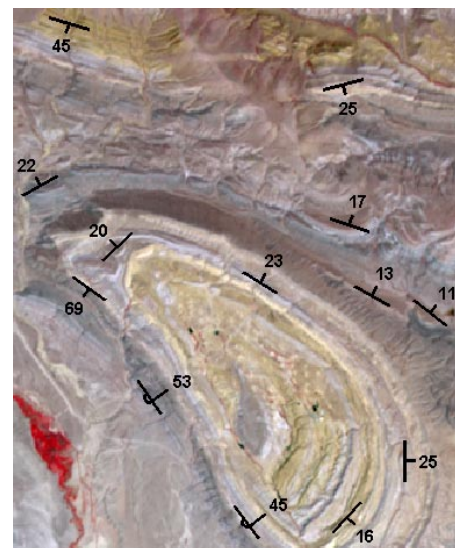


## Sample Tool Script

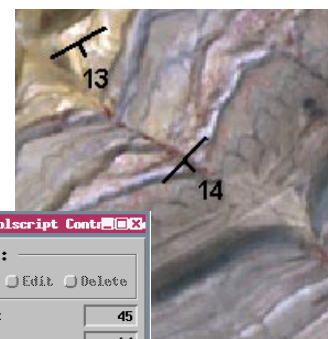
# Measure Strike/Dip of Geologic Features

The patterns traced across the landscape by layered rock units provide key evidence about surface and near-surface geologic structure. Geologists can map these surface features remotely using aerial or satellite images, but in order to quantify the structural geometry, they need measurements of the strike and dip (measures of 3D orientation) of the rock layers at many locations. This Tool Script enables geologists to measure and record the strike and dip of bedding or other map-scale planar features (such as joints, faults, dikes, and others) using a reference image overlaid on an elevation raster in the View. Key portions of the script are excerpted on the opposite side of this page. The script solves the classic “3-point problem” of structural geology: given the x-y-z coordinates of three non-colinear points on a plane, compute the strike and dip of the plane.

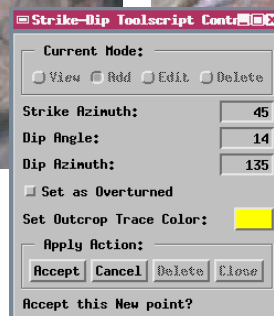
In the Tool Script’s default Add mode, you use a standard polygon tool provided by the script to indicate in the View window the locations of the three required points (as triangle vertices) on a planar geologic feature. The script reads the elevation for each point from the elevation raster (which must be the first layer in the group), determines the corresponding plane, computes its strike azimuth, dip angle, and dip azimuth, and shows these values in the script’s control window. The script also computes the outcrop trace of the computed plane over the terrain surface in the vicinity of the measurement point and draws this trace in the view. Comparing this trace to the local outcrop pattern shown by the reference image provides you with a visual quality-control assessment of the accuracy of the computed orientation of the plane. The polygon tool remains active to allow you to adjust the triangle vertices, if necessary, yielding a revised set of orientation values and a new outcrop trace. When you accept the orientation measurement, a point element is added to the designated Strike/Dip vector object and the orientation values are stored in an attached record in the associated point database. These points are also automatically styled by CartoScript with the appropriately-oriented and labeled symbol for the strike and dip of bedding. The Tool Script can be revised easily to reference a different CartoScript for styling symbols for other planar geologic features.



Measurement locations are added as point elements to a vector object and automatically styled using the appropriately-oriented and labeled strike and dip symbol from an accompanying CartoScript.



The point symbol is placed at the center of the triangle and styled by CartoScript.



The Tool Script’s control window provides View, Add, Edit, and Delete modes.

CartoScripts for various geologic map features are available for free download from:

[www.microimages.com/freestuf/cartoscripts/](http://www.microimages.com/freestuf/cartoscripts/)



In Add mode, use the polygon tool to draw a triangle on the desired planar feature (above left). Use the Line/Polygon Edit Controls (left) to adjust the triangle if needed, then press [Apply] or press the right mouse button. The computed values for the plane are shown in the Tool Script’s control window (right) and the outcrop trace is drawn in the View in the selected color (above right). Press [Accept] on the control window to store the point and its data.



In the Tool Script’s View mode, you can select one or more data points and turn on their outcrop traces for mutual comparison and to help map outcrop patterns.

The outcrop trace line for each measurement location is also added to a separate vector object with your selected line color. In the Tool Script’s View mode, you can left-click on any strike-dip symbol to select it and use the Accept button on the Tool Script’s control window to toggle the point’s outcrop trace on or off. In this manner you can turn on and view several outcrop traces simultaneously. These computed outcrop traces can help you trace outcrop patterns and contact positions through areas of poor exposure (due to vegetation or soil cover) that might surround your measurement points.

The Tool Script’s Edit mode allows you to select and edit existing strike-dip measurements. Selecting a point in this mode reactivates the polygon tool with its former vertex locations so that you can adjust its position and compute revised orientation measurements and a revised outcrop trace. You can also delete points and their accompanying outcrop traces using the Tool Script’s Delete mode.

