CONTENTS

サンプルの Experiment – Fuzzy Image Segmentation Demo

クイックノート

メニュー File → Example Experiments → Imaging → Fuzzy Image Segmentation Demo

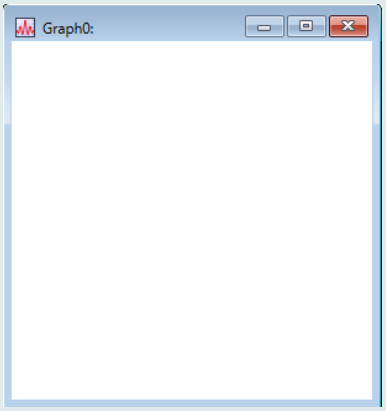
このデモには、ImageTransform (fuzzyClassify) コマンドの適用例が含まれています。

コマンドを実行するには、青いコード行を選択し、Control-Enter キーを押して実行することができます。

手順

グレースケール画像に対するファジー画像分割の例：

ここでは、コマンドを 1 つずつ実行して動作を説明します。



1. コマンドラインで次を実行します。

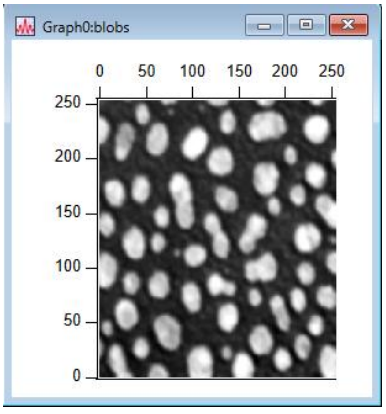
```
Display/K=1/W=(9,38.75,228,256.25)
```

画像を表示するための空のウィンドウです。

2. コマンドラインで次を実行します。

```
AppendImage/T blobs
```

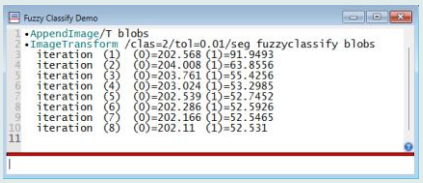
Data Browser 内の「blobs」という画像ウェーブが分析対象です。



3. コマンドラインで次を実行します。

```
ImageTransform /clas=2/tol=0.01/seg fuzzyClassify blobs
```

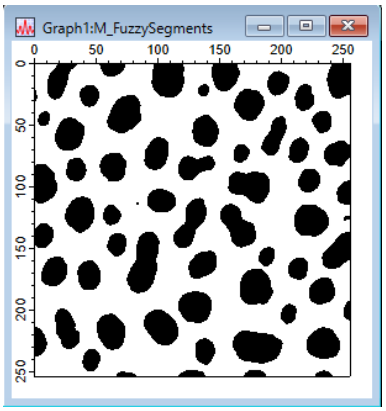
`ImageTransform` コマンドが呼び出され、パラメーター `fuzzyClassify` が渡されます。`/clas` と `/tol` で反復回数が決まります。
M_FuzzySegments というウェーブが生成されます。



4. コマンドラインで次を実行します。

```
NewImage M_FuzzySegments
```

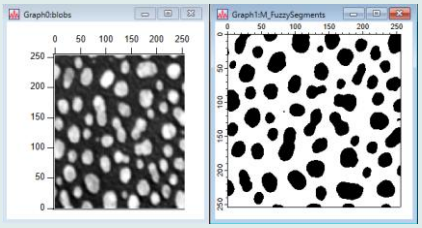
結果の画像が表示されます。



5. コマンドラインで次を実行します。

```
AutopositionWindow/m=0 Graph1
```

元の画像に並べて表示されます。



この例は、ImageThreshold コマンドを使った場合と同じです（グレースケール画像でより複雑な分類を作成しようとしている場合）

次に進む前に、すべてのグラフウィンドウを閉じてください。

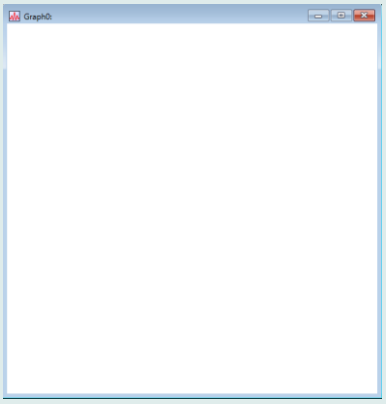
```
KillWindow/Z Graph0
KillWindow/Z Graph1
```

カラー画像は処理に時間がかかります。
次の例では、ファジー画像セグメンテーションを使って 3 つの代表的な色を選択しています（この処理には長い時間がかかる場合があります）。

6. コマンドラインで次を実行します。

```
Graph1Display/K=1 /W=(9,40.25,420,451.25)
```

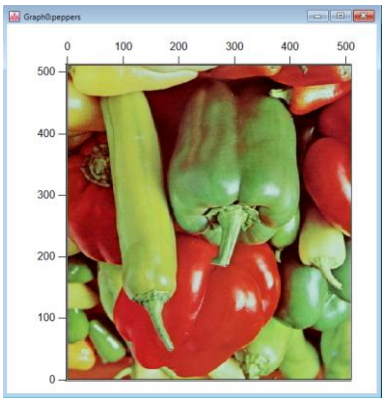
画像を表示するための空のウィンドウです。



7. コマンドラインで次を実行します。

```
AppendImage/T peppers
```

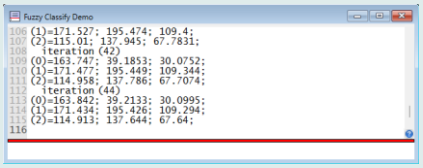
Data Browser 内の「peppers」という画像ウェーブが分析対象です。



8. コマンドラインで次を実行します。

```
ImageTransform /clas=3/tol=0.01/seg fuzzyclassify peppers
```

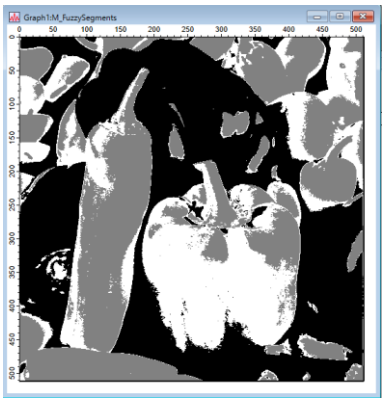
ImageTransform コマンドが呼び出され、パラメーター fuzzyClassify が渡されます。/clas と /tol で反復回数が決まります。
M_FuzzyClasses というウェーブが生成されます。



9. コマンドラインで次を実行します。

```
NewImage/K=1 M_FuzzySegments
```

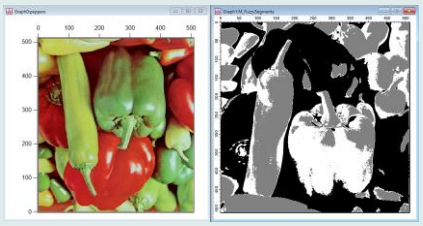
結果の画像が表示されます。



10. コマンドラインで次を実行します。

```
AutoPositionWindow/M=0 Graph1
```

元の画像に並べて表示されます。



履歴に出力された結果は、色セグメンテーションを提供します。
セグメントの順序は異なる場合があり、結果は四捨五入されています。

#	RGB Value			M_FuzzySegments	Source Color
(0)	171	195	109	darkest gray	yellow-green
(1)	114	137	67	midlevel gray	green
(2)	164	39	30	white	red

ImageTransform コマンドのヘルプ

ImageTransform *[flags] Method imageMatrix*

ImageTransform コマンドは、ImageMatrix に対して複数の変換のうちの 1 つを実行します。
ほとんどのキーワードでは、現在のデータフォルダーに新しいウェーブが保存されます。
ユーザーは、関連する出力ウェーブフラグ（/DEST、/DSTB、または /DSTC）を使って、出力ウェーブを指定することができます。
このコマンドで使われるフラグのほとんどは、それらが定義されているキーワードに固有です。

