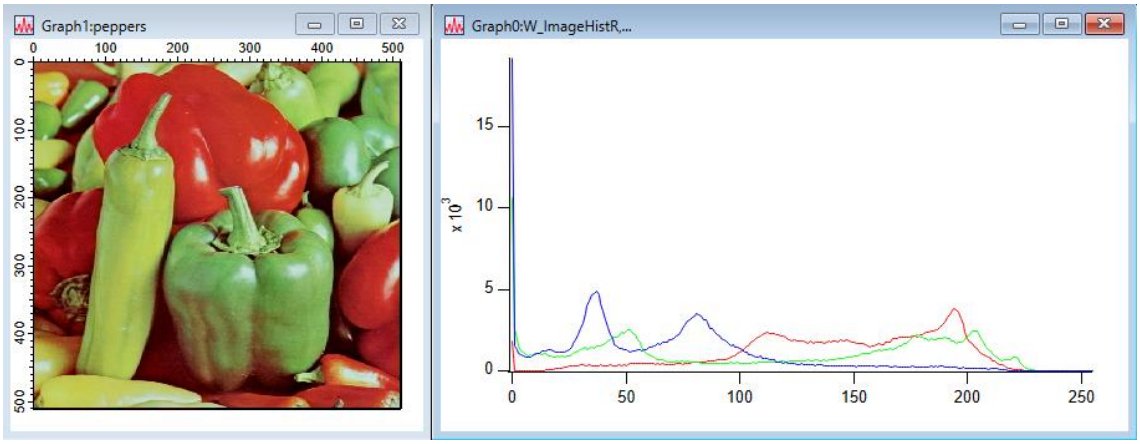
```
// ヒストグラムに基づいて係数ウェーブを推定
Make/O/N=6 coeff={0,3000,50,10,500,210,20}
Funcfit/Q twoGaussians,coeff,W_ImageHist /D
ModifyGraph rgb(fit_W_ImageHist)=(0,0,65000)
```



グラフに示されている曲線は、２つのガウス関数の和の関数近似です。
視覚的に確認しながら、２つのガウス関数の間の x 値を選択できます。
おそらく 100～150 の範囲内でしょう。
実際、前述した組み込みの閾値処理で同じ画像をテストすると、反復アルゴリズムが 125 を選択し、ファジーエントロピーが 109 を選択するなど、結果が異なることがわかります。

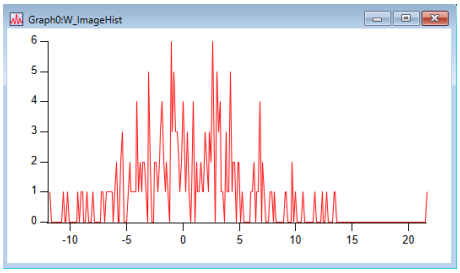
RGB または HSL 画像のヒストグラムは、各色チャンネルごとに個別のヒストグラムを生成します。

```
ImageHistogram root:images:peppers
Display W_ImageHistR,W_ImageHistG,W_ImageHistB
ModifyGraph rgb(W_ImageHistG)=(0,65000,0),rgb(W_ImageHistB)=(0,0,65000)
```



３レイヤー以上の 3D ウェーブのヒストグラムは、/P フラグでレイヤーを指定することで計算できます。
例えば：

```
Make/N=(10,20,30) ddd=gnoise(5)
ImageHistogram/P=10 ddd
Display W_ImageHist
```



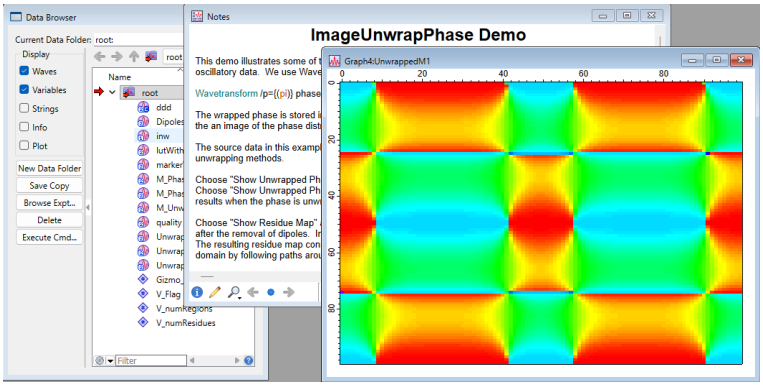
位相の展開

２次元における位相の展開は１次元よりも複雑です。
というのも、この操作の結果は展開経路に依存してはならないからです。

経路不変性とは、展開された領域内の閉曲線に対する経路積分がすべて消えることを意味します。
多くの場合、領域内には閉曲線経路積分が消えない点（残留点）が存在します。
反時計回りの経路積分が正の値を取る場合、残留値は正となります。
２次元で位相を展開する際、残留値は通常 ±1 となります。
これは、２つの反対の残留物が線（「分岐切断」と呼ばれます）で結ばれている場合、経路が分岐切断を横切らない任意の輪郭積分は消えることを示唆しています。
正と負の残留物が隣接している場合、それらは「双極子」を形成し、この双極子周囲の経路積分も消えるため除去可能です。
したがって、展開は、不均衡な残留物を囲まない経路、あるいは分岐切断線を横切らない経路を使って実行できます。

ImageUnwrapPhase コマンドは、残留物を無視する高速手法、または残留物を特定しそれらを迂回する経路を探索する低速手法のいずれかを使って２次元位相展開を実行します。
高速な方法は微分位相の直接積分を使います。
領域内に残留物が存在する場合、誤った結果を導く可能性があります。
遅い方法では、まず全ての残留物を特定し、分岐切断線を追加した内部ビットマップに描画します。
その後、ImageSeedFill で使われるアルゴリズムを繰り返し適用し、全てのピクセルが処理されるまで残留物と分岐切断線周囲の経路を取得します。
残留物と分岐切断線の分布によっては、データの領域が複数の領域に分割される場合があります。
各領域は分岐切断線またはデータ境界によって完全に囲まれています。
この場合、位相は個々の領域ごとに独立して計算され、その領域で最初に処理されたピクセルに基づくオフセットが適用されます。
残留物を計算するメソッドを使って ImageUnwrapPhase を使う場合、このコマンドによって変数 V_numResidues と V_numRegions が作成されることに注意してください。
結果の分析に有用な内部ビットマップのコピーも取得できます。

