Automated relief representation for visualisation of archaeological monuments and other anthropogenic forms

Nicolas Regnauld\textsuperscript{a}, William A. Mackaness\textsuperscript{a,}\textsuperscript{*}, Glen Hart\textsuperscript{b}

\textsuperscript{a}Department of Geography, The University of Edinburgh, Drummond Street, Edinburgh, EH8 9XP, Scotland, UK
\textsuperscript{b}Ordnance Survey, Romsey Road, Maybush, Southampton, SO9 4DH, UK

Accepted 13 September 2001

Abstract

Both within the Royal Commission on Ancient and Historic Monuments for Scotland (RCAHMS) and the Ordnance Survey (OS) there are specific requirements to record Historical Monuments (such as ramparts and mounds) and various anthropogenic forms (such as road and rail embankments). The traditional method of relief representation within archaeology employs ‘hachures’ (a hachure being a hand drawn line along the line of steepest gradient of a slope and in the direction of that slope). Hachuring has proved to be a highly effective form of representation—superior in specific circumstances, to alternatives such as contours and relief shading. However, it is acknowledged to be extremely time consuming in the context of current GIS solutions. Developments in automated cartography and digital databases such as the UK’s Digital National Framework, have led to a requirement for an algorithm that automatically generates hachures with the minimum of human input, and the simplest of database requirements. This paper reports on a method for automatic relief representation required in order to visualise archaeological monuments within a GIS. The program takes as input the isolines defining the top and bottom of the slope and populates the region with hachures whilst addressing a comprehensive set of design issues. Results of the implementation are presented and evaluated against hand drawn results. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Relief depiction; Hachures; Digital cartography; Qualitative methods
1. Introduction

The shape of the ground at both coarse and fine scale is a manifestation of geographic process and anthropogenic activities. Representation of relief is probably the most difficult problem in mapping and stems from the need to represent three dimensions in a two-dimensional media. The three major elements of a surface are: slope, height and the shape of the surface formed by the combinations of elevations and gradients (Robinson, Sale, & Morrison, 1978). Many cartographic methods have been developed in order to depict the relief of both natural and artificial features (Brandes, 1983; Hutchinson & Gallant, 1999) in order to unravel the past and represent the present. Early relief representations portrayed mountains as seen from the side. This type of representation was used in various incarnations such as ‘sugar loaf’ mountains and fish scales until the Renaissance (Imhof, 1982). With the development of planmetric maps the problem of how best to represent relief became more complex. Hachuring techniques previously used to illustrate slopes on the oblique views were applied to planmetric maps. Hachures are lines of varying width and length used to depict slope steepness. When many are drawn, they collectively show the forms of the surface configuration giving emphasis to relative gradient and distribution of shape. Hachures form the basis of archaeological models and in topographic maps they are used to symbolise very steep slopes such as cliffs, quarries, railway cuttings, levees, and situations where contour lines would merge (Imhof, 1982).

The appropriateness of hachuring and other techniques depends on the relative size of the morphological features and the thematic intent of the map. Lawrence (1979) considers these various methods under two categories: those in which a visual impression of relief is presented (such as hachures), and those where precise values are set out (such as contours). Contours convey information about the elevation of a terrain. It is possible to infer relative steepness and general form from the proximity and shape of height contours, but they do not directly symbolise the features they describe. Often the resolution of the underlying digital terrain model or the contour interval used is insufficient to capture the fine and subtle morphology of, say, an archaeological site. A similar line of reasoning is applicable in the use of relief shading. In these instances hachuring is a more appropriate form of qualitative representation. Thus the technique is seen as complementary to some of the more conventional techniques used to represent a surface.

