Drawing up cave surveys by computer: the Tunnel software suite

Dave Loeffler

Once you are back on the surface after a surveying trip, there are two main parts to the job of drawing up your survey. The first is a purely mathematical one, that of calculating the most probable positions of your stations based on the tape, compass and clinometer data (or whatever other combination of instruments you are using). This job is ideally suited to computer calculation, and many software packages exist to solve the problem, whose merits have been extensively debated in past Compass Points articles. Once this first task is done, the next task is more artistic: to take the centreline you have calculated, and draw in the walls and other passage details around it. This is harder to automate to quite the same extent as calculating the centreline. Nonetheless a dedicated software package can make the task much easier. This article describes one such program, "Tunnel", which was written largely by Julian Todd with the collaboration of Martin Green and the author.

It is, of course, possible to draw up surveys using any of the standard computer drawing packages, without using any cave-specific software. This approach works well for certain caves - particularly small, simple caves that are fully surveyed within a short period. However, these packages become extremely frustrating for larger, more complex systems, or for long-term surveying projects where a major loop closure can require the whole survey to be redrawn. The difficulties encountered in correcting the first sheet of the Red Rose Cave and Pothole Club's Easegill survey when one of the entrance locations was found to be in error demonstrate this. The aim of Tunnel is to make this process easier, in several main ways. One of the most important is that the drawn-up survey can be warped if changes are made to the centreline (for example, due to a changes in the distribution of loop closure errors). In this article, I aim to describe Tunnel's main features, and to provide a quick-start guide to drawing up a cave using Tunnel.

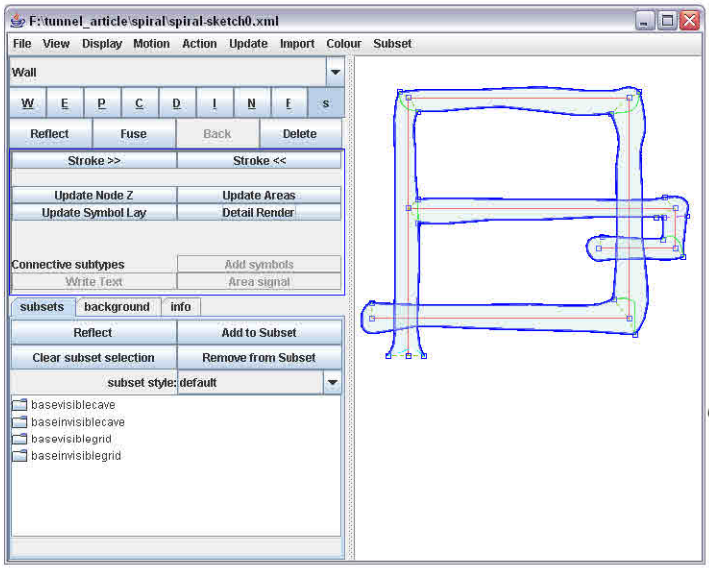
Importing your centreline data

Tunnel was designed to import centreline data in Survex's .svx format, but it can also import .TOP (Toporobot) and .PRJ (Walls) files; other import filters could be added if required. See [1] for a description of the Survex file format. When Tunnel imports a Survex file, it processes the data into its own format, which is based on XML. It also splits up the data into one file for each *begin/*end prefix block in the Survex file, which would typically represent one passage or survey trip (it is considered good practice to split your Survex files up in this way anyway) and puts each of these in a separate folder. Each of these folders will hold sketches and other files relating to this section of the cave. The advantage of using lots of small files is that it allows teams of people to draw up different parts of the same cave simultaneously, particularly when combined with versioning systems such as CVS [2].

Drawing a sketch

The core of Tunnel is its sketch drawing window, which allows you to draw in the walls of your cave using the mouse. We usually draw up each individual survey trip by hand around a print-out of the centreline. These sheets are then scanned and traced into individual Tunnel sketches. It is also possible to trace from scans of original underground survey notes, or even draw freehand; the latter requires some practice, but that is true for all computer drawing packages. What is shown in the sketch window differs from the final output, as different line types are highlighted in colour, but all labels and symbols are shown (unlike one of Tunnel's competitors, Therion, where labels are not visible until the final render, which makes accurate placement difficult – see [4] for a discussion of Therion's capabilities).

