ACCESSING OUTLINE SHAPE INFORMATION EFFICIENTLY WITHIN A LARGE DATABASE

Peter L. Main
British Museum Research Laboratory

Abstract: A general framework is suggested within which a computer database of the outline shapes of archaeological artefacts could be set up to permit on-line retrieval of those artefacts in the database that are similar in shape to some given artefact. In particular, it is shown how techniques of design already in use in certain types of text database can usefully be employed.

Keywords: COMPUTER,DATABASE,INFORMATION RETRIEVAL, GRAPHICS,ARTEFACT,SHAPE,ARCHAEOLOGY

1. Introduction

Much effort has been expended in designing database(DB) systems for fast on-line access to text information. The main difficulties arise where the DBs are large, and a fast response to enquiries is required. Much less interest has been shown in designing such DBs where graphical information is involved. The applications for such systems are of course limited, but do occur, notably in physical anthropology, botany and archaeology. Such a system is of particular potential in archaeology, especially in a museum context, for the retrieval of those artefacts which are in some sense 'similar' in shape to a given one. It should be stressed that the problem is not necessarily that of finding an artefact that is already known to exist somewhere within a DB but rather of retrieving other artefacts similar in shape. The problems of designing a DB to permit this sort of retrieval are considerable and this may explain why such systems have as yet been little explored. The principal difficulty lies in defining 'shape similarity' in an archaeologically useful way, in other words in developing similarity measures that do more than simple template matching. Work already done on the quantitative treatment of artefact shape has been almost entirely concerned with numerical typology as an aid to understanding the relationships between groups of artefacts, rather than as a means of retrieving information. The two problems have much in common of course since in both cases the definition of a measure of shape similarity is fundamental. Such measures have usually been based on agreement between a number of subjectively chosen measurements, ratios of measurements, and angles between key points on the artefacts. Only recently has the possiblility of comparing complete outlines been considered.

This paper examines some of the problems of accessing DBs containing graphical information in the form of complete outlines, in the light of those techniques already in use for more traditional text DB systems, and shows how many of the ideas can usefully be carried across. Fortunately the problems of designing such systems can be divided quite clearly into two parts:

i) defining measure(s) of similarity between any two artefacts which are appropriate for that class of artefact in that they reflect how an archaeological 'expert' might view them

ii) designing a system which uses these measures to retrieve information efficiently from a large DB.

This paper deals only with the second of these aspects, that is to say it assumes that the similarity measures in i) are already available. This is not to deny the difficulties of defining such measures - for some classes of artefacts this can be a serious problem - but it has been demonstrated that for some at least such measures can be defined (Main,1981). In order to illustrate the methods in ii) above this paper makes use of a particular storage format for outline shape, and related similarity measure (described in 3.1 below), but the methods are not for the most part dependent on these and would equally apply to other formats and measures.

2. Accessing information within large text databases

2.1 Types of system in use

On-line DB systems can be divided into two broad classes depending on the type of query they are designed to deal with. The most widely used type deals with specific, well-defined queries where the records required can be unambiguously identified by relatively simple selection conditions as is appropriate, for example, in an airline booking system. Such systems can potentially cope with very large amounts of information by using a variety of fast indexing techniques though only with substantial overheads in disc usage. The techniques used in designing such systems are fully described in e.g. Martin(1977).

The second type of system is designed to cope with the problems that arise when more sophisticated (and often less well-defined) queries need to be answered, as typically occur when searching the bibliographic DBs in use in some libraries. It is the features of the latter type of system that have the most relevance to accessing outline shape information efficiently, precisely because there is not necessarily any 'right answer' to such queries. Rather than determining whether a given record satisfies the query or not, one has to estimate how close (in some sense) the record is to satisfying it. The parallels with the process of searching for shapes 'similar' to a given one will be obvious. It is therefore only the second type of system that will be considered from here on. The design of such systems is discussed in e.g. Salton(1971).