A range of OS products use hachures to represent ‘small scale’ changes in slope; hachures are found on 1:1250 (Superplan—paper map), 1:10000 (LandPlan—paper map) and 1:50000 (Landranger—paper map) products. The 1:25000 (Explore and Outdoor Leisure—paper map) also contain hachures. The technique is used extensively by the Royal Commission on Ancient and Historic Monuments for Scotland (RCAHMS) and by English Heritage in the creation of antiquity models, both detailed and general form. Within archaeology, hachures have remained a comprehensible and acceptable form of representation to the archaeologist and allows a reader to identify relatively small scale features within that terrain (Putnam, 1988). Thus it is important to emphasise that many modern day map products utilise both quantitative and qualitative methods of impression.
A systematic approach to hachuring was proposed by Lehmann in 1799 (Lyons, 1914; Robinson et al., 1978). Lehmann defined a formal set of rules, described in detail by Imhof (1982) which was subsequently used in the development of automated solutions such as the method proposed by Yeoli (1985). This solution ‘lacked the refinement and subtleness of a good manual product’ (Yeoli, 1985, p. 121), and required editing of the contours. It also used ‘valley lines’ and ‘lines of discontinuity’ to resolve ambiguities in the symbolisation process. An embryonic hachuring solution was developed by O’Loughlin and Mackaness (1999) but lacked many of the aesthetics of a hand drawn solution. Hurni, Neumann, and Hutzler (1999) presents a recent attempt where the traditional technique of cliff drawing has been automated. Similarly, this process also requires human intervention via an experienced operator. The digital ambitions of the Ordnance Survey led to a requirement for a hachuring algorithm that utilised Digital National Framework data (DNF). DNF is the fine scale national database of the Ordnance Survey, re-engineered from the national topographic database covering the UK. An important specification was that the algorithm required minimum human interaction in the creation of solutions that were of the highest aesthetic quality. This requirement has provided the impetus for the work presented here, which begins by outlining a strategy, before describing the algorithm itself. The results are presented together with an evaluation.

2. Implementation strategy and analysis

The implementation strategy used was based on the ideas of Henderson (1991); an initial set of observations was used as a basis for specifying a design. Once the design was implemented, a first set of results was produced which were presented as part of a formal evaluation. Issues arising in the evaluation provided the basis for further refinement of the algorithm. This cyclic process (Fig. 1) is essential in environments such as these—where ‘success’ tends to be subjective and for which it is very difficult to define meaningful cartometric measures. Furthermore, the process is an excellent way of drawing consensus in the requirements analysis phase, and in understanding the context of such work in both a production environment and in optimising data gathering techniques in the field.

The first phase of research combined the formal definitions of Lehmann (1799) with observations of how hachures had been drawn in contemporary and historical maps in order to form a basic set of design rules. Two paper maps were chosen: Dorset sheet LV1. 2., 1:2500, First edition 1889, and Plan SY9682, 1:2500, Isle of Purbeck, published in 1960. They were specifically chosen because they contained a broad variety of archaeological and anthropogenic forms (old and modern) that vary considerably in shape, convolution, form and size. The analysis of these images continued throughout the implementation in order to assess intermediate results and modify the rules governing the process. The following observations were made:

*Hachures and their orientation.* Hachures are straight lines linking a top slope line to a bottom slope line, each stroke following the steepest gradient. The hachures
tend to be drawn perpendicular to the slope line. Otherwise hachures are orientated to approach an average value locally perpendicular to the top slope and locally perpendicular to the bottom slope.

Variation of orientation. There is an interdependence between adjacent hachures that modifies the previous rule. The orientation of the hachures varies smoothly along the embankment, to provide an aesthetic representation of the whole slope. This rule only became apparent once results were created based on the first rule.

Spacing of hachures. Hachures should not overlap, nor be in such proximity as to visually collide with one another. Looking at the whole slope, it appears that hachures are regularly spaced. In fact they are not, even when taking into account the curvature of the embankment. For example in Fig. 2, it appears that the intervals between consecutive hachures are regular, but on close inspection we notice that the size of intervals vary according to the length of the hachures. The longer the hachures, the wider the interval between them. The smaller the items the closer they need to be in order to be associated with the same entity.