パラメーター

Mathod では変換の種類を選択します。
次のいずれかの名前です：

averageImage　3D の imageMatrix に含まれる画像のスタックの平均画像を計算します。
平均画像はウェーブ M_AveImage に、各ピクセルの標準偏差は現在のデータフォルダー内のウェーブ M_StdvImage に保存されます。
このキーワードは、ROI ウェーブ内のゼロ値ポイントによって関心領域が定義されている場合に、オプションの /R フラグと組み合わせて使用できます。
このコマンドは、ROI（関心領域）外にあるピクセルに対応する M_AveImage と M_StdvImage のエントリを NaN に設定します。
imageMatrix には少なくとも 3 つのレイヤーが含まれていなければなりません。

averageRGBImages
4D ウェーブで表される一連の RGB(A) 画像から平均画像を計算します。
R、G、B、およびオプションの A チャンネルについてそれぞれ平均が計算され、その結果の RGB(A) 画像が、現在のデータフォルダー内のウェーブ M_AverageRGBImage に保存されます。
このコマンドは、すべての実際のデータ型をサポートしています。
Igor Pro 7.0 で追加されました。

backProjection　投影スライスからソース画像を再構築し、その結果をウェーブ M_BackProjection に格納します。
投影スライスは、このコマンドの projectionSlice キーワードによって生成されたウェーブ、または同様にスケーリングされるウェーブである必要があります。
入力は、単精度または倍精度の実数 2D ウェーブでなければなりません。
行のスケーリングは、0 を中心に対称でなければなりません。
例えば、再構築される画像の行数が 256 行になると予想される場合、入力の行のスケーリングは -128 から 127 までにする必要があります。
同様に、入力の列スケーリングは 0 から π の範囲内にある必要があります。
ユーザー関数としての同等の実装は、デモの Experience で提供しています。
フィルタリングされたバックプロジェクションを開発したい場合は、この実装を起点として使うことができます。
参照： projectionSlice キーワードと RadonTransformDemo experiment を参照してください。アルゴリズムの詳細については、Born and Wolf（1999）の "Reconstruction of cross-sections and the projection-slice theorem of computerized tomography" の章を参照してください。

ccsubdivision　Catmull-Clark 再帰的生成により B-spline 曲面を生成します。
この実装には 2 つの有効な入力があります：三角形メッシュまたは四角形メッシュ。

四角形メッシュは、3D ウェーブの形式であると仮定します。
この形式では、最初の平面は X 値、2 番目の平面は Y 値、3 番目の平面は Z 値を含みます。

三角形メッシュは、面-辺-頂点配列に変換する時に非常に複雑になるため、好ましくない選択肢です。
これは 3 列（トリプレット）のウェーブで格納され、1 列目は X 座標、2 列目は Y 座標、3 列目は Z 座標に対応しています。
各三角形は、ウェーブ内の 3 行で記述され、トリプレットウェーブ内の 3 行が連続する三角形が 1 つの三角形に対応するように、共通の頂点は繰り返される必要があります。
また、各頂点にスカラー値を関連付けることもでき、新しい頂点が計算され、古い頂点がシフトされる際に、その値が適切に補間されます。
この場合、トリプレットウェーブの場合は入力ソースウェーブに 1 列、クワッドウェーブの場合は 1 面が多く含まれます。
スカラー値は、任意の点計算の空間的な部分に対して、追加の次元としてあらゆる場所に追加されます。

反復回数は /I フラグを使って指定できます。
デフォルトでは、コマンドは 1 回の反復を実行します。
出力は、4 列で構成されるクワッドウェーブ M_CCBSplines に保存されます。
各行は、X、Y、Z 成分を含む 3 つの平面を含む四角形に対応しています。
入力にオプションのスカラーを使っている場合、スカラーの結果はウェーブ M_CCBScalar に保存されます。

一部の状況では、表面の 2 つ以上の部分が共通の頂点で交わる際に、空間的退化が発生する可能性があります。
これにより、コマンドウィンドウの履歴領域に 1 つまたは複数のエラーメッセージが表示されます。
退化を解消するには、入力座標に小さな擾（じょう）乱を加えることで、自分で試すことができます。
または、/PRTF フラグを使うと、各次元におけるデータの範囲の 1e-10 倍程度のランダムな擾乱が追加されます。
擾乱はスカラーに影響を与えません。

cmap2rgb　画像とその関連カラーマップウェーブ（/C フラグで指定）を、3D ウェーブ M_RGBOut に格納された RGB 画像に変換します。

CMYK2RGB　４レイヤーの符号なしバイトウェーブとして保存されている CMYK 画像を、3 レイヤーの標準 RGB 画像ウェーブに変換します。
出力ウェーブは、現在のデータフォルダーに保存されている M_CMYK2RGB です。

compress　imageWave 内のデータを非損失アルゴリズムで圧縮し、現在のデータフォルダー内の W_Compressed というウェーブに保存します。
圧縮ウェーブには、単位やウェーブノートなど、画像ウェーブに関連するすべてのデータが含まれます。
元のウェーブを復元するには、decompress キーワードを使います。
このコマンドは、すべての数値データ型に対応しています。

注記： 2GB を超えるサイズのウェーブの圧縮形式は、Igor Pro バージョン 6.30B02 で変更されました。
Igor 64 6.30B01 で 2GB を超えるウェーブを圧縮した場合は、同じバージョンを使って解凍する必要があります。
バージョン 6.30B02 以降では解凍できません。

convert2gray　任意の 2D ウェーブを 8 ビットの正規化された 2D ウェーブに変換します。
デフォルトの出力ウェーブ名は M_Image2Gray です。

decompress　圧縮されたウェーブを解凍します。
解凍されたウェーブのコピーは、現在のデータフォルダーに W_DeCompressed という名前で保存されます。

注記： 2GB を超えるサイズのウェーブの圧縮形式は、Igor Pro バージョン 6.30B02 で変更されました。
Igor 64 6.30B01 で 2GB を超えるウェーブを圧縮した場合は、同じバージョンを使って解凍する必要があります。
バージョン 6.30B02 以降では解凍できません。

distance　入力画像の距離変換（別名「距離マップ」）を計算します。
Igor Pro 7.0 で追加されました。

距離変換/マップは、画像内の各ピクセルが「オブジェクト」に属する場合、そのピクセルから背景/境界までの最短距離で置き換えられた画像です。

imageMatrix は、オブジェクトに対応するピクセルが 0 に、背景に対応するピクセルが 255 に設定された、unsigned byte (Make/B/U) 型の 2D ウェーブでなければなりません。
このような画像は、例えば ImageThreshold を使って取得できます。

結果の距離マップは、現在のデータフォルダーのウェーブ M_DistanceMap に保存されます。
このコマンドは、/METR フラグで設定された 3 つの測定基準（マンハッタン、ユークリッド、ユークリッド + スケーリング）をサポートしています。

extractSurface　3D ボリュームウェーブ（ImageMatrix）と交差する平面に相当する値を抽出します。
/X フラグを使って、抽出パラメーターを指定する必要があります。
ボリュームは、3D ウェーブのウェーブスケーリングによって定義されます。
3 線式補間によって得られた結果は、ウェーブ M_ExtractedSurface に保存され、NT_FP64 型になります。
ボリュームの外側にある平面上の点は NaN に設定されます。

fht　/T フラグに従って高速 Hartley（ハートリー）変換を実行します。
ソースウェーブは、行と列の数が 2 の累乗である 2 次元実数行列でなければなりません。
デフォルトの出力は、現在のデータフォルダーに倍精度ウェーブ M_Hartley として保存されます。
/O フラグを使うと、結果が数値型を変更せずに imageMatrix を上書きします。
imageMatrix が単精度浮動小数点型の場合、変換は簡単です。
その他の数値型はすべてスケーリングされます。
シングルバイト型とダブルバイト型は、フルダイナミックレンジにスケーリングされます。
32 ビット整数は、同等の 16 ビット型（つまり、unsigned int は unsigned short 範囲にスケーリングされるなど）の範囲にスケーリングされます。
ウェーブスケーリングや NaN エントリはサポートされていません。

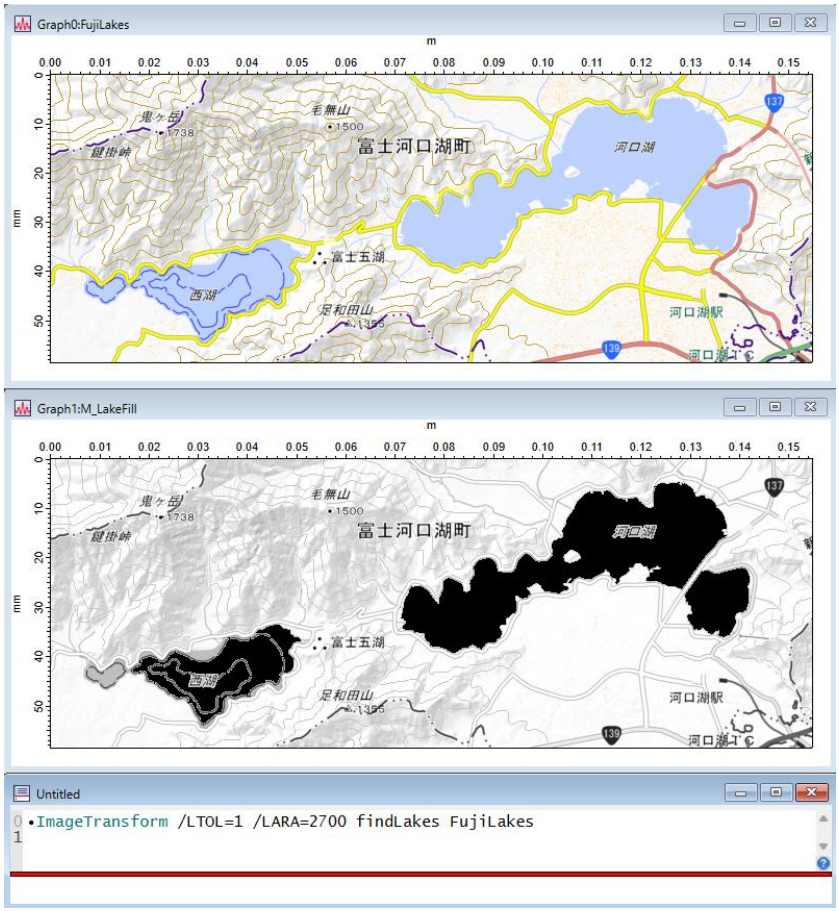
fillImage　2D ターゲット画像のウェーブを、1D 画像ウェーブ（/D=wave1D で指定）のデータで埋めます。
両方のウェーブは、同じデータタイプであり、ターゲットウェーブのデータポイントの数は、データウェーブのポイントの数と一致している必要があります。
/M フラグで指定される 4 つのフィルモードがあります。
このコマンドは、すべての非複素数値データ型をサポートします。

findLakes　もともと地理データ内の湖を識別するために設計されたこのコマンドは、値が互いに近いすべての連続したポイントに対して、2D ウェーブのマスキングを作成します。
連続した領域内の最小ピクセル数を、/LARA=minPixels フラグを使って指定できます