File→Examples Experiments→Analysis→ImageUnwrapPhase Demo は、異なる種類の残留物、分岐カット、および結果として得られる展開された位相を詳細に示す例を提供します。

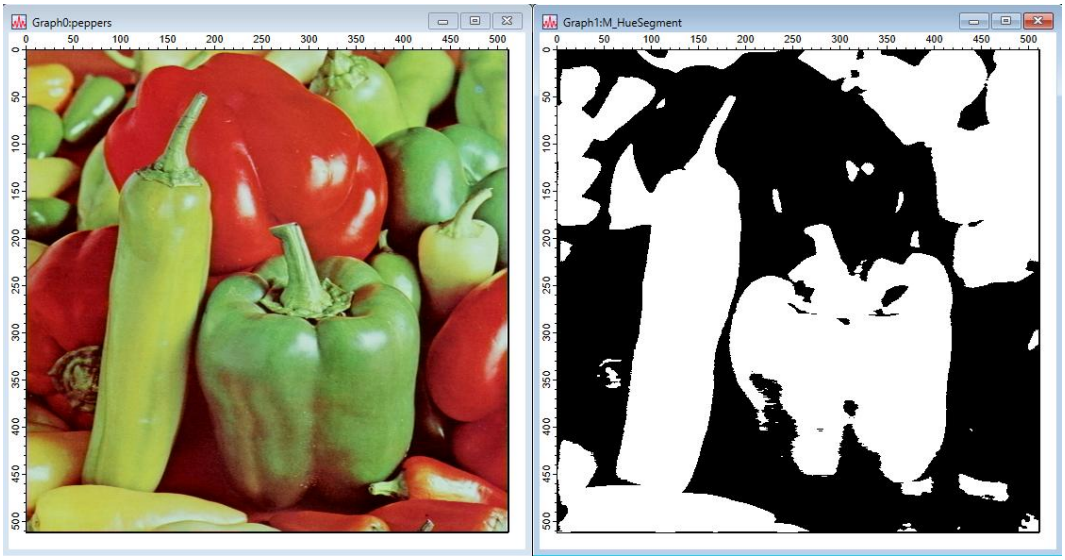


画像登録のデモ

デモエクスペリメント ImageUnwrapPhase Demo を参照してください。

HSL セグメンテーション

カラー画像処理では、グレースケール閾値処理に相当する方法が２つ存在します。
１つ目は、画像の輝度に対する単純な閾値処理です。
これを行うには、画像を RGB から HSL に変換し、輝度成分に対して閾値処理を適用します。
２つ目の閾値処理に相当する方法は HSL セグメンテーションで、画像が特定の範囲内の HSL 値を持つ領域に分割されます。
次の例では、ピーマン（peppers）の画像（Image Processing Tutorial）をセグメンテーションし、赤ピーマンに対応する領域を特定します。



粒子解析

典型的な粒子解析は３つのステップで構成されます。
まず、画像を前処理する必要があります。
これには、ノイズの除去／低減、背景の調整（ImageRemoveBackground コマンドのヘルプを参照）、閾値処理が含まれます。
２番目のステップは、ImageAnalyzeParticles コマンドを呼び出すことです。
３番目、最後のステップは、ImageAnalyzeParticles コマンドによって生成されたすべてのデータを理解すること、つまり「後処理」です。

前処理に関する問題は、このヘルプファイルの別の箇所ですでに説明されています。
ここでは、前処理済みでクリーンな二値画像（粒子を含む）を起点とするものとします。

```
NewImage root:images:blobs                                    // 元の画像を表示

// Step 1: 二値画像を作成
// /I フラグに注意。これにより出力ウェーブが反転され、粒子はゼロでマークされる
ImageThreshold/I/Q/M=1 root:images:blobs                      // /I フラグに注意!!!

// Step 2: ここでは静音モードでコマンドを呼び出し、２ピクセル以上の大きさの粒子を指定
// 粒子運動情報、境界ウェーブ、および粒子マスキングウェーブについても要求
ImageAnalyzeParticles /Q/A=2/E/W/M=2 stats M_ImageThresh

// Step 3: 後処理の選択
// 検出された境界を粒子の上に表示
AppendToGraph/T W_BoundaryY vs W_BoundaryX

// 数値データを閲覧する場合：
Edit W_SpotX,W_SpotY,W_circularity,W_rectangularity,W_ImageObjPerimeter
AppendToTable W_xmin,W_xmax,W_ymin,W_ymax,M_Moments,M_RawMoments
```

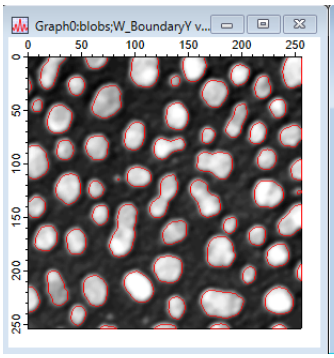


Table0:W_SpotX,W_SpotY,W_circularity,W_rectangularity,W_ImageObjPerimeter,W_xmin...