2.2 Measuring 'relevance'

A crucial component of such a DB system is way of measuring relevance between a document within the DB and the initial query. At a simple level this measure might be the number of keywords in common between the document and the query. At a more sophisticated level this could take account of where words and phrases occur within the records, and the contexts in which they occur. Implicit in the use of a relevance measure is some threshold value above which a document is regarded as relevant. Depending on the searching algorithms employed, both the relevance measure and the threshold value may be allowed to vary during the progress of a search.

An important aspect of a relevance measure is that it is usually no more than an imperfect reflection of what is actually relevant to the user of the system. The correct keywords may not have been identified within the records of the DB; the measure of relevance may be inappropriate for the type of query; the user may have failed to define his query precisely enough. The degree of imperfection, moreover, will vary from one user to another, depending on his own idea of what is actually relevant to a particular query.

2.3 Structuring the database

As well as the problem of measuring relevance, there is the (different) problem of using that measure to retrieve information. A naive procedure would be to simply read through the entire DB in sequence and check the relevance of each record against the query. The problem is, of course, that for a DB of any magnitude this would give an impossibly slow response for an on-line system. In practice it is necessary to structure the DB in such a way that 'similar' documents are linked together, and that for a given query only part of the DB is actually searched. This usually means that some proportion of relevant documents will not in fact be retrieved, but this is accepted as a result of the compromise necessary in order to obtain an acceptable response time to requests.

The sorting of items into groups is the concern of cluster analysis - not in fact any single technique but a variety of methods. Many of these were designed for numerical typology, but some varieties more specifically designed for information retrieval are described in Salton(1971). The following is a simple example of how such a clustering technique might be employed in a bibliographic retrieval system:

i) Decide on a measure of similarity between any two documents e.g. by counting keywords or phrases held in common.

ii) Decide on a way of defining the 'centroid' of a given group of documents. In DB terminology the centroid is analogous to the mean of a group of numbers and represents some (hypothetical) document which can be regarded as representative of the whole group, so that a single comparison can then be made with the centroid rather than a number of comparisons with individual members of the group. Some degradation of the overall accuracy of the comparison is accepted, and is the reason why speed of search can be enhanced at the expense of accuracy.

iii) Choose a sorting algorithm (clustering method) and use it to divide up the document collection into groups in such a way that similar documents are in the same group and dissimilar ones are in different ones. This is equivalent to requiring that the group centroids should be well separated.

iv) Repeat step iii) as often as required, at each stage grouping the centroids from the previous stage. This leads to progressively larger groups until finally all documents come together into a single group (See Fig 1).

It should be stressed that this is only one possible way of structuring a DB. Although the diagram in Fig. 1 represents a hierarchical structure (i.e. where each step from bottom to top represents a partition of the current groups into disjoint subgroups), algorithms are available for generating overlapping clusters (sometimes known as 'clumping' algorithms), and lead to 'network' structures rather than trees. Such algorithms are not commonly used in numerical typology, where it is often assumed that typologies are inherently hiearchical in nature, but are more widely used in constructing DBs where a freer structure seems more appropriate.

If we are sorting a large DB it can be imagined that generating the

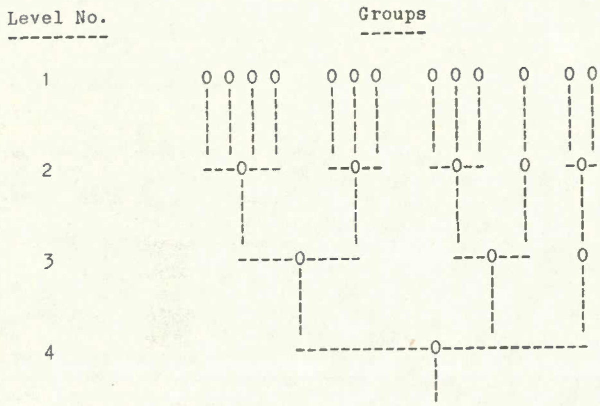

Fig.1. Schematic representation of the result of clustering items repeatedly to form a tree structure. The nodes at level 1 are the original items. The lower level nodes are group centroids.

