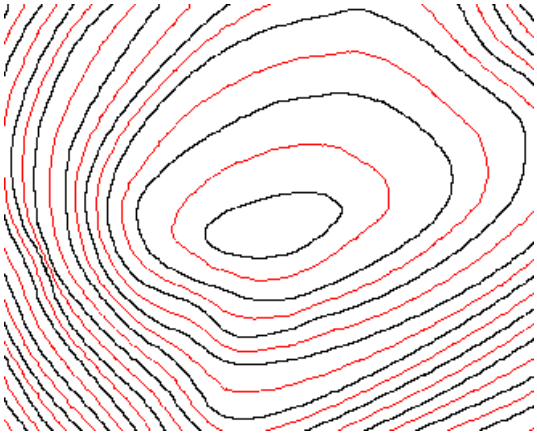
Sample Geospatial Script

Contours to DEM via Morphological Interpolation

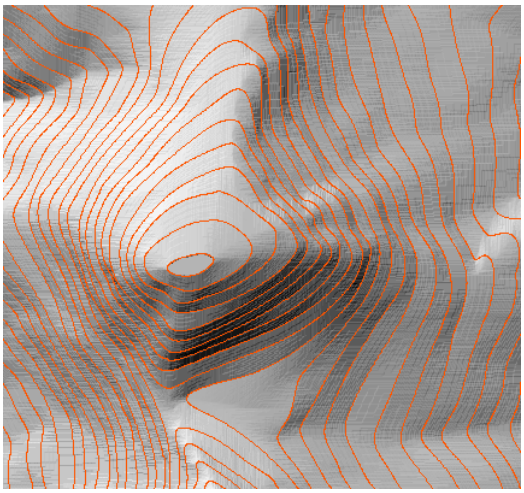
The MorphContour script (excerpted on the opposite side of this page) demonstrates a method of interpolating an elevation raster from contour lines using morphological functions, as described by William Barrett, Eric Mortensen, and David Taylor in the 1994 paper entitled *An Image Space Algorithm for Morphological Contour Interpolation* (http://citeseer.ist.psu.edu/178773.html). MicroImages implemented this algorithm as a sample script in order to evaluate its results at the request of several clients who referenced this paper.

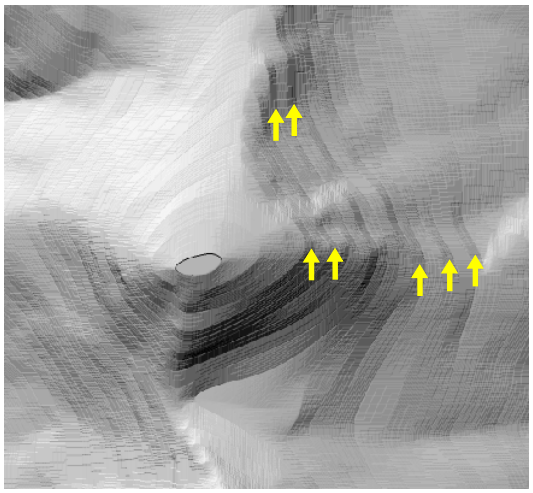


Raster with input contours (black) and first set of midlines (red) created by one pass of the morphological interpolation procedure. Cells defining a midline are assigned the average value of the enclosing contours.

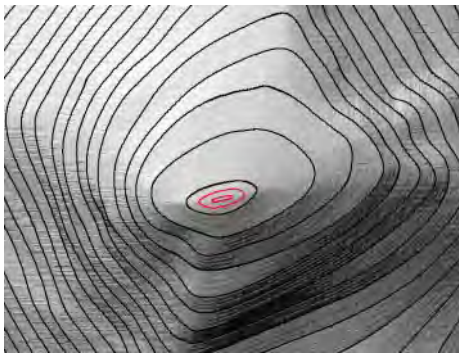
The script first transfers elevation values from contour lines in a vector object to corresponding cells in the output raster object, leaving the inter-contour areas initially blank. It then applies raster morphological functions including dilation and erosion to the inter-contour areas to find the midline between each pair of contours and assign those midline cells the average value of the two bounding contour elevations. The original contours and the new set of midlines (intermediate contours) then define a new set of inter-contour areas that are processed as before to find an additional set of finer midlines. This procedure is repeated until all possible midlines have been found.

As described in the above-cited paper, the morphological interpolation algorithm only treats areas lying between pairs of contours with different values. Thus areas surrounded by only a single closed contour (hilltops, the bottoms of closed depressions, and local areas at the boundary of the region being processed) are not interpolated. The procedure outlined in the paper simply leaves thes cells unfilled as holes in the DEM. As each of these areas left blank by the interpolation procedure is surrounded by cells with the same elevation, MicroImages added a routine to the script to fill each of these areas with the surrounding elevation value. Thus hilltops and other closed-contour areas are flat in the output elevation raster. To provide additional detail in these areas, you can use the TNTmips Editor to add contours with estimated elevation values to the vector contour object to smoothly fill in these areas prior to running the script.