R0 19

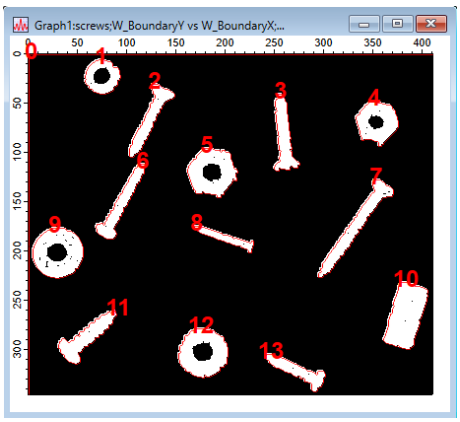
Row	W_SpotX	W_SpotY	W_circularity	W_rectangularity	W_ImageObjPerin
0	19	0	1.05951	0.569333	75.4
1	53	0	0.538796	0.792208	35.2
2	98	0	0.685997	0.867725	75.2
3	146	0	0.673576	0.814745	60.4
4	242	0	1.14716	0.854809	82.4
5	197	6	1.08934	0.738095	61.8
6	137	17	1.02014	0.822222	30.8
7	218	17	1.13606	0.769886	62.2
8	42	18	1.05741	0.784722	54.8
9	0	21	0.634661	0.722222	14.4

注記： 画像境界と交差する粒子については、粒子統計に不正確さが生じる可能性があります。したがって、解析を行う前にこれらの粒子を除去することが有用な場合があります。

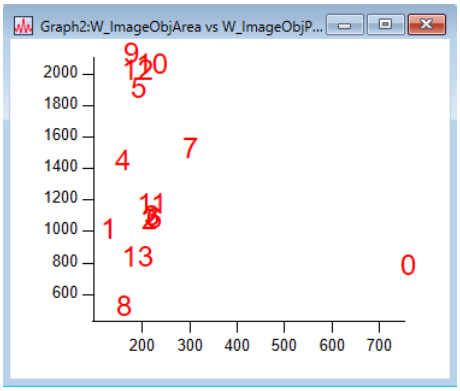
ImageAnalyzeParticles コマンドによって生成された生データは、さらなる処理に使用できます。

次の例は、前処理と後処理がわずかに異なることを示しています。

```
NewImage root:images:screws

// ここでは合成背景を使用しているため、２進数への変換が容易
root:images:screws= root:images:screws==163 ? 255:0
ImageMorphology /O/I=2/E=1 binarydilation root:images:screws
ImageMorphology /O/I=2/E=1 erosion root:images:screws

// 穴埋めオプション付きの粒子解析コマンド
ImageAnalyzeParticles/E/W/Q/M=3/A=5/F stats, root:images:screws

NewImage root:images:screws
AutoPositionWindow/E $WinName(0,1)

// 検出した境界を表示
AppendToGraph/T W_BoundaryY vs W_BoundaryX
AppendToGraph/T W_SpotY vs W_SpotX
Duplicate/O w_spotx w_index
w_index=p
ModifyGraph mode(W_SpotY)=3
ModifyGraph textMarker(W_SpotY)={w_index,"default",1,0,5,0.00,0.00}
ModifyGraph msize(W_SpotY)=6
```



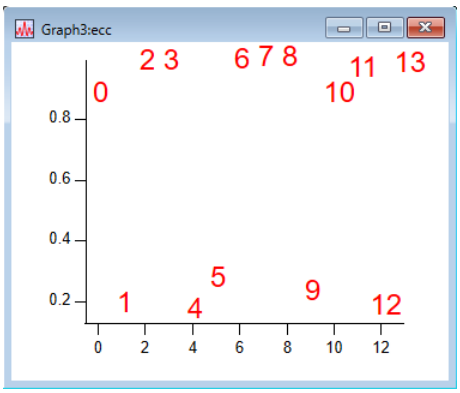

```
// 次に形状分類に対して：粒子面積と周囲長をプロット
Display/W=(23.4,299.6,297,511.4) W_ImageObjArea vs W_ImageObjPerimeter
ModifyGraph mode=3,textMarker(W_ImageObjArea)={w_index,"default",0,0,5,0.00,0.00}
ModifyGraph msize=6
```



この作成した分類図は、画像ノイズの影響を非常に受けやすい２つのパラメーター（面積と周囲長）を使っています。
境界の丸みに関連付けられる基本的なクラスが２つ存在することは確認できますが、分類結果の一部については受け入れがたいものです。

以下では、物体の偏心率に基づく別の分類を計算します。

```
Make/O/N=(DimSize(M_Moments,0)) ecc
ecc=sqrt(1-M_Moments[p][3]^2/M_Moments[p][2]^2)

Display /W=(23.4,299.6,297,511.4) ecc
ModifyGraph mode=3,textMarker(ecc)={w_index,"default",0,0,5,0.00,0.00}
ModifyGraph msize=6
```



２番目の分類法は、ねじとワッシャー／ナットを完璧に区別するものです。
また、最適なパラメーターを選択することの重要性を示しています。

ImageAnalyzeParticles コマンドは、特定の粒子用のマスクを作成する目的でも使用できます。
例えば、上記の例で粒子 9 用のマスクを作成するには：

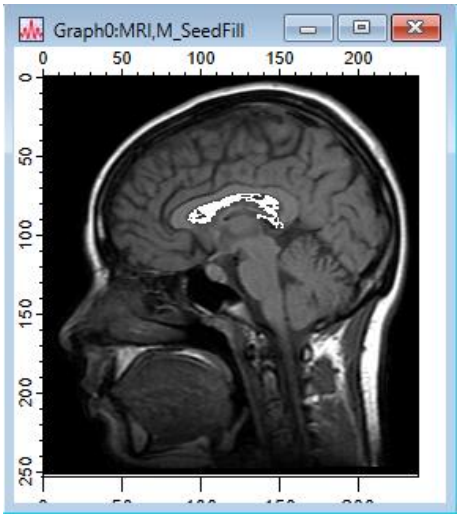
```
ImageAnalyzeParticles /L=(w_spotX[9],w_spotY[9]) mark screws
NewImage M_ParticleMarker
```

