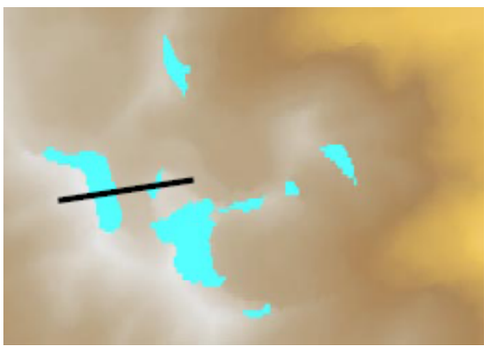
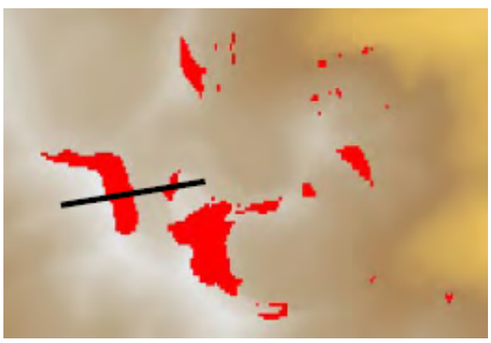
サンプルスクリプト

SRTM DEM の穴を埋める

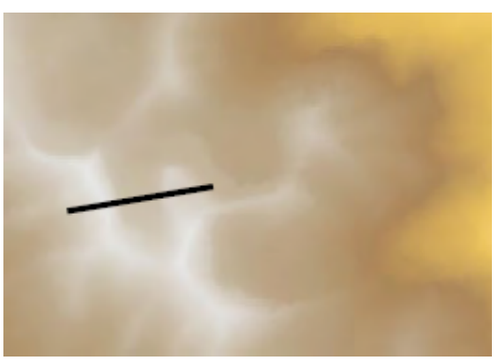


山の多い地域の 1 秒角 SRTM DEM の一部。データの穴 (ヌル値の領域) は青で表されています。断面の線が黒で表されています。



赤い部分はデータの穴と、参照用 DEM からの高度差が 75 メートル以上ある SRTM DEM セルを示しています。75 メートルは、DEM 値と置き換えるためにこの例で使われた高度差のしきい値です。