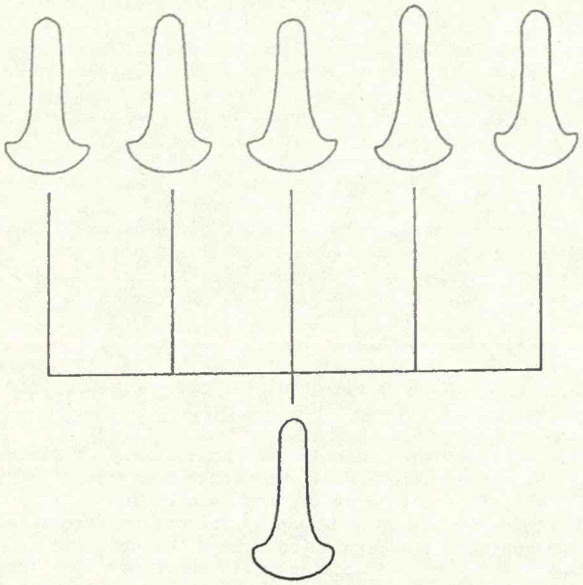

Fig.2. A group of five axes plotted from their STPs, and the group centroid.

complete tree structure for it may take a long time. In general, however, this is relatively unimportant (provided that it is not necessary to regenerate the structure too often as a result of amendments to the DB) since the primary considerations are <u>speed</u> and <u>accuracy</u> of search, and any time spent in generating the DB that enhances the subsequent searching of it will be worthwhile. Operations such as re-sorting the DB can be carried out when the system is not in use, for example overnight.

2.4 Searching the database

Having generated a tree structure for the DB, what algorithm should be used to search it - in other words what process will be initiated when the user enters his query? The simplest procedure is to calculate the relevance between the query and each centroid at the second bottom level of the tree (level 3 of Fig.1), move up that branch of the tree having the closest centroid, and so on up the tree until reaching a centroid whose relevance to the query is less than some minimum value. All documents within the group represented by that centroid would then be regarded as having 'satisfied' the query. A more thorough strategy would be to follow at each level <u>all</u> branches whose centroids lie close enough to the query.

Whatever strategy is employed, and however the tree was originally generated, the overall effectiveness of the system can be quantified by a number of parameters which are commonly used for this purpose. The two most important of these are known as <u>recall</u> and <u>precision</u>, and are defined, for a particular query, as follows:

$$\text{recall} = \frac{\text{no. of relevant documents retrieved}}{\text{total no. of relevant documents in system}}$$

$$\text{precision} = \frac{\text{no. of relevant documents retrieved}}{\text{total no. retrieved}}$$

For an ideal system both recall and precision would be 1.0, but in practice both will usually be less than 1.0. Of course the values of these parameters will vary from one query to another, so that some sort of averaging over a number of typical queries is necessary to get an overall idea of the system's performance.

2.5 User interaction

DB systems of the type outlined above have potentially much more to gain from user interaction than do those of the more straightforward airline booking variety. By 'user interaction' we mean building in to the search strategy mechanisms by which the user can examine and control the progress of the search. A degree of feedback from the user may well lead to a result that is more satisfactory to him than would otherwise be possible using a completely automatic system. As will be discussed in the following section, this idea is particularly appealing in the context of shape retrieval.

3. Accessing information within large graphical databases

3.1 The storage format

In the light of the ideas of Section 2 we can now turn to the problems of storing, and searching for, two-dimensional outline shapes within a DB. For the purpose of illustrating how methods similar to those already described can be employed, we will make use of some outlines of bronze axes from the Early Bronze Age of southern Britain. This dataset, and the procedures used to record the outlines, have previously been described in Leese and Main(1983). To summarise, the procedure was to digitise a sequence of coordinates from around the edge of each axe, to pass a smooth curve through these points, and to convert the resulting curve to tangent profile(TP) form. This involves a mathematical transformation of the curve to a function of tangent angle against arclength, the latter being measured from some standard point on each outline, in this case the midpoint of the cutting-edge of the axe. The TP is scaled to have total arclength equal to 1.0, so that the effect of size is removed when the shapes are subsequently compared. The type of curve-fitting used to connect the digitised points (bicircular arc fitting) is such that the resulting tangent profiles are piecewise linear. No information is lost by converting from two-dimensional cartesian form to TP form, so that provided enough points are digitised from the original drawing an outline visually indistinguishable from it can be regenerated from the TP. The procedures for curve-fitting, conversion to TP form and re-generation of the cartesian outline are discussed fully in Main(1981).