このコマンドの機能を使って、オーバーレイで異なるクラスのオブジェクトに色付けできます。

シード塗りつぶし

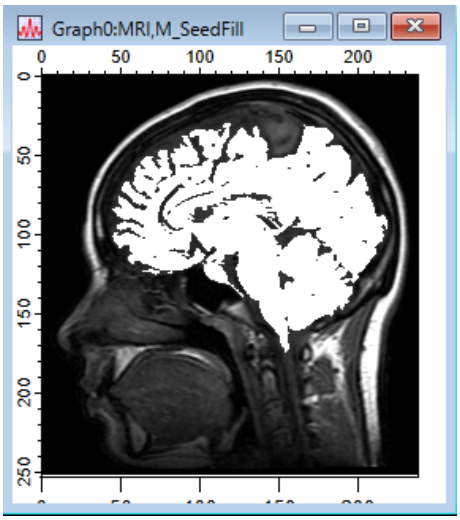
特定の状況では、値が一定の範囲内にある連続したピクセル領域に基づいて画像のセグメントを定義する必要が生じる場合があります。
ImageSeedFill コマンドは、まさにそのためのコマンドです。

```
NewImage root:images:mri
ImageSeedFill/B=64 seedX=132,seedY=77,min=52,max=65,target=255,srcWave=root:images:mri
AppendImage M_SeedFill
ModifyImage M_SeedFill explicit=1, eval={255,65535,65535,65535}
```



ここではオーバーレイ画像を作成するために /B フラグを使いましたが、このフラグはさらなる処理で使う ROI ウェーブを作成するためにも使用できます。
この例は、このコマンドの最も単純な使用法を示しています。
状況によっては、塗りつぶされた領域にピクセルを含める基準が明確でない場合があり、適応アルゴリズムやファジーアルゴリズムを使うと操作がより効果的に機能することがあります。
例えば：

```
ImageSeedFill/B=64/c seedX=144,seedY=83,min=60,max=150,
        target=255,srcWave=root:images:mri,adaptive=3
```



最小値と最大値の制限は緩和されましたが、適応パラメーターが代替の連続性基準を提供する点に留意してください。

その他のツール

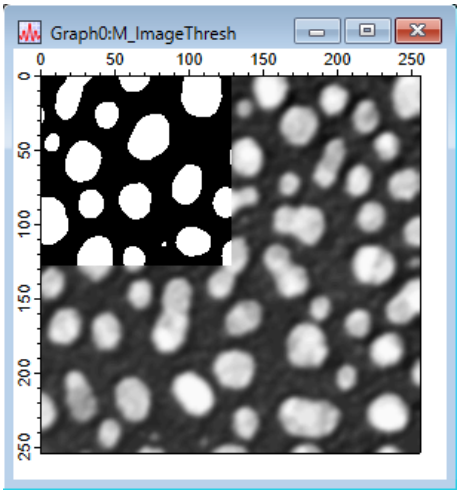
Igor は、画像データの管理と操作を支援する数多くのユーティリティコマンドを提供します。

ROI での作業

多くの画像処理コマンドは、関心領域（ROI）をサポートしています。
関心領域とは、コマンドの対象とする画像の一部です。
Igor は完全に汎用的な ROI をサポートしており、画像と同じディメンションを持つバイナリウェーブ（符号なしバイト）として指定されます。
関心領域内の全てのピクセルを ROI ウェーブでゼロに設定し、関心領域外のピクセルには任意の値を設定します。
なお、状況によっては ROI ピクセルをゼロ以外の値に設定することが有用な場合があります（例：ウェーブを数学演算における乗数として使う場合）。
ImageTransform にキーワード invert を指定することで、この２つのオプションを素早く変換できます。

ROI ウェーブを作成する最も簡単な方法は、コマンドラインから直接行うことです。
例えば、blob 画像の４分の１をカバーする ROI ウェーブは、次のように生成できます。

```
Duplicate root:images:blobs myROI
Redimension/B/U myROI
myROI=64                                    // 任意の非ゼロ値
myROI[0,128][0,127]=0                       // 実際の関心領域

// 使い方の例：
ImageThreshold/Q/M=1/R=myROI root:images:blobs
NewImage M_ImageThresh
```



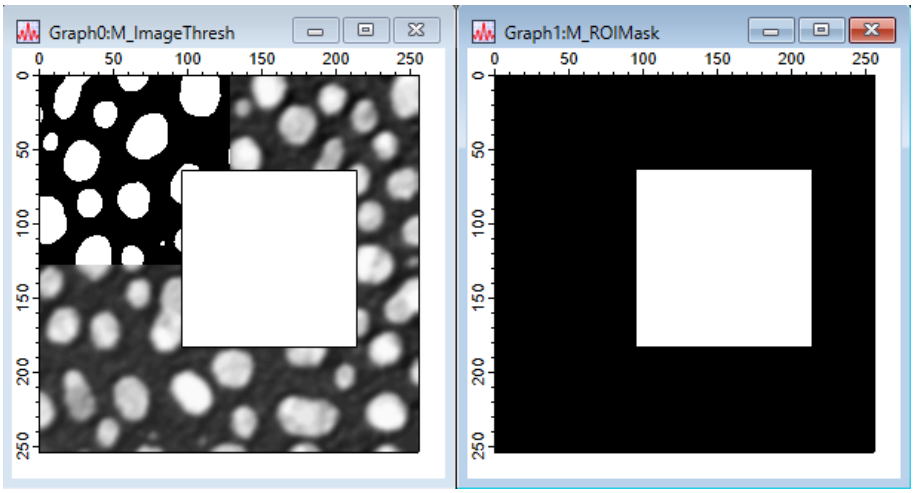
ROI をグラフィカル描画ツールで定義したい場合は、ツールを開き、描画レイヤーを progFront に設定する必要があります。
以下の手順で実行できます。

```
SetDrawLayer progFront
ShowTools/A rect                    // 最初に長方形描画ツールを選択
```