Width of hachures. The length of each stroke corresponds to the local horizontal distance between the top and bottom of slope. The hachure should follow the convention of being thicker at the top of the slope.

Bend problem. When the embankment’s shape contains bends, the spacing between hachures increases on the convex side of the bend and decreases on the concave side. This difference increases with the ‘tightness’ of the corner and the distance between the top and bottom of slope. The degree of variation depends on symbol width, resolution of the medium, and limit of eye perception; the hachures need to be sufficiently dense to ensure continuity, and sufficiently spaced to avoid over inked regions on the convex side. Fig. 3 shows different solutions for solving this problem. Some hachures can be shortened (touching only the convex slope limit), or the corner can be filled by a tree-like arrangement of small hachures.
Differences from one embankment to another. Depending on the distance between the top and the bottom of the embankment, hachure density can vary dramatically from one embankment to the next. Fig 4 shows a section of the long embankment surrounding Corfe castle, and the thin linear form of a railway passing beneath the castle.

Symbol shape and width. In the nineteenth century, hachures were drawn as squiggly lines (Fig. 4). A refinement to this technique was to draw them as straight lines with constant width, apart from the extremities of the hachure at the top of the slope which was given an increased width to convey the top of the slope. The current representation used by the Ordnance Survey varies depending on scale. On maps at 1:2500, the shape of a drop is given to the top extremity of the hachures, while on 1:10 000 maps, the whole hachure is represented as a triangle, the base being at the top slope and the apex at the bottom.

3. A strategy for automation: requirements analysis

The global process for generating hachures autonomously (with minimum human input) on a full dataset can be viewed in seven stages, outlined below and detailed in the next section.

1. Formatting the input data: the methodology has been developed to generate hachures for closed regions defined as having a top and a bottom line. The initial data provided by the Ordnance Survey contains lines which are labelled as “Top of Artificial Slope” or “Bottom of Artificial Slope”, but with no information about the embankment itself. The first stage is to automatically detect the group of lines that form the top and bottom of a single embankment.
2. **Assign values to the parameters**: during the first stage, an analysis of the embankment is done to determine the parameters that should be used for the hachuring algorithm on a case by case basis. Three parameters are defined: The step, specifying the ideal frequency of hachures along the slope extent. The minimum spacing, specifying the threshold for the smallest distance between two consecutive hachures. The base length, specifying the length of the base of the triangle represented at the top side of the hachures.

3. **Build initial hachures**: the algorithm works by populating the region with hachures using a very simplistic approach. The algorithm then proceeds to incrementally refine the solution. This first set of hachures is generated by
going through the top slope line and drawing, at regular intervals, a hachure perpendicular to the top line of the slope, and ending on the bottom line. The following steps gradually refine this first result.

4. **Solve the intersections**: consecutive hachures drawn perpendicular to the top line of the slope can intersect in different configurations. This often occurs when the top of slope is convoluted rather than turning gently, and especially where there is a large distance between the top and bottom lines. The algorithm detects the intersections and solves them by aggregating any pair of consecutive intersecting hachures into an average one (Fig. 6 is the product of Fig. 5 using this approach).

5. **Smooth the rate of change of orientation between consecutive hachures**: because the hachures have been generated independently from each other, the result is not pleasing to the eye. To improve the result, we have added a step where the orientation of each hachure is refined by changing its orientation relative to its two immediate neighbouring hachures. This has the effect of smoothing the variation of orientation between hachures.

6. **Insert hachures in gaps**: as a result of the previous stages, the hachures are no longer regularly spaced, which can lead to ‘gaps’ appearing in the top and bottom line of the slope. The hachures have been drawn at regular intervals from the top line, but the intersection solving process may generate gaps. Similarly at the bottom of slope, gaps can occur because of bends in the shape of the embankment (see the bend problem in the analysis section). In both cases, these gaps have to be filled in order to visually preserve the continuity of both top and bottom lines of the slope.