（デフォルトは 100 です）。
/LTOL=tolerance フラグ（デフォルトは 0）を使うことで、連続した領域に属する値がどれほど近接している必要があるかを指定できます。
/LRCT フラグを使うことで、検索範囲を制限することもできます。
/LTAR=target フラグを使って、マスクされた領域の値を設定します。
デフォルトでは、アルゴリズムは隣接するピクセルを評価する時、4-connectivity を使います。
/LCVT フラグを使って、8-connectivity に設定できます。
このコマンドの結果は、ウェーブ M_LakeFill に保存されます。
これは、ソースウェーブと同じデータ型で、マスクされたピクセル以外のすべてのソース値を含みます。

国土地理院の地図画像で実行したときの例：

```
ImageTransform /LTOL=1 /LARA=2700 findLakes FujiLakes
```



flipCols　　ピクセルを、中心列または画像の中心（列の数が偶数の場合）を対称軸として、列を交換して再配置します。
交換はその場で行われ、コマンドを繰り返すことで元に戻すことができます。
3D ウェーブを扱う場合は、/P フラグを使って、処理を行う平面を指定してください。

flipRows　　画像のピクセルを、中央の行または画像の中央（画像の行数が偶数の場合）を対称軸として、行を交換して再配置します。
交換はその場で行われ、コマンドを繰り返し行うことで元に戻すことができます。
3D ウェーブを扱う場合は、/P フラグを使って、処理を行う平面を指定してください。

flipPlanes　　中央の平面を対称軸として平面を交換し、3D ウェーブ内のデータを再配置します。
この処理はその場で行われ、同じコマンドを繰り返し行うことで元に戻すことができます。

fuzzyClassify　ファジーロジックを使ってセグメントをグレースケール画像とカラー画像に分割します。
反復処理は、/TOL で定義された収束条件または /I で指定された最大反復回数に達した時点で停止します。
許容誤差と反復回数を明示的に指定することは、良い使い方です。
クラスの数が少ない場合、コマンドはクラスの値を履歴に表示します。
/Q オプションを使うと、出力を省略し、パフォーマンスを向上させます。
/CLAS オプションを使ってクラス数を指定し、必要に応じて /CLAM オプションを使ってファジー確率値を調整します。
/SEG オプションを使ってセグメンテーション画像を生成します。
クラスは、現在のデータフォルダー内のウェーブ W_FuzzyClasses に保存され、すでに存在する場合は上書きされます。

imageMatrix がグレースケール画像の場合、各クラスは 1 つのウェーブエントリになります。
imageMatrix が RGB 画像の場合、クラスはウェーブ W_FuzzyClasses に連続して格納されます。
imageMatrix に存在するクラス数よりも多くのクラスを要求した場合、データクラスの空間が 1 つのクラスではなく複数のクラスによって張られる退化現象が発生する可能性があります。
すべてのクラスの組み合わせについてユークリッド距離を計算し、その距離が一定のしきい値を下回った場合に縮退を排除することを推奨します。

任意の実数型が使用可能ですが、値の範囲が [0,255] であることが最適です。
3 層以上の 3D ウェーブをセグメント化できます。
この場合、クラスは imageMatrix の層数と同じ次元数のベクトルになります。

例については、Examples/Imaging/fuzzyClassifyDemo.pxp を参照してください。

getBeam　3D ウェーブからビームを抽出します。

「ビーム」は Z 方向の 1 次元配列です。
行が第 1 次元で 1 次元配列であり、列が第 2 次元で 1 次元配列である場合、ビームは第 3 次元で 1 次元配列です。

ビーム内のポイントの数は、imageWave 内のレイヤーの数と等しくなります。
/BEAM={row,column } フラグでビームを指定します。
結果は、現在のデータフォルダーのウェーブ W_Beam に保存されます。
W_Beam は imageWave と同じ数値型です。
beam の値を設定するには setBeam を使います。（関連項目：MatrixOp beam）

getChunk　imageMatrix からチャンクインデックス /CHIX で指定されたチャンクを抽出し、現在のデータフォルダーのウェーブ M_Chunk に保存します。
例えば、imageMatrix のディメンションが (10,20,3,10) の場合、結果の M_Chunk のディメンションは (10,20,3) になります。
setChunk と insertChunk も参照してください。

getCol　2 次元または 3 次元の任意のタイプのウェーブから、W_ExtractedCol という名前の 1 次元ウェーブを作成します。
列は、/G=columnNumber フラグを使って指定します。
ソースウェーブが 3D の場合は、/P=planeNumber フラグを使って平面を指定することもできます。
平面を指定しない場合、Igor は最初の平面を使っていると想定します。
imageMatrix は実数または複素数です。(putCol キーワード、MatrixOp col も参照してください)

getPlane

/P フラグで指定した平面のデータを含む、M_ImagePlane という名前の新しいウェーブを作成します。
新しいウェーブは、ソースウェーブと同じデータ型になります。
/PTYP フラグを使って、平面のタイプを指定することができます。(setPlane キーワード、MatrixOp も参照してください)

getRow

2D または 3D ウェーブの任意のタイプから、W_ExtractedRow という名前の 1D ウェーブを作成します。
行は、/G=rowNumber フラグを使って指定します。
ソースウェーブが 3D の場合は、/P= planeNumber フラグを使って平面を指定することもできます。
平面を指定しない場合、Igor は最初の平面を使っているとみなします。
imageMatrix は実数または複素数です。(putRow キーワード、MatrixOp row も参照してください)

Hough

入力ウェーブの Hough 変換を実行します。
結果は、列が角度、行が放射領域に対応する 2 次元ウェーブ M_Hough に保存されます。

デフォルトでは、出力は 180 列で構成されています。
/F フラグを使って、角度解像度を変更できます。

入力画像が N 行と M 列の場合、M_Hough の行数は $1 + \sqrt{(N^2 + M^2)}$に設定されます。
出力半径は、中心行を基準として読み取ります。

入力ウェーブは、単位サイズの正方形のピクセルで構成され、背景値が 0 のバイナリ（/B/U）であると仮定します。

ヘルプ Hough Transform も参照してください。

hsl2rgb

3 平面 HSL ウェーブを 3 平面 RGB ウェーブに変換します。
このコマンドのソースウェーブがバイトまたは符号なしバイト以外のタイプの場合、HSL 値は 0 から 65535 の範囲である必要があります。
すべてのソースウェーブタイプについて、結果の RGB ウェーブは unsigned short タイプになります。
この演算の結果は、RGB 値が 0 から 65535 の範囲のウェーブ M_HSL2RGB (タイプは unsigned word) になります。

hslSegment

指定された画像から、同じディメンションを持つ二値画像を作成します。
この画像では、指定された 3 つの Hue（色相）、Saturation（彩度）、Lightness（明度）の範囲に属するすべてのピクセルが 255 に設定され、それ以外のピクセルは 0 に設定されます。
HSL の範囲は、/H、/S、/L フラグを使って指定できます。
各フラグは、引数として、値のペアまたは値のペアを含むウェーブを受け取ります。
/H フラグを必ず指定する必要がありますが、/S と /L フラグは省略可能です。
その場合、デフォルト値（フル範囲 [0,1] に対応する値）が使われます。
imageMatrix は RGB 画像であると仮定されます。

imageToTexture

2D または 3D 画像のウェーブを、連続したピクセルコンポーネントの 1D ウェーブに変換します。
この変換は、OpenGL テクスチャ (Gizmo 用) を作成する場合や、RGB または RGBA シーケンスを必要とする形式でカラー画像を保存する場合に役立ちます。

/O フラグは imageToTexture には適用されません。

変換のタイプを指定するには、/TEXT フラグを使います。
imageMatrix は、符号なしバイトウェーブでなければなりません。
W_Texture という名前の 1D 符号なしバイトウェーブが、現在のデータフォルダーに作成されます。
W_Texture のウェーブのノートは、StringByKey および NumberByKey を使って解析できる、セミコロンで区切られたキーワードと値のペアのリストに設定されています。