ROI マスクを生成

ROI は、描画したすべての閉じた形状の内側の領域として定義できます。
ROI の描画が完了したら、以下のコマンドを実行する必要があります。

```
HideTools/A                                 // 描画ツールはもう必要ありません
ImageGenerateROIMask M_ImageThresh          // M_ImageThresh はこの例における最上位の画像

// ROI ウェーブが作成された。確認するには、
NewImage M_ROIMask
```

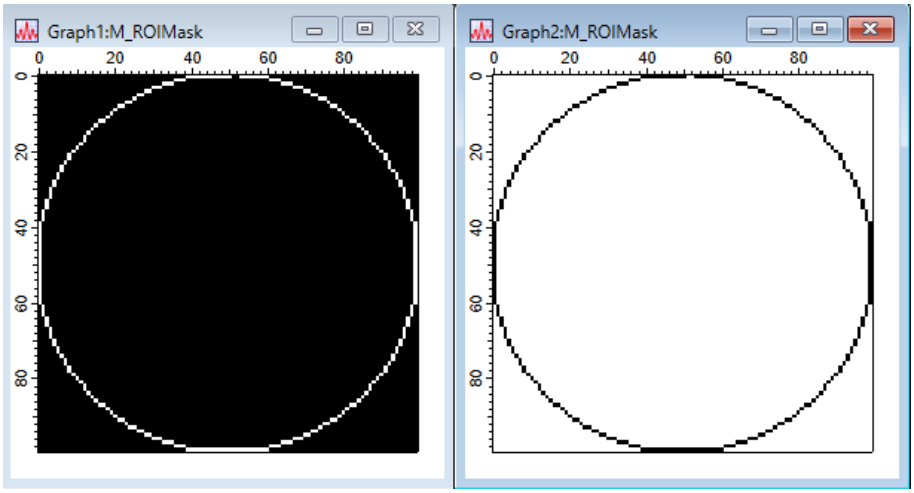


画像処理プロシージャは、表示された画像上に描画することで ROI ウェーブを作成するユーティリティを提供します。

境界をマスクに変換

ROI マスクを生成する 3 つ目の方法は、ImageBoundaryToMask コマンドを使うことです。
このコマンドは、ピクセル値を含むウェーブのペア（y 対 x）を受け取り、それらをスキャン変換してマスクに変換します。
コマンドを呼び出す時には、出力マスクの矩形の幅と高さも指定する必要があります。

```
// 円を作成
Make/N=100 ddx,ddy
ddx=50*(1-sin(2*pi*x/100))
ddy=50*(1-cos(2*pi*x/100))
ImageBoundaryToMask width=100,height=100,xwave=ddx,ywave=ddy
// 結果は曲線ではなく画像
NewImage M_ROIMask
```

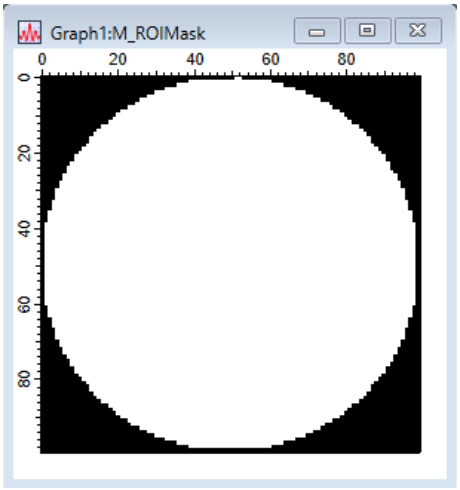


結果として得られるバイナリウェーブは 0 と 255 の値を持つため、一部の操作で使用する前に反転する必要がある場合があることに注意してください。

多くの場合、ImageBoundaryToMask コマンドの後に ImageSeedFill を実行し、マスクを塗りつぶされた領域に変換します。

ImageBoundaryToMask で seedX と seedy キーワードを使えば、目的のマスクを一工程で得られますが、境界ウェーブによって生成されたマスクが閉領域であることを確認する必要があります。

```
ImageBoundaryToMask width=100,height=100,xwave=ddx,ywave=ddy, seedX=50,seedY=50
ModifyImage M_ROIMask explicit=0
```

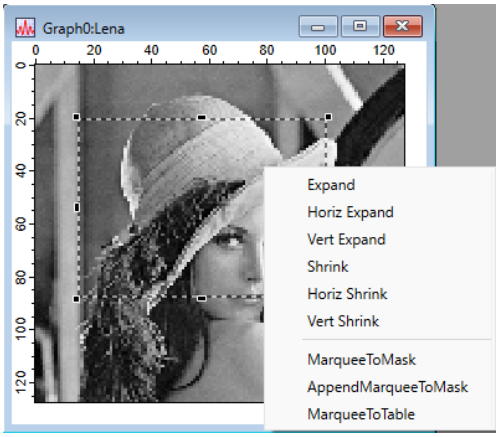


マーキープロシージャ

ROI マスクを作成する４つ目の方法は、Marquee2Mask プロシージャを使うことです。
自身のエクスペリメントでこれを使うには、Procedure ウィンドウに次の行を追加する必要があります。

```
#include <Marquee2Mask>
```

画像内で１つ以上の矩形選択範囲（マウスをクリック＆ドラッグ）を選択することで、ROI マスクを作成できます。
各選択範囲を選択後、選択範囲内をクリックし、MarqueeToMask または AppendMarqueeToMask を選択してください。