7. **Smoothing the spacing between hachures**: once the gaps have been filled, a problem remains with the lack of regularity in the intervals between hachures. When a hachure is inserted to fill a gap on one side, it creates a local congestion on the other. We deal with this problem by centring each hachure in the

---

Fig. 5. Initial hachures (Images derived using Ordnance Survey DNF Data, with permission of The Controller of HMSO ©Crown Copyright.).

Fig. 6. Intersections resolved (Images derived using Ordnance Survey DNF Data, with permission of The Controller of HMSO ©Crown Copyright.).
middle of its two neighbours. This smoothes the variation in intervals along the embankment. It does not make the intervals constant however; this would take us back to problems resolved earlier by the algorithm!

8. **Solve the proximity conflicts by shortening hachures**: the last stage consists of resolving congestion that may occur on the concave side of any bend of the embankment. This is done by shortening some hachures until the minimum distance between consecutive hachures reaches a threshold value that ensures visual separation.

### 4. The algorithm

This section describes how the requirements defined in Section 3 were implemented.

#### 4.1. Formatting the input data

The data provided by the Ordnance Survey consist of a set of linear objects, attributed as either: “Top of Artificial Slope” or “Bottom of Artificial Slope”. This stage of the algorithm takes as input one line (top or bottom) and identifies the embankment boundary to which it belongs. The embankment boundary has to be made of a single top line and a single bottom line. During this stage, malformed data are detected.

**Stages:**

- Isolate all the lines that are Top or Bottom of Artificial Slope from other objects; call this set of lines `initial_lines`.
- By using an iterative process, build the subset of `initial_lines` that contains all the lines directly or indirectly connected to the input line.
- Split this subset into two groups: (either top or bottom).
- Three situations may then arise:
  - Both sets contain a chain of lines, and the two chains are connected at their extremities—this is a well-formed linear embankment (the example in Fig. 7 shows one made of three top lines and two bottom lines).
  - One of the sets contains a closed chain of lines called a ‘ring’, and the other set is empty—this defines a circular embankment. The method for detecting a well formed circular embankment is defined below.
  - In any other case, the data are malformed and are highlighted as such.

![Fig. 7. A well formed linear embankment.](image-url)
To complete the detection of a circular embankment, we use a recursive function that checks inside the input ring to see if there is a consistent hierarchy of enclosed rings. A consistent hierarchy is found when each ring is made of a single type (top XOR bottom), and from the bottom of the hierarchy, it is possible to pair up rings of opposite types. Recursive_function:

input parameters: `input_ring`, `input_type` (top or bottom)
output parameters: `output_ring`, `output_type` (top, bottom, none, or failed)

check if there are any top or bottom lines inside `input_ring`. If not, return with `output_type = none`

check if among the enclosed lines, there exists a well-formed ring (all the adjoining lines are of a single type) that contains all the other enclosed lines. If not return with `output_type = failed`. If so, this ring is the `output_ring`, and the `output_type` is set to its type (top or bottom).

call the recursive function (`output_ring`, `output_type`, `output_ring2`, `output_type2`)

if `output_type2 = failed`, then return with `output_type = failed` (propagate the failure)

if `output_type2 = none`, then return with the current parameter values

if `output_type2` is the opposite type of `output_type`, then return with `output_type = none`, because a well formed pair of rings has been found.

if `output_type2` is the same as `output_type`, then return with `output_type = failed`, because an inconsistency has been detected.

Upon completion of this phase, one of four cases can arise:

- `output_type = failed`—the algorithm has detected a well formed circular embankment.
- `output_type = none`—there is no inconsistency, but either there is nothing inside the ring, or there are pairs of rings that form complete embankments. This means that the `input_ring` is either isolated (malformed embankment) or it is the inside ring of an embankment.
- `output_type = input_type`, there is an inconsistency, we have not found a well formed circular embankment.
- `output_type` is the opposite of `input_type`, the algorithm has detected a well-formed circular embankment, of which the boundaries are made of an `input_ring` and an `output_ring`.