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 Excerpts for Strike-Dip Tool Script (StrikeDipTool.sml)

### Function to get the x and y screen coordinates from a polyline vertex

```
func class POINT3D setXY(numeric num) {
  local class POINT3D tmp = poly.GetVertex(num - 1);
  return tmp;
}
```

### Function to get the z value from the elevation raster at the location of a polyline vertex

```
func class POINT3D setZ(class POINT3D pt3d, numeric num) {
  local class POINT3D tmp;
  tmp.x = pt3d.x; tmp.y = pt3d.y;

  convert from screen to view coords
  tmp = TransPoint2D(tmp, ViewGetTransViewToScreen(View, 1));

  convert from view to map coords
  tmp = TransPoint2D(tmp, ViewGetTransMapToView(View,
    rasterLayer.CoordRefSys, 1));
```

```
  local class Georef georef;
  local class POINT2D objcoord;
  georef = GetLastUsedGeorefObject(DEM);
  objcoord = MapToObject(georef, tmp.x, tmp.y, DEM);
```

### with the object coordinates we can get the cell value

```
  numeric cellval = DEM[objcoord.y, objcoord.x];
  tmp.z = cellval; a raster cell value
  return tmp;
}
```

### Function to compute linear coefficients of the equation of the plane from the three 3D points in the polyline tool

```
func class POINT3D computePlanarCoefficients() {
  local class POINT3D pt1, pt2, pt3;
  local class POINT3D pt21, pt31;
  local class POINT3D ctrPt;

  pt1 = setXY(1);
  pt1 = setZ(pt1,1);
  pt2 = setXY(2);
  pt2 = setZ(pt2,2);
  pt3 = setXY(3);
  pt3 = setZ(pt3,3);

  call functions to get X,Y coordinates
  from three polyline vertices and to
  find corresponding Z value from
  elevation raster

  if one of the z values is bad, then stop - return invalid ctrPt
  if (!(isZValid(pt1.z) && isZValid(pt2.z) && isZValid(pt3.z))) {
    string s$ = "Error: One (or more) of the points has an invalid elevation.";
    PopupMessage(s$);
    ctrPt.x = NullValue(DEM);
    ctrPt.y = NullValue(DEM);
    ctrPt.z = NullValue(DEM);
    return ctrPt;
  } else {
    pt21 = computeDifference(pt2, pt1);
    pt31 = computeDifference(pt3, pt1);

    Compute linear coefficients for the equation of the plane
    a = Determinant( pt21.y, pt21.z, pt31.y, pt31.z );
    b = Determinant( pt21.x, pt21.z, pt31.x, pt31.z ) * -1;
    c = Determinant( pt21.x, pt21.y, pt31.x, pt31.y );

    compute the center point of the triangle, this is the point that is
    created and attributed with the information about strike and dip
    ctrPt = computeCenterPoint(pt1, pt2, pt3);
    createPlaneRaster(pt1, pt2, pt3);
    return ctrPt;
  }
}
```

### Procedure to compute the strike and dip for a nonvertical plane

```
proc computeNonVerticalPlane() {
  dipX = a / c; partial derivative of dip in x-direction
  (opposite of the slope component)

  if ( b == 0 ) plane is parallel to y-axis (north-south), dipY is 0
  {
    if ( a == 0 ) plane is parallel to both x and y and thus is horizontal
    {
      dipAng = 0;
      strikeAng = null; strikeAng is undefined
      dipDir = null;
    }
    else { plane is not horizontal, strike is north-south
      dipAng = atand( dipX ); compute dip angle from dipX

      if ( dipAng < 0 ) then dipAng = -1 * dipAng;

      if ( dipX > 0 ) {
        dipDir = 90; dip direction is due east
        strikeAng = 0;
      }
      else {
        dipDir = 270; dip direction is due west
        strikeAng = 180;
      }
    }
  }
  else { strike not parallel to y-axis, dipY is nonzero
    dipY = b / c; partial derivative of dip in y-direction
    (opposite of the slope component)

    dipAng = atand( sqrt( sqr(dipX) + sqr(dipY) ) ); compute dip angle
    from dipX and dipY

    if ( dipAng < 0 ) dipAng = -1 * dipAng;
    compute azimuth of dip direction from dipX and dipY; value
    returned by atand() is between -90 and +90, so must adjust
    result to get positive azimuth value between 0 and 360
    depending on which quadrant dip vector is in.
    dipDir = atand( dipX / dipY );

    if ( dipDir == 0 ) { dip direction is due north or south
      if ( dipY < 0 ) dipDir = 180; dip direction is south
    }
    if ( dipDir < 0 ) { dip direction in SE or NW quadrants,
      must adjust value
      if ( dipX > 0 ) {
        dipDir = 180 + dipDir; dip direction in SE quadrant
      }
      else {
        dipDir = 360 + dipDir; dip direction in NW quadrant
      }
    }
    dip direction in SW quadrant, must adjust value

    else if ( dipX < 0 ) dipDir = 180 + dipDir;
    compute strike angle from dip direction and convert
    negative values to positive azimuth in range 0 to 360
    strikeAng = dipDir - 90; strike direction from right-hand rule

    if ( strikeAng < 0 ) strikeAng = 360 + strikeAng;
  }
}
```