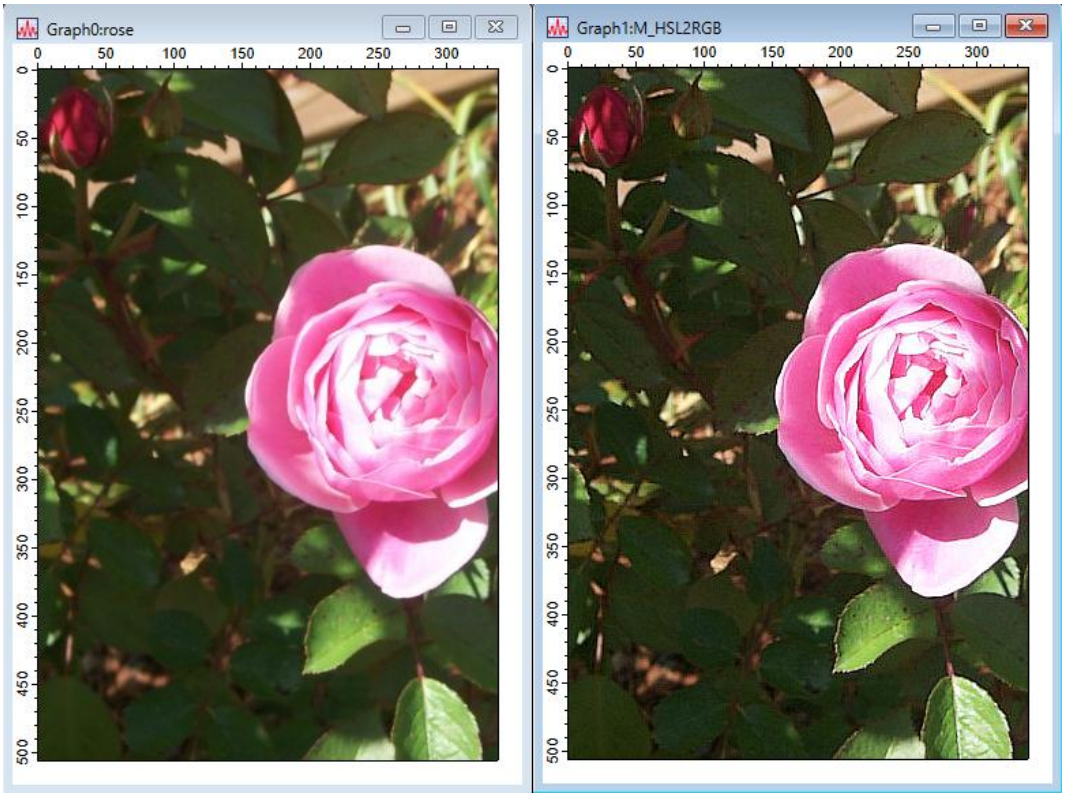
サブイメージの選択

ROI を使うと、画像の選択した部分にさまざまな画像処理コマンドを適用できます。
ROI は、関心領域が連続していない場合や長方形ではない場合に特に有用なツールです。
関心領域が長方形の場合、通常、ROI だけで構成される新しいサブイメージを作成することでパフォーマンスを向上できます。
ROI の座標とサイズがわかっている場合は、Duplicate/R コマンドを使うのが最も簡単です。
インタラクティブな選択を行う場合は、マーキー選択と CopyImageSubset マーキープロシージャを組み合わせて使用できます（画像内でマーキー選択を作成後、マーキー内をクリックし CopyImageSubset を選択）。

色の処理

ほとんどの画像操作はグレースケール画像を対象に設計されています。
カラー画像に対して操作を行う必要がある場合、特定の側面がやや複雑になります。
例えば、カラー画像をシャープにする方法を説明します。

```
NewImage root:images:rose
ImageTransform rgb2hsl root:images:rose                    // 最初に hsl に変換
ImageTransform/P=2 getPlane M_RGB2HSL
ImageFilter Sharpen M_ImagePlane                           // sharpenmore も使用可能
ImageTransform /D=M_ImagePlane /P=2 setPlane M_RGB2HSL
ImageTransform hsl2rgb M_RGB2HSL
NewImage M_HSL2RGB
```

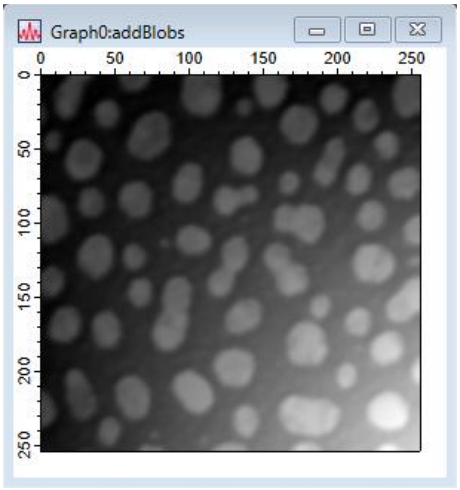


背景の削除

画像から不均一な背景の影響を除去する手法は数多く存在します。
不均一性が加法的である場合、背景に関連する複数の点に多項式を適合させ、得られた多項式曲面を画像全体から差し引く手法が有効な場合があります。
不均一性が乗法的である場合、多項式曲面に対応する画像を生成し、それを用いて元の画像をスケーリングする必要があります。

加法的背景

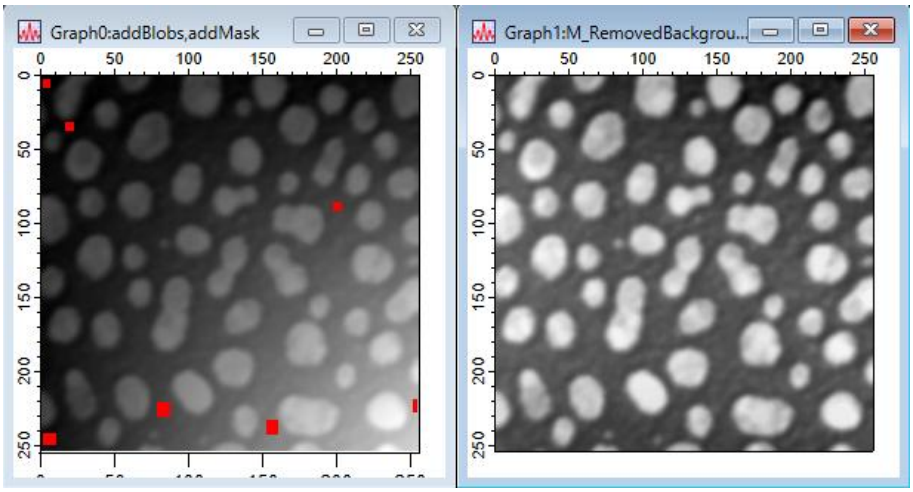
```
Duplicate/O root:images:blobs addBlobs
Redimension/S addBlobs          // 単精度に変換
addBlobs+=0.01*x*y              // 傾いた平面を追加
NewImage addBlobs
```