Fig. 8 shows two well-formed embankments, one made of lines 1 and 2, and the other made of lines 3 and 4. Passing line 1 to the recursive function would return line 2. Passing line 3 would return line 4, and passing line 2 or 4 would return ‘none’.

- Once a well-formed embankment has been detected (a pair of lines), these are passed as input to the hachuring algorithm.

Fig. 9 shows different cases where the system would detect a malformed embankment. Cases where the top or bottom lines do not form a single chain are rejected. In addition, for linear embankments, each extremity of the top line has to match with an extremity on the bottom line.
4.2. Calculate parameter values:

Empirical observations revealed that hachure spacing shortened where distance between top and bottom of slope narrowed and vice versa. In the current version of the prototype, we have given a constant value to this parameter for each case, based on the average distance between the top and bottom lines of the embankment. This average distance is computed by measuring the distance between top and bottom slope at twenty regular points and computing their average length.

All three parameters are set according to a linear function based on mean distance between top and bottom slope. Each equation has been based on empirical observations made on the Isle of Purbeck dataset (1:2500): one embankment with an average distance of 40 m, and one with an average distance of 2.5 m. The three equations take the following form:

\[
\text{step} = \frac{3.25 \times \text{average}_{\text{dist}} + 57.5}{37.5}
\]
\[
\text{base\_length} = \frac{0.75 \times \text{average\_dist} + 26.25}{37.5}
\]

\[
\text{minimum\_spacing} = \frac{\text{average\_dist} + 6.875}{37.5}
\]

These parameters are only valid for drawing hachures at 1:2500 scale, which is the scale of the map from which the measures have been taken. Further empirical analysis of 1:10 000 scale maps should enable us to derive a similar set of equations with different settings.

4.3. Creating the initial hachures

This stage contains three steps. First, a set of regularly spaced points are marked on the top line. Next, the relative position of the bottom line is determined (to the right or left of the top line). At each marked point, a hachure is generated. The orientation of the hachure at one point \( A \) is computed by taking the previous and next marked points \( A' \) and \( A'' \), finding the centre \( C \) of the circle going through these three points \( (A, A', A'') \), then drawing a line, \( AC \), from \( A \) through the centre of the circle, \( C \), as far as the bottom line boundary (Fig. 10).

When \( AA' \) and \( AA'' \) are parallel, the orientation of the hachure is perpendicular to \( AA' \). In some rare cases, this method does not provide a valid orientation. This happens for a triangle \( (A, A', A'') \), where one of the angles of the apex \( A' \) or \( A'' \) is obtuse (Fig. 11). In such a case, a hachure is not generated, but left blank, to be ‘filled’ by a later stage of the algorithm.

Once the line is drawn from the top to the bottom, a validation check is made to ensure that the hachure is reasonably perpendicular to the bottom line. If not, the hachure is removed, for subsequent ‘filling’. This validation check is achieved by inspecting the previous and next point on the bottom line, computing the centre of the triangle, and comparing the angle variation between the line going through the bottom of the hachure, its centre, and the hachure itself. If the angle variation is less than \( \pi/6 \), the hachure is considered valid.

![Fig. 10. Orientation of a hachure.](image-url)
Special cases occur at the start and end of linear embankments. Typically the first hachure has the same orientation as the first edge on the bottom line, and the last one has the same orientation as the last edge on the bottom line. In most cases, this works well, but can produce odd results depending on the point at which the bottom meets the top line. Fig. 12 shows initial hachures generated for three embankments (2 linear and one circular). Note that on the linear embankments, some gaps have been left, corresponding to hachures that were not accepted at the validation stage.

