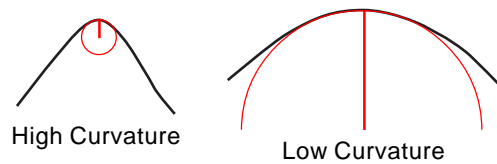


## Sample SML Script

# Terrain Curvature

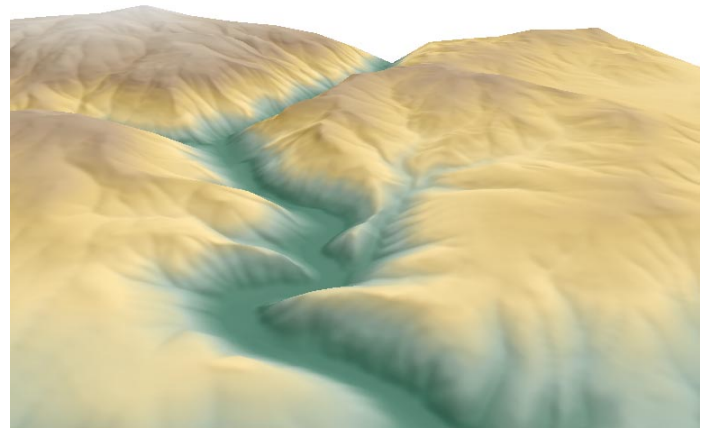
You can use SML scripts to implement custom processing routines tailored for different kinds of project materials. To illustrate custom processing of raster objects that represent terrain surfaces, MicroImages has created a sample SML script that computes profile and plan curvature values for the terrain surface.

Surface curvature is the curvature of a line formed by intersecting a plane (in some chosen orientation) with the terrain surface. The curvature value is the reciprocal of the radius of curvature of the line, so a broad curve has a small curvature and a tight curve has a high curvature value. The script allows you to choose the curvature unit: radians per meter or radians per hundred meters.

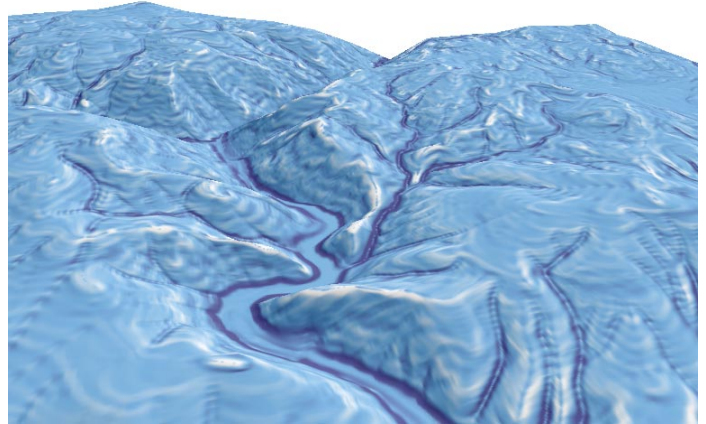
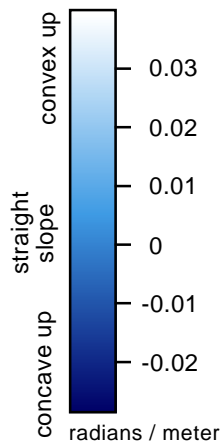


Profile curvature is the curvature in the vertical plane parallel to the slope direction. It measures the rate of change of slope and therefore influences the flow velocity of water draining the surface and thus erosion and the resulting downslope movement of sediment. Plan curvature (also called contour curvature) is the curvature of a contour line formed by intersecting a horizontal plane with the surface. Plan curvature influences the convergence or divergence of water during downhill flow.

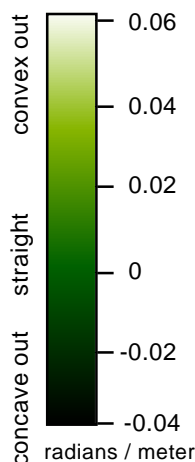
The sample SML script computes both curvatures in one pass through the elevation raster (DEM). Because most DEMs include errors and processing artifacts (such as terracing on slopes) that can affect the curvature computation, the script provides the option of applying an initial smoothing filter to the elevation values, and the choice of filter size. The curvature values are computed from a local best-fit mathematical surface for each cell neighborhood, a procedure that provides some additional smoothing.



3D perspective view of a sample terrain (elevation raster with color palette displayed with partial transparency over a shaded-relief image to create color-shaded relief).



**Profile (vertical) curvature** raster produced by the SURFCURVE script, displayed with color palette over the terrain. Positive values indicate convex-upward surfaces. Horizontal banding on slopes results from artifacts in production of the source DEM that produced terraced rather than more uniform slopes.



**Plan (contour) curvature** raster produced by the SURFCURVE script, displayed with color palette over the terrain. Positive values indicate convex-outward surfaces.

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/freestuf/scripts.htm](http://www.microimages.com/freestuf/scripts.htm).