ImageRemoveBackground コマンドを使うには、画像内の背景領域を示す ROI マスクが必要です。
前述の ROI 構築手法のどれかを使って作成できます。
例えば、以下の劣化したソース画像に示される７つの四角で構成される ROI を選択します。

```
// ROI 背景選択を表示
AppendImage root:images:addMask
ModifyImage addMask explicit=1, eval={1,65000,0,0}

// 修正した画像を作成し、表示
ImageRemoveBackground /R=root:images:addMask /w/P=2 addBlobs
NewImage M_RemovedBackground
```

品質の落ちている元画像　　**背景を削除**



ソース画像に不均一な背景上に比較的小さな粒子が含まれている場合、粒子分析の目的で背景を除去するには、粒子がすべて消えるまでグレースケール収縮を反復処理します。
これにより、元の画像から差し引くことができる、かなり良好な背景の表現が残されます。

乗法的背景（不均一照明）

このケースははるかに複雑です。
背景を除去するには、画像を計算された背景値で除算する必要があるためです（ここでは、画像を生成するシステムの全体的なガンマ値が 1 であると仮定します）。
最初の複雑さは、計算された背景値にゼロが含まれる可能性がある点に関わります。
第二の複雑さは、計算によって結果の画像をスケーリングするための定数係数を自由に選択できる余地が生じることです。
乗法的背景補正には多くの手法が存在します。

例えば、理想的な状況では ROI マスクで特定されたピーク値が同一であるべきと仮定した場合の画像補正方法を示します。

```
Duplicate/O root:images:blobs mulBlobs
Redimension/S mulBlobs                                  // 単精度に変換
mulBlobs*=(1+0.005*x*y)
NewImage mulBlobs

// ROI 前景選択を表示
// 暗い領域内の適切な領域を見つけるためにヒストグラム均等化を使うことができる
AppendImage root:images:multMask
ModifyImage multMask explicit=1, eval={1,65000,0,0}

ImageRemoveBackground /R=root:images:multMask/F/w/P=2 mulBlobs
// フィッティングを正規化
WaveStats/Q/M=1 M_RemovedBackground

// フィッティングを再正規化――その自由因子を一つ利用できる
M_RemovedBackground=(M_RemovedBackground-V_min)/(V_max-V_min)
// 平均値で置き換えることでゼロを除去
WaveStats/Q/M=1 M_RemovedBackground
MatrixOp/O M_RemovedBackground=M_RemovedBackground+V_avg*equal(M_RemovedBackground,0)
MatrixOp/O mulBlobs=mulBlobs/M_RemovedBackground                  // スケーリングされた画像
```

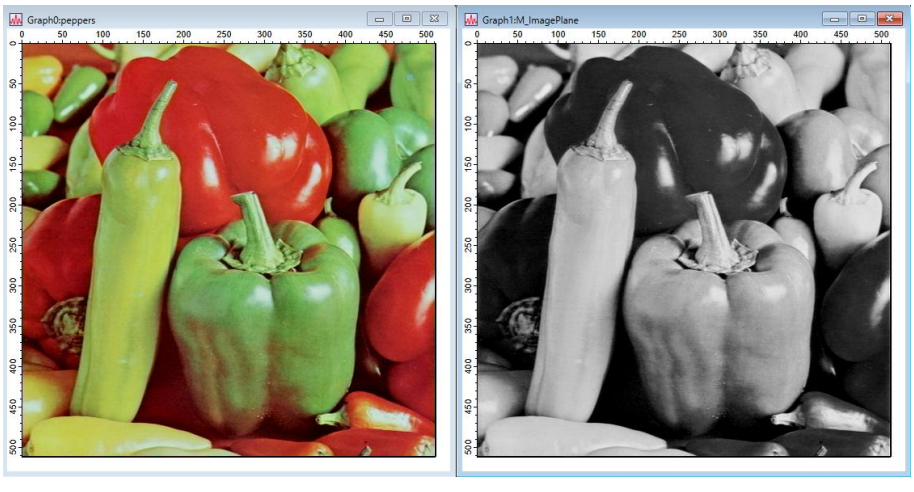
上記の例では、フィッティングに必要な ROI マスクを手動で作成しました。
このプロセスを自動化するには、画像を複数の小さな矩形に分割し、各矩形内で最も高い（または低い）ピクセル値を選択します。
このようなプロシージャの例は、ImageStats コマンドに関連して提供されています。

汎用ユーティリティ: ImageTransform コマンド

前述のように、ImageTransform コマンドは数多くの画像ユーティリティを提供します。
原則として、適切な画像操作が見つからない場合は、ImageTransform で利用可能なオプションを確認してください。
以下に例を挙げます。