キーワード	**キーワードに続く情報**
WIDTHPIXELS	DimSize(imageMatrix, 0) または /TEXT 値が奇数の場合、2 の累乗に切り詰められます。
HEIGHTPIXELS	DimSize(imageMatrix, 1) または 2 の累乗に最も近い値に切り詰められます。
LAYERS	DimSize(imageMatrix,2)
TEXTUREMODE	/TEXT フラグの val パラメーター
SOURCEWAVE	GetWavesDataFolder(imageMatrix, 2)

indexWave　現在のデータフォルダーに、インデックスウェーブ（/IWAV を参照）が指す imageMatrix の値を含む 1D ウェーブ W_IndexedValues を作成します。
インデックスウェーブの各行は、imageMatrix の 1 つの値に対応しています。
（imageMatrix の次元内に）有効なインデックスを指さない行がある場合、対応する値は 0 に設定され、コマンドはエラーを返します。
インデックスは 0 を基点とする整数です。
このコマンドは補間をサポートしておらず、ウェーブのスケールを無視します。

insertChunk　/D フラグで指定したチャンク（3D ウェーブ）を、/CHIX フラグで指定したチャンクインデックスで imageMatrix に挿入します。
挿入されるチャンクのディメンションは、imageMatrix の最初の 3 つのディメンションと一致する必要があります。
ウェーブも、同じ数値型である必要があります。
画像行列の 4 次元目が 1 増加され、新しいデータを格納するためのスペースが確保されます。
getChunk と setChunk も参照してください。

insertImage　フラグ /INSI で指定された画像を、フラグ /INSX と /INSY で指定された位置から imageMatrix に挿入します。
imageMatrix が 3D ウェーブの場合、/P フラグで指定されたレイヤーに画像を挿入します。
挿入された画像と imageMatrix は同じデータ型でなければなりません。
挿入されたデータは imageMatrix の境界で切り取られます。

insertXplane　3D ウェーブに、X 軸に垂直な新しい平面として 2D ウェーブを挿入します。
/P フラグは挿入位置を指定し、/INSW フラグは挿入するウェーブを指定します。
2D ウェーブは、3D ウェーブと同じ数値データ型であり、そのディメンションは 3D ウェーブの列数 x レイヤー数でなければなりません。
例えば、3D ウェーブのディメンションが (10x20x30) の場合、2D ウェーブは 20x30 になります。
/O フラグを使わない場合、結果は現在のデータフォルダーの M_InsertedWave に保存されます。

insertYplane　3D ウェーブに、Y 軸に垂直な新しい平面として 2D ウェーブを挿入します。
/P フラグは挿入位置を指定し、/INSW フラグは挿入するウェーブを指定します。
2D ウェーブは、3D ウェーブと同じ数値データ型であり、そのディメンションは 3D ウェーブの行数 x レイヤー数でなければなりません。
例えば、3D ウェーブのディメンションが (10x20x30) の場合、2D ウェーブは 10x30 になります。

/O フラグを使わない場合、結果は現在のデータフォルダーの M_InsertedWave に保存されます。

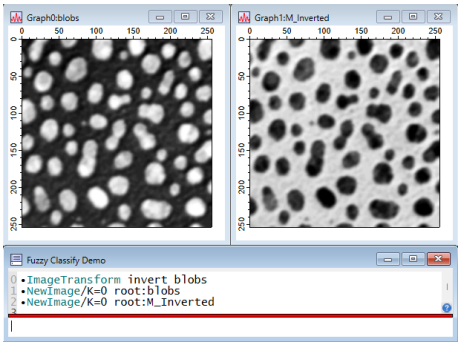
insertZplane

3D ウェーブに、Z 軸に垂直な新しい平面として 2D ウェーブを挿入します。
/P フラグは挿入位置を指定し、/INSW フラグは挿入するウェーブを指定します。
2D ウェーブは、3D ウェーブと同じ数値データ型であり、そのディメンションは 3D ウェーブの行数 x 列数でなければなりません。
例えば、3D ウェーブのディメンションが (10x20x30) の場合、2D ウェーブは 10x20 である必要があります。
/O フラグを使わない場合、結果は現在のデータフォルダーの M_InsertedWave に保存されます。
このキーワードは、完全性を確保するために含まれています。
同じタスクは、InsertPoints を使うことで実行できます。

invert

ピクセル値を、newValue=255-oldValue の式を使って変換します。
あらゆる次元のウェーブで機能しますが、タイプが unsigned byte のウェーブでのみ機能します。
結果は、/O フラグを指定しない限り、ウェーブ M_Inverted に格納されます。
（Fuzzy Classify Demo を使うと）

```
ImageTransform invert blobs
```



matchPlanes

3D ウェーブのすべてのレイヤーでテスト条件と一致するピクセルを検索します。
2 つの値 0 および 255 に設定された 2D 符号なしバイト出力ウェーブ M_matchPlanes を作成します。
値 255 は、条件が指定されたウェーブのすべてのレイヤーで、対応するピクセルがテスト条件を満たしていることを示します。
それ以外の場合は、ピクセル値は 0 になります。

テスト条件は、/D フラグを使って 2D ウェーブとして入力します。
条件ウェーブは倍精度であり、3D ソースウェーブのレイヤー数と同じ数の列を含む必要があります。
ソースウェーブのレイヤー j の条件は、条件ウェーブの列 j の 2 行で指定されます。
最初の行のエントリ（例えば A）と 2 番目の行のエントリ（例えば B）は、層 j のピクセルに対して $A \le x \le B$ という条件を意味します。
条件ウェーブに行のペアを追加することで、特定のレイヤーに複数の条件を設定することができます。
例えば、連続する行に値 C と D を追加した場合、これは次のテストを意味します：

$$(A \le x \le B) \parallel (C \le x \le D)$$

一部のレイヤーに条件が設定されていない場合、そのレイヤーに対応する条件列を NaN に設定してください。
同様に、第 1 レイヤーに 2 つの条件、第 2 レイヤーに 1 つの条件がある場合、条件ウェーブの 1 列目の下部に NaN をパディングします。
このキーワードを使って色相/彩度のセグメンテーションを行う例については、後半の例を参照してください。

offsetImage

画像の座標を XY 平面上で dx、dy ピクセル分移動します（/IOFF フラグで指定されます）。
シフトされた画像の外側のピクセルは、指定された背景色に設定されます。
このコマンドは、2D ウェーブ、またはオプションの /P フラグを指定した 3D ウェー

ブに対して機能します。
指定した平面がない 3D ウェーブをシフトすると、すべての平面が同じ量だけオフセットされた 3D ウェーブが作成されます。
ウェーブ M_OffsetImage には、現在のデータフォルダーの結果が含まれます。

/O フラグは offsetImage ではサポートされていません。

padimage　ソース画像をリサイズします。
拡大時に、最後の行と列の値が新しい領域に埋め込まれます。
/N フラグは、行と列の変更に基づいて新しい画像のサイズを指定します。
/W フラグは、画像のパディング時にデータを折り返すかどうかを指定します。
/O フラグを使わない場合、結果は現在のデータフォルダー内のウェーブ M_PaddedImage に保存されます。

projectionSlice　さまざまな角度で画像を通過する平行な光線の投影スライスを計算します。
ファン内のすべての光線について、このコマンドは画像を通る線積分（光線に沿った線プロファイルの和に相当）を計算します。
この演算は、光線の数と位置、および正の x 方向との角度によって定義される複数のファンの線積分を計算します。
/PSL フラグを使って、投影パラメーターを指定します。
投影スライス自体は、2 次元ウェーブ M_ProjectionSlice に格納されます。このウェーブでは、行は光線に対応し、列は選択した角度の範囲に対応します。
この操作は、ウェーブスケーリングをサポートしていません。
ソースウェーブが 3D の場合、投影スライスは現在、Z 軸に垂直で、平面番号で指定されるスライスをサポートしています。

参照： backProjection キーワードおよび RadonTransformDemo experiment を参照してください。
アルゴリズムの詳細については、Born and Wolf, 1999 の「Reconstruction of cross-sections and the projection-slice theorem of computerized tomography」の章を参照してください。

putCol　imageMatrix の列を、/D フラグで指定されたウェーブの値に設定します。
/G フラグを使って列番号を指定し、/P フラグを使って平面を指定します。
指定したウェーブ間のエントリ数が一致しない場合、コマンドは少ない方の数を使うことに注意してください。
getCol キーワードも参照してください。

putRow　imageMatrix の行を、/D フラグで指定されたウェーブの値に設定します。
/G フラグを使って行番号を指定し、/P フラグを使って平面を指定します。
指定したウェーブ間のエントリ数が一致しない場合、コマンドは少ない方の数を使うことに注意してください。
getRow キーワードも参照してください。

removeXplane　3D ウェーブから、X 軸に垂直な 1 つ以上の平面を削除します。
/P フラグは開始位置を指定します。
デフォルトでは 1 つの平面が削除されますが、/NP フラグを使うと、複数の平面を削除することができます。
/O フラグを使わない場合、結果は現在のデータフォルダー内のウェーブ M_ReducedWave に保存されます。

removeYplane　3D ウェーブから、Y 軸に垂直な 1 つ以上の平面を削除します。
/P フラグは開始位置を指定します。
デフォルトでは 1 つの平面が削除されますが、/NP フラグを使うと、複数の平面を削除することができます。