4.4. Solving intersections:

The procedure for solving hachure intersections starts by detecting and storing the initial intersections, then an iterative process solves them one by one. Each iteration contains three steps: selecting the intersection to solve, solving the intersection, and updating the list of intersections.

Detecting and storing initial intersections: for each hachure, test if it intersects with the following hachure. Store the intersection as a pair of hachures, indexed by the
distance between the intersection point and the closest anchor of the two hachures on the top line.

*Selecting the next intersection to be solved*: use the smallest value of the index to determine the pair of intersecting hachures to solve first.

*Solving an intersection*: replace the two intersecting hachures by their weighted average. Each hachure is weighted by the number of hachures it initially intersects with. Fig. 13 shows an example of intersection solving. The number by each hachure is their associated ‘weight’ based on the intersection count.

*Updating the list of intersections*: the pair of intersecting hachures that have been resolved has to be deleted from the intersection list. In addition, if the new hachure intersects with one of its neighbours, this has to be added to the intersection list.

4.5. *Smoothing the orientation*

The fifth phase is to smooth the angle changes among the hachures. Some smoothing mechanisms are required to ensure that the curves on the slope line are represented by a gradual variation in the orientation of the hachures, and not by sudden variation that could be interpreted by a map reader as a rupture in the shape of the slope. This smoothing is done by identifying points of inflection, and then ‘smoothing’ the angle of hachures from one inflexion point to another. The first step is to identify those inflexion points, then an iterative process identifies the “less smooth” hachure and smoothes it.

Finding inflexion points:

We detect inflexion points by comparing each hachure’s orientation with the preceding and following ones. If it is not somewhere in between these two orientations, then the middle hachure is flagged as an inflexion point. It is not desirable to keep too many inflexion points, since this would reduce the effect of smoothing. We use a filtering method that cancels any inflexion point which would have a similar orientation as the surrounding hachures.

![Fig. 13. Sequence of steps in the removal and replacement of intersecting hachures.](image)
Smoothing:
This is an iterative process that compares the orientation of each hachure that is not at an inflexion point, with an average value depending on the position and orientation of the previous and following hachures. The average is actually a weighted average using the curvilinear positions of the hachures on the top line. The hachure that has a value furthest from the average position is chosen and smoothed. This is done by keeping its anchor position on the top line, and updating its orientation according to the average value. The process stops when the less smoothed hachure is close enough to its ideal orientation. Creating a very smooth result is computationally intensive, so the algorithm attempts to create a smooth transition up to a limit of 500 iterations. Only in rare cases is this cut off mechanism required.

Figs. 14 and 15 illustrate the effects of smoothing. Inflexion points are shown in black. In addition to this visual improvement, it also provides a basis for the next phase, which is the insertion of additional hachures.

4.6. Inserting hachures in the gaps

In the final phase, gaps between consecutive hachures along the top and bottom lines are identified. The size of the gap dictates how many hachures need to be added. For each line, a 'gap' is defined as a curvilinear distance between two consecutive hachures greater than $1.5 \times \text{step}$. The number of hachures to be inserted is the closest integer to the ratio between this curvilinear distance and \text{step}. The insertion process uses one of two methods, based on the variation of orientation between the two hachures delimiting the gap:

If the insertion between hachures has an orientation variation less than $\pi/4$:
In this case, the co-ordinates of the anchors of the hachures delimiting the gap are used to define the average positions for the additional hachures. From the

![Fig. 14. Hachures before smoothing.](image1)

![Fig. 15. Hachures after smoothing (dark lines are those lines identified as inflexion points).](image2)
two top anchors a line is drawn and segmented at regular intervals depending on the number of hachures that can be inserted. The same process is applied to the bottom anchors, and lines connecting through these pairs of points are drawn. The lines are extended to meet the top and bottom slope. Fig. 16 shows a gap in which two hachures have been inserted using this ‘Cartesian’ method. This method should not be used when the delimiting hachures have very different orientations, as the orientation of the lines used to support the additional hachures becomes less and less reliable (as is illustrated in Fig. 17).