Tunnel supports a range of line types. Some of these are self-explanatory: a thick solid line for walls (dotted for estimated walls), a thinner line for passage detail, and the usual pitch and aven indicators with their tick marks. In addition, Tunnel calculates the two-dimensional shapes that these lines surround, so that it can work out which parts are inside the cave; by default, Tunnel shades in passages light grey in the final rendered output, to make them stand out against the white background (this is, of course, customisable). The two other main line types are invisible lines, which form the boundary of an area without having a line visible in the final output (so that, for example, areas of sand or mud can be defined), and connective lines, which are ignored in the area calculation and which serve to tie walls and other parts of the drawing to the centreline. Tunnel uses these connective lines to determine which areas should be drawn over or under others, based on the height of the centreline itself. They are also needed by Tunnel's warping algorithm used for loop closures. See Figures 1 and 2 for some examples of the area shading system.



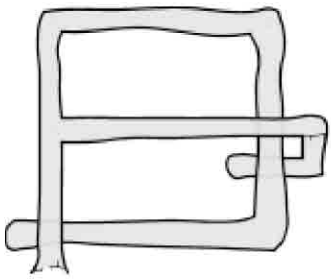


Figure 1: Here I have created a (rather contrived) example of a centreline that spirals downwards, with a side passage crossing over the main one then looping back under it. Tunnel automatically calculates which passages should be rendered above and below, based on the height of the centreline, to produce the output on the right. Note that the lower passages are still visible, as by default all passages are drawn slightly translucent; passages can be drawn opaque if desired, or made more transparent. The latter is useful for very complex surveys with many superimposed levels.

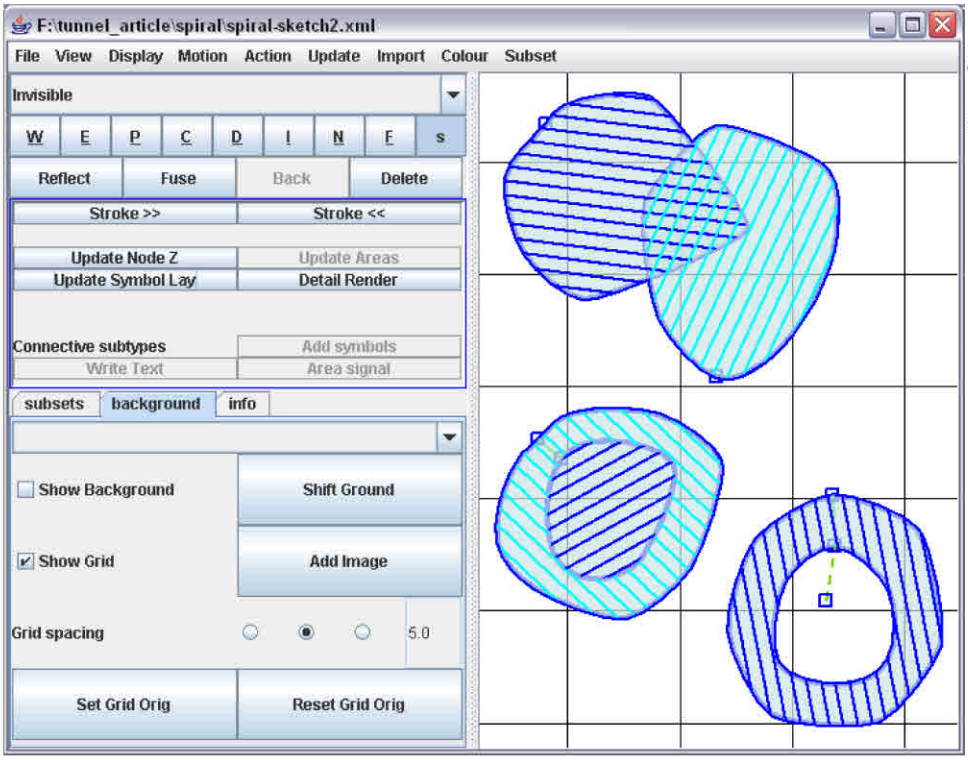


Figure 2: More examples of the area-detection algorithm. As Tunnel detects areas more or less by following walls, it assumes all areas are simply connected (that is, they have no "holes"). Areas may overlap, as in the first example, to allow for passages on multiple levels. Connecting two loops of wall with an invisible line, as in the second example, defines a ring-shaped area, but the central hole is also assumed to be passage; to draw a pillar, we also need to mark the inner area as rock, using a connective line with an "area signal".