/O フラグを使わない場合、結果は現在のデータフォルダー内のウェーブ M_ReducedWave に保存されます。

removeZplane　3D ウェーブから、Z 軸に垂直な 1 つ以上の平面を削除します。
/P フラグは開始位置を指定します。
デフォルトでは 1 つの平面が削除されますが、/NP フラグを使うと、複数の平面を削除することができます。
/O フラグを使わない場合、結果は現在のデータフォルダー内のウェーブ M_ReducedWave に保存されます。

rgb2cmap　入力 RGB 画像を表す 256 色のデフォルトカラーマップを計算します。
色は、RGB 空間で入力ピクセルをクラスタリングして計算されます。
結果のカラーマップは、現在のデータフォルダーのウェーブ M_ColorIndex に保存されます。
このコマンドでは、コマンドを使って画像を表示するために使用できる、カラーマップへのインデックスを含むウェーブ M_IndexImage も保存されます。

```
NewImage M_IndexImage
ModifyImage M_IndexImage cindex= M_ColorIndex
```

デフォルトの色数を変更するには、/NCLR フラグを使います。
色の数が 256 を超える場合、M_IndexImage は、その数に応じて 16 ビットの符号なし整数または 32 ビットの整数ウェーブになります。
rgb2cmap は、符号なしバイトまたは単精度浮動小数点型の 3D ウェーブ形式の入力画像をサポートしています。
浮動小数点オプションは、符号付き数値データを使う色空間で画像を入力する場合に使用できます。

rgb2gray　入力 imageMatrix が 3D RGB ウェーブの場合、rgb2gray は、入力のグレースケール表現を含む、タイプ unsigned byte の 2D ウェーブを生成します。
デフォルトでは、このコマンドは、現在のデータフォルダー内のウェーブ M_RGB2Gray に出力を保存します。
RGB 値は、YIQ 標準の輝度 Y に変換されます。
変換式は次のとおりです：

Y = 0.299R + 0.587G + 0.114B

入力 imageMatrix が複数の (3 層) RGB チャンクを含む 4D ウェーブの場合、変換により、各層が入力ウェーブの対応するチャンクのグレースケール変換に対応する 3D ウェーブが生成されます。
この場合、出力の数値型は入力の数値型と同じですが、変換式は同じです。
4D RGB ウェーブを変換する場合、/O フラグはサポートされていません。
デフォルトの出力ウェーブ名は M_RGB2Gray です。
（Fuzzy Classify Demo を使うと）

```
ImageTransform rgb2gray peppers
```

rgb2hsl　3D ウェーブに格納されている RGB 画像を、3 つの平面が HSL カラーモデルの彩度、彩度、明度に相当する別の 3D ウェーブに変換します。
/U フラグが使われていない場合、またはソースウェーブが 8 ビットでない場合、すべてのコンポーネントの値は [0,255] の範囲に正規化されます。
その場合は、範囲は [0,65535] になります。
デフォルトの出力ウェーブ名は M_RGB2HSL です。

rgb2i123　RGB 画像の色空間変換を、以下の量に変換します：

$$I_3 = |G - B|/2,$$

I_1、I_2、I_3 は、現在のデータフォルダー内のウェーブ M_I123（元の RGB ウェーブと同じデータタイプを使用）に保存されます。
I_1 は第 1 レイヤーに格納され、I_2 は第 2 レイヤーに、I_3 は第 3 レイヤーに格納されます。
この色変換は、マシンビジョンにおいて有用な応用が期待されています。

詳細については、次の文献を参照してください：Gevers, T., and A.W.M. Smeulders, Color Based Object Recognition, Pattern Recognition, 32, 453-464, 1999.

rgb2xyz　3D RGB 画像ウェーブを、XYZ 色空間に相当する 3D ウェーブに変換します。
変換は D65 ホワイトポイントに基づいており、以下の変換式を使っています：

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

/O フラグが使われていない場合、XYZ 値は「M_RGB2XYZ」という名前のウェーブに格納されます。
/O フラグが使われている場合、ソース画像が上書きされ、単精度ウェーブ（NT_FP32）に変換されます。

roiTo1D　ROI 内のすべてのピクセルをコピーし、1D ウェーブに順番に保存します。
ROI は /R で指定されます。
ROI ウェーブは、imageMatrix と同じディメンションでなければなりません。
imageMatrix が 3D ウェーブの場合、ROI は imageMatrix と同じ数のレイヤーを持つ必要があります。
ウェーブ W_roi_to_1d には、現在のデータフォルダー内の出力が、imageMatrix と同じ数値型で、選択したピクセルが列優先の順序で格納されます。

rotateCols　行をその場で回転させます。
このコマンドは Rotate コマンドと類似していますが、画像に対して実行され、整数の行数を回転させます。
行数は /G フラグで指定されます。

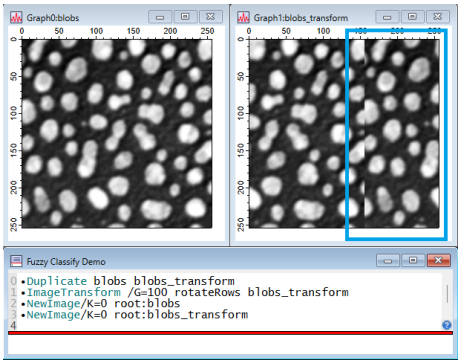
imageMatrix に複数のレイヤーが含まれている場合、/P フラグを使って、回転させるウェーブのレイヤーを指定することができます。
デフォルトでは、/P フラグを指定しない場合、かつ imageMatrix が 3 層（RGB）から構成されている場合、すべての 3 層が回転されます。
それ以外の場合、コマンドはウェーブの最初のレイヤーのみを回転させます。

rotateRows　列をその場で回転させます。
このコマンドは Rotate コマンドと類似していますが、画像に対して実行され、整数の列数を回転させます。
行数は /G フラグで指定されます。

imageMatrix に複数のレイヤーが含まれている場合、/P フラグを使って、回転させるウェーブのレイヤーを指定することができます。

デフォルトでは、/P フラグを指定しない場合、かつ imageMatrix が 3 層（RGB）から構成されている場合、すべての 3 層が回転されます。
それ以外の場合、コマンドはウェーブの最初のレイヤーのみを回転させます。
（Fuzzy Classify Demo を使うと）

```
Duplicate blobs blobs_transform
ImageTransform /G=100 rotateRows
       blobs_transform
```



scalePlanes

3D ウェーブ imageMatrix の各平面を、/D フラグで指定された 1D ウェーブの対応するエントリから取得した定数でスケーリングします。
結果は、/O フラグが指定されていない場合、ウェーブ M_ScaledPlanes に保存されます。
/O フラグが指定されている場合、その場でのスケーリングが行われます。

/O を使う場合は、まずウェーブを別のデータ型にリサイズして、タイプクリッピングによるアーティファクトが発生しないことを確認してください。

imageMatrix が倍精度の場合、M_ScaledPlanes は倍精度になります。
それ以外の場合、M_ScaledPlanes は単精度になります。

このコマンドでは、オプションの /R=ROIWave フラグもサポートされています。

スケーリングバイトデータから生成された 3 つの平面を持つ M_ScaledPlanes を表示する場合、IGOR の SP および DP データの RGB フォーマットでは [0,65535] の範囲が必要であるため、画像を表示するにはウェーブに 255 を乗算する必要があることに注意してください。

selectColor

画像のピクセル値が、画像の色が指定された中央色への近さに依存するマスクを作成します。
中央の色、許容値、およびグレースケール指標は、/E フラグを使って指定する必要があります。

例えば、/E={174,187,75,10,1} は、RGB (174,187,75)、許容レベル 10、および要求されるグレースケール出力を指定します。

RGB 値は、ソース画像に適した範囲で指定する必要があります。
ソースのウェーブタイプがバイトまたは符号なしバイトでない場合、RGB コンポーネントの範囲は 0 から 65535 である必要があります。

色近接度は非一様な RGB 空間で計算され、許容値は中央の色に対応する成分との最大成分差に適用されます。

許容値は、中心色と同様、光源のウェーブの種類に適切なものを使ってください。

生成されたマスクは、現在のデータフォルダー内のウェーブ M_SelectColor に保存されます。
このコマンドの実行前にその名前のウェーブが存在する場合、それは上書きされます。
このコマンドでは、/R フラグを使って、ROI ウェーブの値が 0 のピクセルに色の選択を制限することもできます。

setBeam

画像行列 imageMatrix の特定のビームのデータを設定します。

「ビーム」は Z 方向の 1 次元配列です。
行が第 1 次元の 1 次元配列であり、列が第 2 次元の 1 次元配列である場合、ビームは第 3 次元の 1 次元配列です。