## Script Excerpt for Terrain Curvature (surfcurve.sml)

Procedure to read 3x3 kernel of DEM values surrounding current cell.

```
proc readkernel (class raster K, numeric line, numeric colm){
  numeric z1 = K[line-1, colm-1];
  numeric z2 = K[line-1, colm];
  numeric z3 = K[line-1, colm+1];
  numeric z4 = K[line, colm-1];
  numeric z5 = K[line, colm];
  numeric z6 = K[line, colm+1];
  numeric z7 = K[line+1, colm-1];
  numeric z8 = K[line+1, colm];
  numeric z9 = K[line+1, colm+1];
}
```

Function to check surrounding area of kernel. If no difference, there is no need to do some computations. Also used as a safety against divide by zero.

```
func levelcheck(class raster L, numeric x, numeric y) {
  if (z5 != z1) return true;
  if (z5 != z2) return true;
  if (z5 != z3) return true;
  if (z5 != z4) return true;
  if (z5 != z6) return true;
  if (z5 != z7) return true;
  if (z5 != z8) return true;
  if (z5 != z9) return true;
  return false;
}
```

Procedure to compute coefficients for local quadratic surface fit to kernel.

```
proc coeff(numeric dx, numeric dy) {
  a = (z1 + z3 + z4 + z6 + z7 + z9)/(6 * dx^2) - (z2 + z5 + z8)/(3 * dx^2);
  b = (z1 + z2 + z3 + z7 + z8 + z9)/(6 * dy^2) - (z4 + z5 + z6)/(3 * dy^2);
  c = (z3 + z7 - z1 - z9)/(4 * dx * dy);
  d = (z3 + z6 + z9 - z1 - z4 - z7)/(6 * dx);
  ee = (z1 + z2 + z3 - z7 - z8 - z9)/(6 * dy);
  f = (2 * (z2 + z4 + z6 + z8) - (z1 + z3 + z7 + z9) + (5 * z5))/9;
}
```

Procedure to compute curvature values from coefficients of best-fit quadratic surface.

```
proc makeCurv(class raster D) {
  for each D[line,col] {
    if (line != 1 && line != D.$Info.NumLines && col != 1 && col != D.$Info.NumCols) {
      readkernel(D, line, col);
      if (nullcheck()) {
        if(levelcheck(D, line, col)) {
          coeff(D, line, col, cellX, cellY);
          prfdenom = round(10^7*((sqr(ee) + sqr(d))*((1 + sqr(d) + sqr(ee)) ^ 1.5)))*10^-7;
          plndenom = round(10^7*((sqr(ee) + sqr(d))^1.5) * 10^-7);
          if (prfdenom == 0 || plndenom == 0) {
            if ((a > 0 && b > 0) || (a < 0 && b < 0)) {
              Profile[line, col] = - (a + b) * scale;
              Plan[line, col] = -32768;
            }
            else {
              Profile[line,col] = 0;
              Plan[line, col] = 0;
            }
          }
        }
      }
    }
  }
}
```

curvature computation for typical cells

```
else {
  Profile[line, col] = scale * -2 * ( a * sqr(d) + b * sqr(ee) + c * d * ee) / prfdenom;
  Plan[line, col] = scale * -2 * (b * sqr(d) + a * sqr(ee) - c * d * ee) / plndenom;
}
else {
  Plan[line, col] = 0;
  Profile[line, col] = 0;
}
else {
  Plan[line,col] = -32768;
  Profile[line,col] = -32768;
}
else {
  Plan[line,col] = -32768;
  Profile[line,col] = -32768;
}
```

case for flat kernel area

case for kernel including null cells, not enough data to compute curvature

case for edge cells, not enough data to compute curvature

Main Program.

```
GetInputRaster(DEM);
cellX = ColScale(DEM);
cellY = LinScale(DEM);

GetOutputRaster(Profile, DEM.$Info.NumLines, DEM.$Info.NumCols, "32-bit float");
GetOutputRaster(Plan, DEM.$Info.NumLines, DEM.$Info.NumCols, "32-bit float");
SetNull(Profile,-32768); SetNull(Plan,-32768);
```

```
scale = PopupNum("Enter 1 to compute curvature in radians/m or
  \nenter 100 for curvature in radians/100 m", 1, 100, 0);
doSmooth = PopupYesNo("Use averaged DEM values for curvature?", 1);
```

```
if (doSmooth == 1) {
  class RASTER Temp;
  local numeric filtsize = PopupNum("Size of averaging window in
  cells? \n(3, 5, 7, or 9)", 5, 3, 9, 0);
```

```
CreateTempRaster(Temp, DEM.$Info.NumLines,
  DEM.$Info.NumCols, DEM.$Info.Type);
```

compute averaged DEM value for non-null cell  
else assign null value to Temp raster

```
numeric i, j;
for i = 1 to DEM.$Info.NumLines {
  for j = 1 to DEM.$Info.NumCols {
    if ( !IsNull(DEM[i,j]) ) then
      Temp[i,j] = -32768;
    else
      Temp = FocalMean(DEM, filtsize, filtsize);
    }
  }
}
```

```
SetNull(Temp, -32768);
makeCurv(Temp);
CloseRaster(Temp);
```

compute curvatures from smoothed temporary raster

end of if (doSmooth())

```
else
  makeCurv(DEM);
}
```

compute curvatures directly from DEM

```
SetNull(Profile, -32768);
SetNull(Plan, -32768);
CreateHistogram(Profile);
CreateHistogram(Plan);
CreatePyramid(Profile);
CreatePyramid(Plan);
CopySubobjects(DEM, Profile, "georef");
CopySubobjects(DEM, Plan, "georef");
```