Symbols

Tunnel allows a variety of types of symbols to be added to survey sketches. Some are added as single symbols (e.g. stream, slope and breeze arrows), but others are placed randomly to fill a specified area (e.g. sand, mud or boulders).

The algorithm for laying out random fields of symbols is a particular strength of Tunnel. It is actually a surprisingly difficult problem: how do you draw a realistic-looking field of random boulders? Placing them by hand is intolerably tedious. It is possible to draw a small section of reasonably random-looking boulders and tile this to cover the entire area, but this produces a rather odd effect, reminiscent of cheap wallpaper, as the eye recognises the similarities in the pattern. On the other hand, most genuinely random algorithms either produce lace-like patterns with numerous holes or perform poorly on areas of convoluted shape, or both. The algorithm Tunnel uses covers the area with a lattice, essentially approximating its shape with a bitmap, then places symbols at some random offset from each point of the lattice. This works remarkably well in practice (see Figure 3), especially when the size and alignment of the symbols is varied randomly at the same time.

The default set of symbols available within Tunnel covers most of those specified in the UIS standard [3], such as pools of water, various sizes of boulders and pebbles, stalactites and so on, but if this isn't enough, it is easy to design new ones: just create a new sketch in the "gsymbols" folder containing a drawing of the symbol, and add a line to the default style definition file specifying how it is to be laid out. The syntax of this file is quite complicated, but it is usually possible to just copy the code for an existing symbol that behaves in the same way.

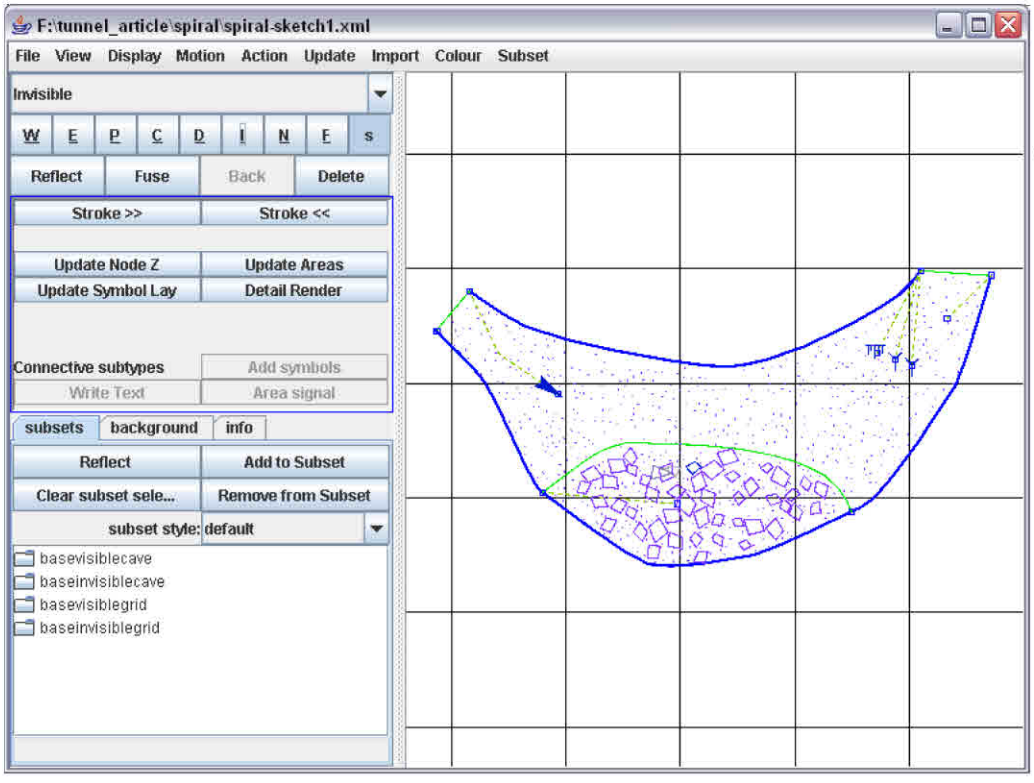


Figure 3: An example of laying out symbols in Tunnel. Notice the invisible line defining the area that contains the boulder symbols (this will not be shown in the final output).