/Beam={row,column} フラグでビームを指定し、/D フラグで 1D ビームデータウェーブを指定します。
ビームデータウェーブは、レイヤー数と同じ要素数、および imageMatrix と同じ数値型を持つ必要があります。getBeam を使ってビームを抽出します。

setChunk　/CHIX で指定したチャンクインデックスのウェーブ imageMatrix のデータを、/D フラグで指定した 3D ウェーブに含まれるデータで上書きします。
割り当てられたデータは、imageMatrix の最初の 3 次元と一致するウェーブに含まれており、同じ数値型である必要があります。
getChunk と insertChunk も参照してください。

setPlane　指定した画像に、/P フラグで指定した平面を設定し、/D フラグで指定したウェーブのデータを設定します。
この機能は、getPlane キーワードの補完として設計されており、複数平面画像を作成するより簡単（高速）な方法を提供します。
このコマンドは、ソースデータが目的の平面よりも小さい場合、ソースデータを目的の平面のコーナーピクセルから連続したメモリ領域に配置する設定をサポートしていることに注意してください。
ソースデータが目的の平面よりも大きい場合、適切な行と列に切り詰められます。
アルゴリズムで命名されたソースウェーブを使って、目的ウェーブのすべての平面を設定する場合は、代わりにキーワード StackImages を使用できます。
getPlane キーワードも参照してください。

shading　フラグ /A で定義された光源の位置に対する表面の相対反射率を計算します。

この計算では、表面の傾斜を推定し、太陽の方向と、対象点における表面法線との内積として定義される相対反射率を計算します。
反射率は、次の式を使ってスケーリングされます。

*outPixel = shadingA * (sunDirection · surfaceNormal) + shading*

デフォルトでは、*shadingA* =1、*shadingB* =0

このコマンドの結果は、ソースウェーブと同じデータ型のウェーブ M_ShadedImage に保存されます。

ソースウェーブが任意の整数型で、shadingA の値が 1 の場合、このコマンドはその値を 255 に設定します。

サポートされる最小サイズのウェーブは 4x4 要素です。

微分評価を含む計算のため、境界（1 ピクセル幅）の値は任意です。
なぜなら、それらのピクセルには微分が存在しないためです。
任意の値は、内部の行と列の複製で埋めます。

shrinkBox　3D 画像マトリックスを縮小し、外側の値と異なる値を持つすべての Voxel を含む最小の 3 次元矩形領域のみを含むようにします。
外側の値は /F フラグで指定されます。
この機能は、ImageSeedFill が対象物周辺の Voxel をある外側の値に設定しており、興味のあるデータを含む最小のボックスを抽出したい場合に役立ちます。
出力は、ウェーブ M_shrunkBox に保存されます。

Igor Pro 7.0 で追加されました。

shrinkRect　画像マトリックスを縮小し、外側の値と異なる値を持つすべてのピクセルを含む最小の矩形のみを含むようにします。
外側の値は /F フラグで指定されます。
この機能は、ImageSeedFill が対象物周辺のピクセルをある外側の値に設定しており、

その対象物を含む最も小さな矩形を抽出したい場合に便利です。
出力は、ウェーブ M_Shrunk に保存されます。

stackImages　現在のデータフォルダー内の個々の画像ウェーブから 3D または 4D スタックを作成します。
ウェーブは、baseNameN の形式である必要があります。
ここで、N はシーケンスの順序を指定する数字の接尾辞です。
imageMatrix は、スタックに追加する最初のウェーブの名称である必要があります。
/NP フラグを使って、スタックに追加するウェーブの数を指定することができます。

その結果、M_Stack という名前の 3D または 4D ウェーブが作成され、現在のデータフォルダーにある同名の既存のウェーブは上書きされます。

/K フラグが指定されている場合、このコマンドは、スタックにコピーしたすべてのウェーブを削除します。

sumAllCols　各エントリが対応する画像の列のピクセルの合計であるウェーブ W_sumCols を作成します。
/P フラグを使って、3D ウェーブの画像平面を指定します。
デフォルトでは、最上位の平面が処理されます。

sumAllRows　各エントリが対応する画像の行のピクセルの合計であるウェーブ W_sumRows を作成します。
/P フラグを使って、3D ウェーブの画像平面を指定します。
デフォルトでは、最上位の平面が処理されます。

sumCol　変数 V_value に、/G フラグ、オプションで /P フラグで指定された列の要素の合計を格納します。

sumPlane　変数 V_value に、/P フラグで指定された平面内の要素の合計を格納します。

sumPlanes　3D ソースウェーブと同じ行数と列数を持つ 2D ウェーブ M_SumPlanes を作成します。
M_SumPlanes の各エントリは、ソースウェーブのすべての平面における対応するピクセルの合計です。
ソースウェーブが倍精度の場合、M_SumPlanes は倍精度ウェーブになります。
それ以外の場合は、単精度ウェーブになります。

sumRow　変数 V_value に、/G フラグで指定された行の要素の合計を格納し、オプションで /P フラグを指定した場合にもその値を格納します。

swap　2 次元 FFT（高速フーリエ変換）後に画像データを交換します。
この変換は、画像の対角線上の四象限を 1 つまたは複数の平面で交換します。
このキーワードはフラグをサポートしていません。
交換はその場で行われ、ソースウェーブは上書きされます。

swap3D　4D ウェーブを、/TM4D フラグで指定した次元でデータが並べ替えられた新しい 4D ウェーブに変換します。
結果は、現在のデータフォルダー内のウェーブ M_4DTranspose に保存されます。
入力ウェーブを上書きするオプションはないため、transpose4D では /O は効果ありません。

Igor Pro 7.0 で追加されました。

transposeVol　3D ウェーブを転置します。
転置されたウェーブは M_VolumeTranspose に保存されます。
/O フラグは適用されません。

このコマンドは、/G フラグを使って指定される次の 5 つの転置モードをサポートしています：

モード	等価のコマンド
1	M_VolumeTranspose=*imageMatrix* [p][r][q]
2	M_VolumeTranspose=*imageMatrix* [r][p][q]
3	M_VolumeTranspose=*imageMatrix* [r][q][p]
4	M_VolumeTranspose=*imageMatrix* [q][r][p]
5	M_VolumeTranspose=*imageMatrix* [q][p][r]

vol2surf

3D「パーティクル」を包む、4 つのウェーブ出力（Gizmo での表示に適しています）を作成します。
パーティクルは、3D ウェーブにおける値が 0 以外のボクセルの領域として定義されます。
このアルゴリズムは、入力ウェーブの解像度で、データを完全に囲むボックスを効果的に計算します。
出力ウェーブ M_Boxy は、12 列の 2D 単精度ウェーブで、各行は 1 つの分離した四角形に対応し、列は四角形の頂点の X、Y、Z 座標を順番に提供します。

voronoi

入力ウェーブの X、Y 位置によって定義される凸領域のボロノイ分割を計算します。
imageMatrix は、最初の列に X 値、2 列目に Y 値、3 列目に任意の定数（0 を推奨）を含むトリプレットウェーブでなければなりません。
このコマンドの結果は、ボロノイポリゴンの連続したエッジを含む 2 列のウェーブ M_VoronoiEdges に格納されます。
エッジは、NaN の列によって互いに分離されています。
最も外側のポリゴンは、凸領域を含む大きな三角形と 1 つ以上の辺を共有しています。

このコマンドにより、2 つの出力ウェーブが作成されます。

M_Circles は、各自然近傍の中心と半径を含む 3 列のウェーブです。
半径は、等間隔のスケールが適用されている場合にのみ正確です。

M_DelaunayTriangles は、三角形の頂点に対応する 3 つの列で構成されています。
各三角形は 4 つの頂点で構成されています。
4 番目の頂点は 1 番目の頂点と等しくなります。
区切りとして使われる NaN の追加行が含まれています。

例えば、本ドキュメント最後の「ボロノイ分割の例」をご覧ください。

xProjection

X 方向への投影を計算し、その結果をウェーブ M_xProjection に格納します。
詳細は、zProjection を参照してください。

xyz2rgb

3D 単精度 XYZ カラースペースのデータを、D65 ホワイトポイントに基づいて RGB に変換します。
使われる変換は次のとおりです：

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

/O フラグを指定しない場合、結果は単精度 3D ウェーブ (NT_FP32) 「M_XYZ2RGB」として保存されます。

すべての XYZ 値が正の RGB トリプレットにマッピングされるわけではないことに注意してください（XYZ 図の RGB 三角形の外側にある色を考えてみてください）。