If the insertion between hachures has an orientation variation greater than \( \pi/4 \):

The curvilinear co-ordinates of the anchors are used to determine the average positions where the orientation variation is greater than \( \pi/4 \). In this case the positions are marked along the curvilinear boundary between the two hachures. After doing the same on the bottom line, the hachures are connected as in the previous example (Fig. 18).

The Cartesian method is better than the curvilinear one when the difference of angles is small. This is because it is not sensitive to the detailing in the top or bottom lines, thus ensuring that the variation of orientation of the added hachures is smooth. In the curvilinear case, any slight variation as a result of detail is not visually discernible.

4.7. Smoothing the interval distance

At this stage all the necessary hachures are present, but the visual impression of the whole set may be spoilt by localised changes of density among the hachures. This
can arise when the regularity of the intervals between the hachures in the bottom line have not been adequately controlled, or where gap filling is a little uneven. An iterative approach is used to smooth the interval spacing; it passes through the ordered set of hachures, and each hachure is eliminated and reinserted between its preceding and following hachure. This insertion/reinsertion process uses one of the methods described in the previous section (Cartesian or curvilinear method). Through empirical observations, it was found that four iterations were sufficient to produce the best results.

4.8. Solving proximity conflicts by shortening hachures

By this stage we have achieved an aesthetically pleasing result. In some cases, a final problem needs to be addressed: local overcrowding of hachures at moderately sharp bends on the top or bottom line. The shortening comprises two different processes: shortening hachures from the bottom of the slope, and shortening hachures from the top of the slope. The principle is the same for each and consists of shortening a hachure until the distance to its neighbours is sufficient to be discernible. The one difference is that on the top of slope hachures, we have to account for the triangular shape, while on the bottom side it is simply a line shape.

In both cases, the first step is to determine which hachures need shortening. For each hachure in turn, the adjacent hachure is found that has sufficient separation. All the hachures in between these two have to be shortened. Shortening occurs in two stages:

1. shorten each hachure until it is a sufficient distance from adjacent hachures;
2. find among the shortened hachures the one that has been shortened the least. Use this hachure to split the initial interval and do the same process recursively on both subintervals.

Fig. 19 illustrates the shortening principle. Fig. 19a shows the hachures before shortening. Fig. 19b shows the result of shortening the hachures between 1 and 5. Hachure ‘3’ is the closest from the slope line (among the shortened hachures), so Fig. 19c shows the result of shortening hachures inside interval 1–3 (2 has been shortened again in order to be an acceptable distance from ‘3’, and inside interval 3–5 (where nothing has changed).
Formula for shortening the hachures:

Fig. 20 illustrates how the revised length of the hachure is computed. If AB is the hachure we want to retain, CD is the one to be shortened into CD’ and ‘d’ is the minimum distance that we allow between D and the hachure AB; we need to determine a point I—the intersection of the lines supporting the two hachures. We also need to compute the angle \( \alpha \) between these two lines. From this we can deduce the distance ID’ from the trigonometric formula: ID’ = d / \( \sin(\alpha) \). Note that for shortening the top side of the hachure, the formula is similar, but as the hachures are represented by triangles, instead of taking the axial line of the hachures, we need to take the two sides of triangles that face each other.

5. Results

The following results have been generated automatically, with no interaction required from the user. The parameters have been automatically set for visualisation at a 1:2500 scale. Fig. 21 shows intermediate results of the hachuring process on three embankments, (two linear and one circular), through to the final result (Fig. 21f). Fig. 22 shows the hachures automatically produced for Corfe castle and the surrounding environ, which is a more extreme example. The result was considered to be good except in the shortening phase, which may generate ‘bench like’ feature where the hachures have been shortened. Fig. 23 shows a group of embankments, whose variation in width is more consistent. Here the algorithm performs very well (the original, hand drawn version is included for comparison).