茶色の線は、埋めた空隙の縁をならすために使われた 3 セル幅の緩衝帯の輪郭です。

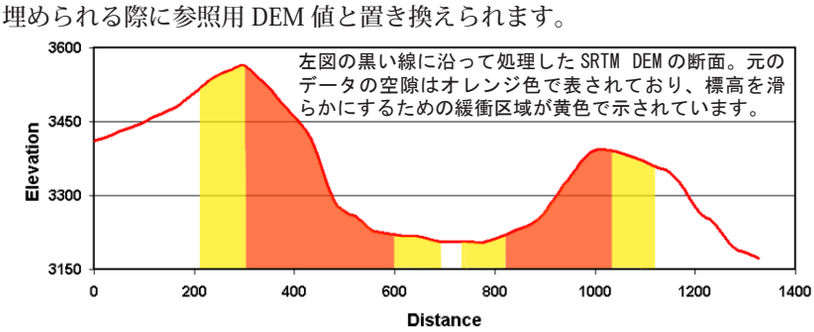


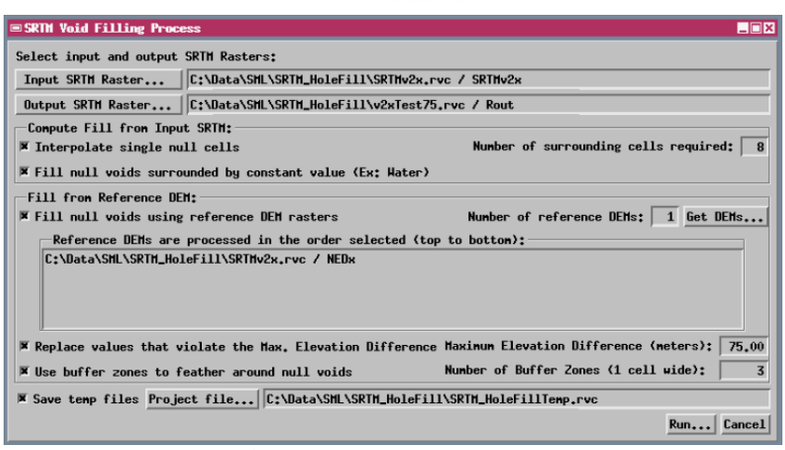
処理後の SRTM DEM。穴を埋めて、滑らかに繋げました。

NASA はスペースシャトルレーダ立体地形ミッション (SRTM) により取得されたレーダデータを使って、地球の陸域の 80 パーセントを覆う数値標高モデル (DEM) を作成しました。北米のセルサイズは 1 秒角であり、その他の地域は 3 秒角です。SRTM の DEM には、レイオーバやシャドウイング、滑らかな水面からの信号リターンの欠如など、レーダによるデータ取得システム固有の問題が原因で出来るデータ空隙 (穴) が含まれています。

マイクロイメージ社は SRTM の DEM のデータが無いセルを埋めるためのサンプルスクリプトを作成しました。SRTMFILL スクリプト (このページの裏面に抜粋を記載) はインポートされた SRTM DEM のラスタオブジェクト内部のヌルセルに作用し、穴を埋めることにより流水解析や地形表示、その他の処理に使うことの出来る連続した標高データを提供します。

いくつかの単純なタイプの穴は SRTM DEM だけで埋めることが出来ます。このような穴としては、孤立したヌルセル (周囲は標高値によって部分的または完全に囲まれています) で、内挿によって埋められます。他に、水面の穴のように一定標高のセルによって囲まれた複数のセルから成る穴もあります。これらの特殊な場合に加え、複数の参照用 DEM を使って穴を埋めることも出来ます。参照用 DEM のセルサイズや地理的範囲、座標参照系は SRTM DEM と一致する必要はありません。SRTM DEM 内の穴のあいたセルは、参照用 DEM 内の地理的位置に対応するセルの標高値によって置き換えられます。オプションとして、各空隙の周りに幅 1 ピクセルの緩衝帯を設けることによって、周囲の値になめらかにつなげるオプションもあります。各緩衝帯のセル値は、滑らかにつなげるために緩衝帯内の距離による重み付けを行って、SRTM と参照用 DEM のセル値を平均して計算されます。更に、SRTM と参照用 DEM の許容高度差の最大値を設定することも出来ます。この差を越える SRTM 値は最初ヌルに設定され、穴が埋められる際に参照用 DEM 値と置き換えられます。





SRTM の穴埋めスクリプトのダイアログウィンドウ。

TNT 製品のスクリプト言語 (スクリプト / クエリ) の使い方を紹介した多くのサンプルスクリプトが用意されています。これらのスクリプトは www.microimages.com/downloads/smlscripts.htm からダウンロードすることが出来ます。

SRTM 穴埋めスクリプトからの抜粋 (SRTMfill.sml)

この処理は SRTM ラスタ内を走査し、その中のヌル値を、対応する参照用 DEM のセル値で置き換えます。設定した参照用 DEM と重なっていない場合、または対応する参照用 DEM のセルがヌルの場合、SRTM のセルはヌルのままになります。

```
proc FillWithRefRaster() {

    print("Filling Voids with Reference Data\n");

    local class GEOREF outG, refG;
    local class POINT2D point;

    outG = GetLastUsedGeorefObject(outR);
    refG = GetLastUsedGeorefObject(RefR);
```

CRS が異なる場合、transparm を作成します

```
    local numeric same = 1;
    if (outG.CoordRefSys.Name != refG.CoordRefSys.Name) {
        same = 0;

        local class TRANSPARM trans;
        trans.InputCoordRefSys = outG.CoordRefSys;
        trans.OutputCoordRefSys = refG.CoordRefSys;
    }

    SetStatusMessage("Filling holes with Reference Data");
```

outR に重なるオブジェクト座標を見つけます

```
    local class RECT overlap = FindIntersectionOfRasters(outR, RefR, 1);
```