このコマンドでは、次の選択肢があります。
デフォルトでは、出力ウェーブは、負の RGB 値を含む可能性のある単精度ウェーブになります。
/U フラグを指定した場合、符号なしの短い出力ウェーブでは、負の成分はすべて 0 に設定され、残りの成分は [0,65535] の範囲にスケーリングされます。

yProjection　Y 方向の投影を計算し、その結果をウェーブ M_yProjection に格納します。
詳細は、zProjection を参照してください。

zDot　srcWave のビームと 1D zVector ウェーブ（/D フラグで指定）の内積を計算します。
これは、zVector のスケールに応じて、スペクトルスキャン画像の重ね合わせを RGB または XYZ に変換します。
srcWave と zVector は同じデータ型（float または double）でなければなりません。
ウェーブ M_StackDot には、現在のデータフォルダー内の結果が含まれます。

zProjection　z 方向への投影を計算し、その結果をウェーブ M_zProjection に格納します。
ソースウェーブは、任意のデータ型の 3D ウェーブでなければなりません。
投影の値は、/METH フラグで指定した方法によって異なります。

フラグ

/A={*azimuth*, *elevation* [, *shadingA*, *shadingB*]}
シェーディングのパラメーターを指定します。
光源の位置は、方位角（反時計回りに測定される度数）と仰角（地平線からの角度）で指定されます。

パラメーター shadingA および shadingB はオプションです。
デフォルトでは、それぞれの値は 1 および 0 です。

/Beam={*row*,*column*}
3D ウェーブ内のビームを指定します。
行と列はどちらも 0 を基準としています。

/BPJ={*width*, *height*}
バックプロジェクションのパラメーターである幅と高さを指定します。
これらは再構築された画像の幅と高さであり、元のウェーブのサイズと同じである必要があります。

/C=*CMapWave*　cmap2rgb コマンドの cmap ウェーブを指定します。
CMapWave は、RGB エントリに対応する 3 列で構成される 2D ウェーブである必要があります。

/CHIX=*chunkIndex*
4D ウェーブ内のデータのチャンクを取得、挿入、または設定するためのチャンクインデックスを識別します。
chunkIndex の範囲は 0 から imageMatrix 内のチャンク数までです。

/CLAM=*fuzzy*　fuzzyClassify でファジー確率を計算するために使う値を設定します。
この値は、fuzzy > 1（デフォルトは 2）を満たす必要があります。

/CLAS=*num*　fuzzyClassify で要求するクラスの数を設定します。
予想されるクラスの数が不明で、num が非常に大きい場合、fuzzyClassify は一部の退化したクラスを生成する可能性があります。

/D=*waveName*　データウェーブを指定します。
このウェーブの詳細は、該当するキーワードのドキュメントを確認してください。

/DEST=*destWave*

キーワードコマンドの最初の出力ウェーブで destWave を作成または上書きします。
それ以外の場合は、キーワードのデフォルトの宛先ウェーブが使われます。
このフラグは、キーワードによって生成される最初の出力ウェーブを具体的にコントロールします。
キーワードによって複数の出力ウェーブが生成される場合、/DSTB および /DSTC は、それぞれ 2 番目と 3 番目の出力ウェーブの宛先ウェーブをコントロールします。
宛先ウェーブはソースウェーブにすることはできないため、/O と組み合わせて使うことはできません。

/DSTB=*destWave*

キーワードコマンドの 2 番目の出力ウェーブで destWave を作成または上書きします。
それ以外の場合、キーワードのデフォルトの宛先ウェーブが使われます。
これは、2 つ以上の出力ウェーブを生成するキーワードにのみ適用できます。
/DSTB を使う場合、/DEST は使う必要はありません。

/DSTC=*destWave*

キーワードコマンドの 3 番目の出力ウェーブで destWave を作成または上書きします。
それ以外の場合、キーワードのデフォルトの宛先ウェーブが使われます。
これは、3 つ以上の出力ウェーブを生成するキーワードにのみ適用できます。
/DSTC を使う場合、/DSTB および /DEST は使う必要はありません。

/F=*value*

Hough キーワードと組み合わせて使った場合、角度領域のサンプリングを増加させます。
デフォルト値は 1 で、コマンドの結果は 0 から 180 の間の単一度の増分で出力されます。
値が 1.5 の場合、変換結果には 180×1.5 行が含まれます。

shrinkRect キーワードと組み合わせて使う場合、関心領域を囲む外側のピクセル値を指定します。

/FEQS

ボロノイ分割を実行する時、アルゴリズムに両軸に対して同じスケールを使うように強制します。
通常、各軸のスケールは、その範囲から決定されます。
2 つの軸の範囲が 1 桁以上異なる場合、両方の軸のスケールに大きな範囲を強制的に適用うと便利です。

/FREE

宛先ウェーブフラグ (/DEST、/DSTB、および /DSTC) のいずれかで指定されている場合、自由な宛先ウェーブを作成します。
複数の宛先ウェーブを出力するキーワードの場合、/DEST、/DSTB、または /DSTC で指定したウェーブのみが自由になります。
指定のないウェーブは、デフォルトのウェーブとして出力されます。

/G=*colNumber*

getRow または getCol キーワードと組み合わせて使われる行番号または列番号を指定します。
このフラグは、transposeVol キーワードの転置モードも指定します。

/H={*minHue*, *maxHue*}
/H=*hueWave*

ピクセルを選択するための色相値の範囲を指定します。
色相値は、0 から 360 度の範囲で、度数で指定されます。
ゼロポイントを含む色相間隔は、高い値を先に指定する必要があります。
例：/H={330,10}

hueWave を使うときは、選択したいピクセルを挟む複数の hue の値のペアがある場合に使います。

/I=*iterations*　ccsubdivision および fuzzyClassify における反復回数を設定します。

/INSI=*imageWave*
insertImage キーワードで使うウェーブ imageWave を指定します。
imageWave は、imageMatrix と同じ数値データ型の 2 次元ウェーブです。

/INSW=*wave*　insertXplane、insertYplane、または insertZplane のいずれかのキーワードを使って、3D ウェーブに挿入する 2D ウェーブを指定します。

/INSX=*xPos*　最初の行を挿入するピクセル位置を指定します。
ウェーブスケーリングは無視されます。

/INSY=*yPos*　最初の列を挿入するピクセル位置を指定します。
ウェーブスケーリングは無視されます。

/IOFF={*dx*,*dy*,*bgValue*}
dx および dy で正または負の整数オフセットの量、および offsetImage キーワードで新しい背景値 bgValue を指定します。

/IWAV=*wave*　キーワード indexWave と一緒に使う時、インデックスを提供するウェーブを指定します。
ウェーブは、imageMatrix の次元（2、3、または 4）と同じ数の列を持つ必要があります。
例えば、画像内のピクセルのインデックスを指定するには、ウェーブは 2 列でなければなりません。
最初の列はピクセルの行番号に対応し、2 番目の列はピクセルの列番号に対応します。
ウェーブは任意の数字型（複素数以外）で、エントリは整数インデックスであると仮定されます。
補間やウェーブのスケーリングはサポートされていません。

/L={*minLight*, *maxLight* }
/L=*lightnessWave*
ピクセルを選択するための明度範囲を指定します。
明度値は 0 から 1 の範囲です。
/L フラグを使わない場合、デフォルトの全範囲が使われます。

lightnessWave を使う時は、選択したいピクセルに対応する lightness 値が 2 組以上ある場合に使います。
各組の値は、小さい値が先頭、大きい値が後ろに配置されるように並べます。
ウェーブ内のペアの順序には、そのコマンドで使われる他のウェーブと一致している以外、制限はありません。

/LARA=*minPixels*
findLakes キーワードでマスクされる領域に必要な最小ピクセル数を指定します。
このフラグを指定しない場合、デフォルト値として 100 が使われます。

/LCVT　4-Connectivity の代わりに 8-Connectivity を使います。

/LRCT={*minX*,*minY*,*maxX*,*maxY*}
findLakes キーワードの矩形領域を設定します。
このコマンドは、指定した矩形の外側の元のデータには影響しません。
X および Y 値は、スケーリングされた値 (つまり、ウェーブスケーリングを使用) です。

/LTAR=*target*　findLakes キーワードで指定されたマスク領域のターゲット値を設定します。

/LTOL=*tol*　findLakes キーワードの許容誤差を指定します。
デフォルトでは許容誤差は 0 です。
許容誤差は非負の数でなければなりません。
このコマンドでは、現在のピクセルの値 V と V+tol の間の値を持つ隣接するピクセルを許容することで、許容範囲を適用します。

/M=*n*　fillImage を使って、2D ターゲット画像に 1D ウェーブのデータを塗りつぶす方法を指定します。

n=0:　直列の列埋め込みです。この処理は、リサイズコマンドでも実行可能です。
n=1:　直列の行埋め込みです。
n=2:　サーペンタインの列埋め込みです。データウェーブのポイントは、最初の列に順番にロードされ、最後のポイントから 2 番目の列の最初のポイントへと続き、3 番目の列へと順番にロードされます。

n=3:　サーペンタインの行埋め込みです。

/METH=*method*

xProjection、yProjection、および zProjection キーワードで指定されたピクセルの値を決定します。
詳細は zProjection を参照してください。

method=1:　M_zProjection 内のピクセル (i, j) には、imageMatrix のすべてのレイヤーにおいて (i, j, *) が取る値のうち最大の値が割り当てられます（デフォルト）。
method=2:　M_zProjection 内のピクセル (i, j) には、imageMatrix のすべてのレイヤーにおいて (i, j, *) が取る値の平均値が割り当てられます。
method =3:　M_zProjection 内のピクセル (i, j) には、imageMatrix のすべてのレイヤーにおいて (i, j, *) が取る値のうち最小の値が割り当てられます。