Relief shading computed in the Topographic Properties process from an elevation raster produced by the MorphContour script from the vector contours (orange lines) shown overlaid on the left-hand image. Shading accentuates the abrupt changes in slope at the contour locations and alternating steeper and shallower slopes (arrows) that result from the inherent linear nature of the morphological interpolation procedure carried out by the script.



Hilltop area with original vector contours (black) and new contour lines (red) added to the vector object in the TNT Editor with estimated elevations to provide a more rounded shape to the hilltop before processing in the MorphContour script. Relief shading raster from DEM produced from these contours is also shown.

The morphological procedure described in this paper effectively performs a linear interpolation in the local slope direction between each pair of contours. As a result, in any given downslope transect the slope is nearly constant between each contour pair, and there may be large changes in slope at the contours themselves. The abrupt changes between constant-slope segments are evident when the interpolated elevation raster is shaded, as shown by the illustrations above.

Many sample scripts have been prepared to illustrate how you might use the features of the TNT products' scripting language for scripts and queries. These scripts can be downloaded from www.microimages.com/downloads/scripts.htm.

Script Excerpts for Morphological Contour Interpolation
(MorphContour.sml)

This function copies the input raster to the output raster (Rtemp) and morphologically erodes the the output raster cells that have a starting value equal to the maskValue (inter-contour cells). The function returns the number of cells that were eroded.

```
func MyErosion(Rin, Rtemp, maskValue) {

   RasterCopy(Rin, Rtemp);

   local numeric r, c, er, ec, tempValue, newValue, count=0;
   for r = 1 to NumLins(Rin) {
      for c = 1 to NumCols(Rin) {
         newValue = maskValue;
         if (Rin[r,c] == maskValue) {
            count++;
                        for each cell in the element
            for er = -1 to 1 {
               for ec = -1 to 1 {
                                    if the element value isn't null
                  if (!IsNull(element[er+2,ec+2])) {
                     if the element is within the bounds of Rin
                     if ( ((r + er) >= 1) and ((r + er) <= NumLins(W)) and
                           ((c + ec) >= 1) and ((c + ec) <= NumCols(W)) ) {

                        tempValue = Rin[r + er, c + ec];
                        if (tempValue < newValue) {
                           newValue = tempValue;
                        }
                     }
                  }
               }          end for ec
            }     end for er

            if (newValue != maskValue) {
               count--;
               Rtemp[r,c] = newValue;
            }
         }
      }          end for c
   } end for r

   RasterCopy(Rtemp, Rin);
   print("Erosion Count:", count);
   return count;
} end of MyErosion
```

This procedure copies the input raster to an output raster and then morphologically dilates the masked cells in output raster.

```
proc MyDilation(Rin, Rout, maskValue, minValue) {

   RasterCopy(Rin, Rout);
   local numeric r, c, er, ec, tempValue, newValue, count=0;
   for r = 1 to NumLins(Rin) {
      for c = 1 to NumCols(Rin) {
         newValue = minValue;
         if (T[r,c] == maskValue) {
                        for each cell in the element
            for er = -1 to 1 {
               for ec = -1 to 1 {
                                    if the element value isn't null
                  if (!IsNull(element[er+2,ec+2])) {
                     if the element is within the bounds of Rin
                     if ( ((r + er) >= 1) and ((r + er) <= NumLins(W)) and
                           ((c + ec) >= 1) and ((c + ec) <= NumCols(W)) ) {
                        tempValue = Rin[r + er, c + ec];
                        if (tempValue > newValue) {
                           newValue = tempValue;
                        }
                     }
                  }
               }          end for ec
            }     end for er

            if (newValue != minValue) {
               Rout[r,c] = newValue;
            }
         }
      }          end for c
   }  end for r
} end of MyDilation
```

This function sets the cells in the raster that symbolizes the new medial lines (Rin1) to the average value of the two contour values that created it.

```
proc MyMedialLine(Rin1, Rin2, Rmask) {

   numeric i, j;
   for i = 1 to NumLins(Rin1) {
      for j = 1 to NumCols(Rin1) {

         if (Rmask[i,j]) {
            Rin1[i,j] = (Rin2[i,j] + Rin1[i,j]) / 2;
         }
      }
   }
}
```

Portion of main procedure that applies the morphological functions to compute midlines for contour pairs.

```
   while (done) {

      done = 0;
      numRuns++;
   filling all inter-contour cells in mask rasters with the maximum value
      RasterCopy(A, W);
      RasterCopy(A, T);
      numeric i, j;
      for i = 1 to NumLins(A) {
         for j = 1 to NumCols(A) {
            if (A[i,j] == Min) {
               W[i,j] = Max;
               T[i,j] = Max;
            }
         }
      }

      while( MyErosion(W, Temp, Max) > 0) { }
      MyDilation(W, U, Max, Min);
      MyRasterXOR(W, U, V);
      MyMedialLine(W, U, V);
      done = MyAddToAccumulator(W, V, A, Min);

      print("Accumulator Count:", done, "\nRun Completed:", numRuns, "\n");

   }
```