6. Discussion and further work

Whilst hachuring might be viewed as an obsolete or outdated method of visualisation, in reality it is an important compliment to shape visualisation, and the various examples in this paper are testimony to the effectiveness of hachures in representing anthropogenic, and natural small scale morphology alike. Hachures give emphasis to relative gradient, direction of slope and shape distribution. As Imhof remarks (Imhof, 1982, p. 229): ‘Hachures alone, without contours...are capable of depicting
since the stroke direction indicates the line of maximum gradient everywhere and in every direction. In small-scale maps, where contours can no longer be used easily, the hachure is still a very good element for portrayal."

Further work is currently in progress to assess the scalability of the algorithm and its applicability to other scales of representation such as 1:10 000. In general greater reductions in scale require additional generalisation algorithms such as enlargement of slope regions and subsequent displacement. The algorithm has led to a much better understanding of how data should be captured in the field; it has revealed how different styles of defining top and bottom of slope can produce different results. It would be useful to generate the specifications for field data capture and help to ensure that the algorithm would work in all cases. At the moment, about 4% of the
embankments are malformed; in a few cases the extreme complexity of the form may lead to a substandard solution. Three specific areas of further research have been identified:

1. The shortening procedure could be modified to avoid creating 'bench like' features where none exist.
2. The smoothing of the intervals has a tendency to attenuate the variation of angles of the hachures that occur inside a bend of the embankment, and marginally propagate it outside of the bend. This could be improved by identifying the extent of the bends and placing some orientation constraints on the hachures.

3. Though a unique setting of parameters exists for each slope, the spacing of hachures within the region does not vary. Ideally the spacing should vary within the slope depending on the local variation in slope width.

7. Conclusion

The emphasis of this research has been on accommodating the aesthetic nature of the hand drawn hachure whilst minimising the amount of human computer interaction required. The work has built on the work of O’Loughlin & Mackaness (1999), but extends beyond it in significant ways. Specifically in the implementation of methods that use recursive techniques for hachure removal, automated methods of re-populating, smoothing of hachure orientation between automatically detected inflexion points, and shortening of hachures to avoid cramming. Most importantly, a method has been implemented that automatically sets various parameters according to the extent of slope and the intended scale of depiction.

This and other research (Hurni et al., 1999) clearly illustrate the feasibility of utilising traditional qualitative techniques as a complement to other more quantitative techniques. Many Ordnance Survey maps combine quantitative and qualitative methods. The automation of the hachuring process and the creation of aesthetically pleasing solutions means that this technique can now be retained in the realm of digital cartography. Prior to this research, it was questionable as to whether hachuring was going to be retained in OS LandLine and Superplan products because of cost. The implementation of such an algorithm opens exciting opportunities for inclusion of hachured regions in Internet delivery of map products, and enables the highly efficient storage of slope information in the database—simply recording the top and bottom of slope for each slope region, rather than each individual hachure.

While it is important to realise what qualities are lost through automation in cartography (Nyerges, 1991) it is equally important to realise what is gained, i.e. an efficient, consistent and comprehensible way to symbolise monuments digitally. Hand hachured solutions to individual slope regions are typically measured in order of days. This is a solution measured in seconds, that has produced results that are as good as the hand, and in some cases, deemed to have surpassed it.

Acknowledgements

The authors are very grateful for funding of this project from the Ordnance Survey. The authors are also very grateful for the involvement of The Royal Commission on the Ancient and Historical Monuments of Scotland (RCHAMS), in particular Jack Stevenson, Strat Halliday from the Archaeology Division, Alan Leith, Ian Parker,
Georgina Brown from Survey and Graphics and Diana Murray, and Peter McKeague from the National Monuments Record of Scotland. Their help in evaluation, and in comments on this paper is gratefully appreciated.

References