/METR=*method*

距離変換で使うメトリックを設定します。
Igor Pro 7.0 で追加されました。

method=0:　マンハッタン距離。デフォルト。
method=1:　ユークリッド距離（ピクセルの中心間の距離を測定し、ピクセルは正方形であると仮定）。
method=2:　入力のウェーブスケーリングを使って、実際のピクセルサイズも考慮したユークリッド距離。

method =0 は他の方法よりも実行速度が速いです。
method =2 は、特に 2 つの軸に沿って異なるスケーリングが適用された場合、2 倍遅くなる可能性があります。

/N={*rowsToAdd*, *colsToAdd*}

行数分（rowsToAdd）と列数分（colsToAdd）分、画像のサイズを拡大または縮小します。
追加されるピクセルは、ソース画像の最後の行と最後の列の値を複製して設定されます。

このコマンドの結果は、ウェーブ M_paddedImage に保存されます。

/NCLR=*M*　rgb2cmap キーワードを使って検出する色の最大数を指定します。
M は正の整数でなければなりません。
デフォルト値は 256 色です。

/NP=*numPlanes*

removeXplane、removeYplane、または removeZplane キーワードを使う場合、3D ウェーブから削除する平面の数を指定します。
stackImages キーワードでスタックに追加するウェーブの数を指定します。

/O　Hough 変換および cmap2rgb の場合を除き、入力ウェーブを結果で上書きします。
transposeVol パラメーターには適用されません。

/P=*planeNum*　rgb2gray または getPlane メソッドで操作する平面を指定します。
ソースウェーブが 3D の場合、getRow または getCol にも使われます。

/PSL={*xStart*,*dx*,*Nx*,*aStart*,*da*,*Na*}

投影スライスのパラメーターを指定します。
xStart は、画像の中心から測定される平行光線の最初のオフセットです。
dx はファン内の次の光線への方向オフセット、Nx はファン内の光線の数です。
aStart は投影が計算される最初の角度です。
角度は、正の X 方向と光線の方向との間で測定されます。
da は、光線のファンが回転する次の角度へのオフセットであり、Na は投影が計算される角度の総数です。

/PTRF　ccsubdivision キーワードと組み合わせて /PTRF を使うと、入力座標に小さな摂動を適用して、空間のある点で複数のファセットが交わる場合に発生する空間の縮退を解消することができます。
詳細は、上記の ccsubdivision を参照してください。
/PTRF は Igor Pro 9.0 で追加されました。

/PTYP=*num*　getPlane キーワードで使う平面を指定します。

num=0:　XY 平面
num=1:　XZ 平面
num=2:　YZ 平面

/Q　静音フラグ。
Hough 変換と組み合わせて使うと、最大値に対応する角度の履歴への出力を抑制します。

/R=*roiWave*　roiWave で定義された関心領域 (ROI) を指定します。
以下のキーワードと組み合わせて使います: averageImage、scalePlanes、selectColor。

/S={*minSat*, *maxSat*}
/S=*saturationWave*

ピクセルを選択するための彩度値の範囲を指定します。
彩度値は[0,1]の範囲内にあります。
/S フラグを使わない場合、デフォルト値はフル彩度範囲になります。

2 つ以上の彩度の値がある場合、saturationWave を使います。
彩度のウェーブを使う場合は、明度のウェーブも使う必要があります (/L を参照)。
彩度のウェーブは、最初のポイントが低い彩度の値、2 番目のポイントが高い彩度の値という値のペアで構成する必要があります。
ウェーブ内のペアの順序に制限はありません。

/SEG　fuzzyClassify のセグメンテーション画像を計算します。
画像は 2 次元ウェーブ M_FuzzySegments に格納されます。
各ピクセルの値は 255*classIndex/クラス数 です。
ここで、classIndex は、ピクセルが最も高い確率で属するクラスのインデックスです。

/T=*flag*　次のフラグの 1 つまたは複数を使ってください。

flag=1: データをスワップして、DC を画像の中心に配置します。
flag=2: 以下の定義に従って電力を計算します：
P(f)= 0.5* [H(f)^2 + H(-f)^2]

/TEXT=*val* imageToTexture キーワードを使って作成するテクスチャの種類を指定します。
val は、次の値の組み合わせとなる二進数フラグです：

val=1: 各次元を 2 の累乗に最も近い値に切り詰めます。
これは OpenGL テクスチャで必要です。
val=2: 1 次元テクスチャを作成します（他のすべてのテクスチャは 2 次元アプリケーション用です）。
val=4: アルファチャンネルまたは輝度チャンネルに適した単一チャンネルのテクスチャを作成します。
val=8: 3 レイヤー（またはそれ以上）のデータから RGB テクスチャを作成します。
val=16: RGBA テクスチャを作成します。
imageMatrix に 4 番目のレイヤーがない場合、アルファは 255 に設定されます。

/TM4D=*mode* transpose4D と一緒に使って、出力ウェーブのフォーマットを指定します。
ここで、mode は、変換のマッピングを表す 4 桁の整数です。
数字の 1 は第 1 次元、2 は第 2 次元、4 は第 3 次元、8 は第 4 次元を表し、元のウェーブは mode=1248 に対応します。
その他のモードは、4 つの数字のうち 1 つまたは複数を並べ替えることで得られ、モードは 4 つの異なる数字から構成される必要があります。

Igor Pro 7.0 で追加されました。

/TOL=*tolerance* fuzzyClassify での反復収束の許容値を設定します。
収束条件は、すべてのクラスの二乗誤差の合計が許容値未満になった場合に満たされます。
許容値は負の値であってはなりません。

/U rgb2hsl キーワードと併用すると、0 から 65535 までの値を含む、符号なし短整数型の HSL ウェーブを作成します。

/W データをラップして画像をパディングします。
ソースウェーブで利用可能な行または列の数よりも多くの行または列を追加する場合、コマンドはソースデータを必要な回数だけ繰り返します。

/X={*Nx*,*Ny*,*x1*,*y1*,*z1*,*x2*,*y2*,*z2*,*x3*,*y3*,*z3*}
Nx と Ny は、ウェーブ M_ExtractedSurface の行と列です。
残りのパラメーターは、抽出された平面上の 3 つの 3D ポイントを指定します。
3 つのポイントは、平面の頂点から選択し、頂点をスキップせずに時計回りに順番に入力する必要があります。

/Z エラーを無視します。

例

2D (M x N) ウェーブ、plane0 を (M x N x 3) ウェーブの平面番号 0 に挿入したい場合、rgbWave に対して：

```
ImageTransform /P=0/D=plane0 setPlane rgbWave
```

ソースウェーブが 100 行 100 列で、この画像のモンタージュを作成する場合は、次のように指定します：

```
ImageTransform /W/N={200,200} padImage srcWaveName
```

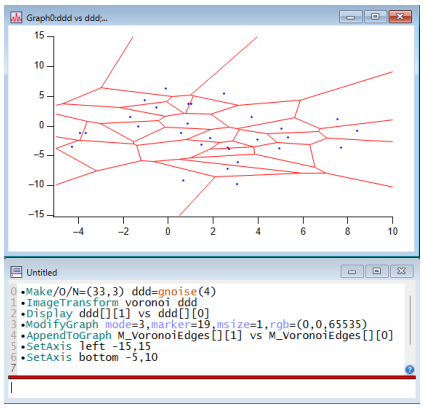
色相と彩度のセグメンテーションの例

```
Function HueSatSegment(hslW,lowH,highH,lowS,highS)
        Wave hslW

        Variable lowH,highH,lowS,highS
        Make/D/O/N=(2,3) condition
        conditionW={{lowH,highH},{lowS,highS},{NaN,NaN}}
        ImageTransform/D=conditionW matchPlanes hslW
        KillWaves/Z condition
End
```

ボロノイ分割の例

```
Make/O/N=(33,3) ddd=gnoise(4)
ImageTransform voronoi ddd
Display ddd[][1] vs ddd[][0]
ModifyGraph mode=3,marker=19,msize=1,rgb=(0,0,65535)
AppendToGraph M_VoronoiEdges[][1] vs M_VoronoiEdges[][0]
SetAxis left -15,15
SetAxis bottom -5,10
```



参考文献

Born, Max, and Emil Wolf, Principles of Optics, 7th ed., Cambridge University Press, 1999.

Details about the rgb2i123 transform:

Gevers, T., and A.W.M. Smeulders, Color Based Object Recognition, Pattern Recognition, 32, 453-464, 1999.

参照

ヘルプ Image Processing
Color Transforms, Handling Color, General Utilities: ImageTransform Operation, MatrixOp