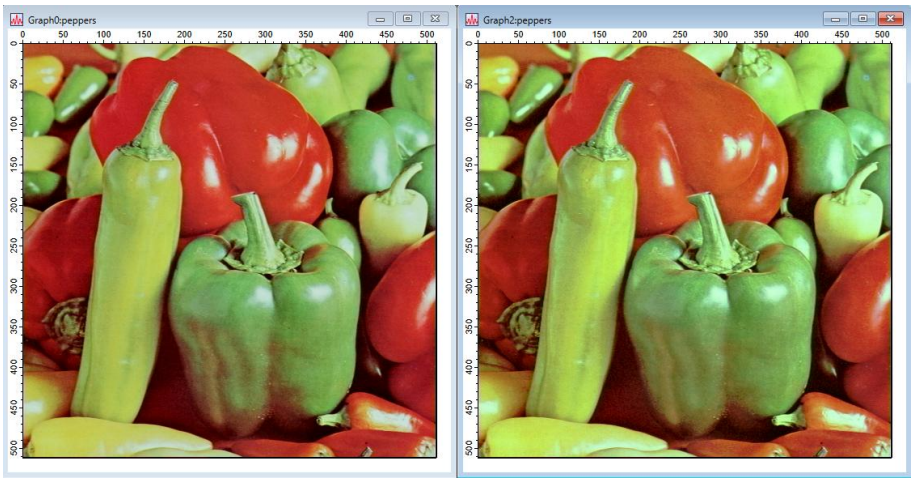
RGB または HSL 画像の処理では、各色成分（平面）を個別に扱う必要が生じることが頻繁にあります。
例えば、peppers 画像の緑成分（緑平面）は次のように取得できます。

```
NewImage root:images:peppers                            // 元の画像を表示
Duplicate/O root:images:peppers peppers
ImageTransform /P=1 getPlane peppers
NewImage M_ImagePlane                                   // グレースケールで緑平面を表示
```



補完コマンドにより、3D ウェーブに平面を挿入できます。
例えば、peppers 画像の緑色の平面を変更したい場合を考えてみます。

```
ImageHistModification/o M_ImagePlane
ImageTransform /p=1 /D=M_ImagePlane setPlane peppers
NewImage peppers                                          // 処理後の画像を表示
```

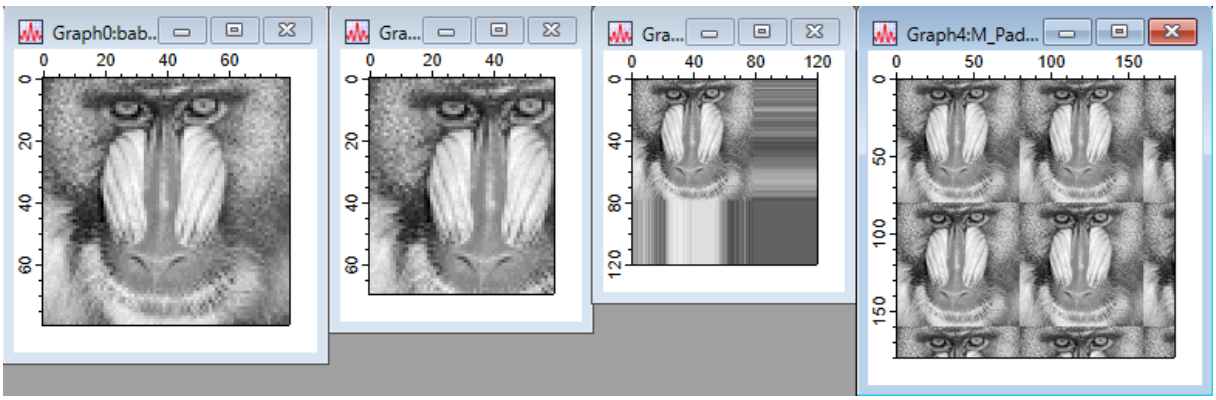


特定のサイズのウェーブに制限される操作があります。
例えば、適応ヒストグラム均等化を使う場合、水平および垂直の分割数は、画像がサブ領域のサイズの正確な倍数であるという要件によって制限されます。
ImageTransform コマンドは 3 つの画像パディングオプションを提供します：

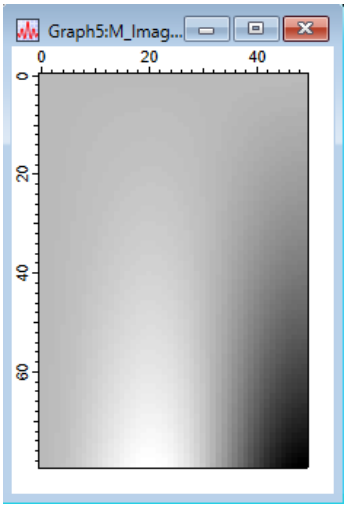
- 変更する行または列に負の数を指定すると、対応する行と列が画像から削除されます。
- 数値が正の場合、行と列が追加されます。
  デフォルトでは、追加された行と列には画像の最後の行と列と全く同じピクセル値が含まれます。
- /W フラグを指定すると、操作により画像の該当部分が新しい行と列に複製されます。

例えば：

```
Duplicate/o root:images:baboon baboon
NewImage baboon
ImageTransform/N={-20,-10} padImage baboon
Rename M_PaddedImage, cropped
NewImage cropped
ImageTransform/N={40,40} padImage baboon
Rename M_PaddedImage, padLastVals
NewImage padLastVals
ImageTransform/W/N={100,100} padImage baboon
NewImage M_PaddedImage
```



もう 1 つのユーティリティ操作は、任意の 2D ウェーブを正規化（0-255）された 8 ビット画像ウェーブに変換することです。
これは ImageTransform コマンドでキーワード convert2gray を使って実現されます。
次に例を示します。




```
// いくつかの数値データセットを作成
Make/O/N=(50,80) numericalWave=x*sin(x/10)*y*exp(y/100)
ImageTransform convert2gray numericalWave
NewImage M_Image2Gray
```

特定のコマンドには 8 ビット画像への変換が必要です。
画像のウェーブサイズを縮小したい場合にも、時に役立ちます。

画像処理参考文献

Ghiglia, Dennis C., and Mark D. Pritt, Two Dimensional Phase Unwrapping -- Theory, Algorithms and Software, John Wiley & Sons, 1998.

Gonzalez, Rafael C., and Richard E. Woods, Digital Image Processing, 3rd ed., Addison-Wesley, 1992.

Pratt, William K., Digital Image Processing, 2nd ed., John Wiley & Sons, 1991.

Thévenaz, P., and M. Unser, A Pyramid Approach to Subpixel Registration Based on Intensity, IEEE Transactions on Image Processing, 7, 27-41, 1998.