重なる領域を走査します

```
    local numeric i, j;
    for i = overlap.y2 to overlap.y1-1 {
        SetStatusBar(i, rows);
        for j = overlap.x1 to overlap.x2-1 {
```

セルがヌルの場合、参照用 DEM 内の対応するセルを見つけます

```
            if ( IsNull(outR[i,j]) ) {
                point = ObjectToMap(outR, j, i, outG);
```

同じでなければ正しい CRS へ変換します

```
                if (!same) {
                    point = trans.ConvertPoint2DFwd(point);
                }
```

地図座標を SRTM ラスタの行 / 列へ変換します

```
                point = MapToObject(refG, point.x, point.y, RefR);
                point.x = round(point.x);
                point.y = round(point.y);
```

point が有効であるか確認し、置き換えます

```
                if ( point.x >= 1 and point.x <= RefR.$Info.NumCols) {
                    if ( (point.y >= 1) && (point.y <= RefR.$Info.NumLins) ) {
                        outR[i,j] = RefR[point.y, point.x] * RefDataScale;
                    }
                }
            }
        }
    }
```

この処理は SRTM データと、ユーザーが指定したヌル空隙からの距離に基いてヌルセルの穴埋めに使用した参照 DEM を混合します。

```
proc BlendBufferZone( class RASTER zoneBufferR, numeric zoneNum ) {

    print(" Buffering in Zone " + NumToStr(zoneNum) +
        " of " + NumToStr(cellBufferDist) + "\n");
    local class POINT2D point;
    local class GEOREF outG, refG;

    outG = GetLastUsedGeorefObject(outR);
    refG = GetLastUsedGeorefObject(RefR);
```

CRS が異なる場合、transparm を作成します

```
    local numeric same = 1;
    if (outG.CoordRefSys.Name != refG.CoordRefSys.Name) {
        same = 0;
        local class TRANSPARM trans;
        trans.InputCoordRefSys = outG.CoordRefSys;
        trans.OutputCoordRefSys = refG.CoordRefSys;
    }

    SetStatusMessage("Computing new values in Buffer Zone");
```

outR に重なるオブジェクト座標を見つけます

```
    local class RECT overlap = FindIntersectionOfRasters(outR, RefR, 1);
    local numeric I = 1, J = 1;
```

重なる領域を走査します

```
    local numeric m, n, buffFactor;
    for m = overlap.y2 to overlap.y1 - 1 {
        SetStatusBar(m, rows);
        for n = overlap.x1 to overlap.x2 - 1 {
```

セルが緩衝帯内にある場合、混合処理を開始します

```
            if ( zoneBufferR[I, J] == 1 ) {
                point = ObjectToMap(outR, n, m, outG);
```

同じでなければ point を正しい CRS へ変換します

```
                if (!same) {
                    point = trans.ConvertPoint2DFwd(point);
                }
```

地図座標を SRTM ラスタの行 / 列へ変換します

```
                point = MapToObject(refG, point.x, point.y, RefR);
                point.x = round(point.x);
                point.y = round(point.y);
```

DEM 中の対応するセルが有効であることを確認します

```
                if ( point.x >= 1 and point.x <= RefR.$Info.NumCols) {
                    if ( (point.y >= 1) && (point.y <= RefR.$Info.NumLins) ) {
```

セルが緩衝区域中のどのゾーンにあるかによって決まる線形的な差と SRTM 値を足して、出力セルに指定します

```
                        buffFactor = (cellBufferDist - zoneNum + 1) * 1.0 /
                            (cellBufferDist* 1.0 + 1);
                        outR[m,n] = outR[m,n] + (buffFactor) *
                            ((RefR[point.y, point.x] *RefDataScale) - outR[m,n]);
                    }
                }
            }
            J++;
        }
        I++;
        J = 1;
    }
}
```