For reasons discussed below, the TPs are subject to one further modification before being stored finally as records on disc. This involves sampling each TP at equal intervals of arclength along it, resulting in what we will call a sampled tangent profile(STP). The sampling interval is the same for all items in the DB. Since TPs are piecewise linear, the sampling procedure involves only simple linear interpolation between the breakpoints.

While a TP is an exact representation of the curve-fitted outline, the corresponding STP is not. In other words, if a two-dimensional cartesian outline is generated from the STP it will only be an approximation to the curve-fitted outline. However, the smaller the sampling interval used to generate the STP the better this approximation will be, although only at the expense of the increased size of each STP disc record. In order to lose as little precision as possible for a given number of sampled points, a TP should not in fact be sampled uniformly but rather in proportion to its slope, or, equivalently, to the curvature of the original outline. The mechanisms to be described for structuring the database, however, require that any such sampling scheme is the same for all STPs in the DB, and would therefore need to be based on some form of mean curvature variation taken over the whole dataset. To simplify the following discusssion we will assume that all TPs have been sampled uniformly.

For the purpose of fast comparison and retrieval, STPs have the following advantages over TPs as a storage format:

i) Whereas in a TP record it is necessary to store a pair of values (arclength and tangent angle) for each breakpoint, STPs sample the TPs at constant intervals of arclength and so it is only necessary to store tangent angle values in the STP records.

ii) Since the sampling interval is defined to be the same for all records, each STP record has the same length and the dataset can therefore be conveniently stored as a fixed record-length random access file. TP records on the other hand have variable record length and random access is more difficult.

iii) As the similarity measure to be described below involves comparing values of tangent angle at corresponding values of arclength on the two outlines, this can be performed on STPs easily, by reading the tangent angle values into main memory and using fast vector arithmetic routines to compare them. Similar vector operations can also be used to calculate the centroids of groups of STPs.

3.2 Measuring similarity

In the same way that 'relevance' measures can be used to retrieve text information, we can define 'similarity' measures that fulfil the same function for outline shape. Just as relevance measures used in a text DB system will not precisely reflect what a particular user sees as relevant, so shape similarity measures can only approximate to a user's visual perception of differences in shape. The difficulties of imitating the complexities of human shape perception by analytical methods are well established.

The measure we describe here is in fact a dissimilarity (or distance) measure and is defined as follows. The distance $D(T_1,T_2)$ between two tangent profiles T_1 and T_2 is defined as:

$$D(T_1,T_2) = \int_0^1 |T_1(s)-T_2(s)|\,ds$$

This distance measure is defined to operate on TPs, but we require an equivalent form for STPs. The function

$$D'(T_1,T_2) = \frac{1}{S}\sum_{i=1}^{S} |T_{i1}-T_{i2}| \;-\; \frac{1}{2S}\{|T_{11}-T_{12}|+|T_{S1}-T_{S2}|\}$$

where T_{ij} is the ith sampled value of tangent angle from T_j, and S is the number of points sampled from each TP, approximates well to $D(T_1,T_2)$ for large S. Once the two STPs to be compared have been read into main memory, therefore, $D'(T_1,T_2)$ can be calculated quite efficiently using one vector subtraction and one vector modulus sum. This particular distance measure is independent of scale (since all TPs were originally standardised to have total arclength equal to 1.0) but is not independent of the orientations of the shapes being compared, and is thus only appropriate where the orientation of the drawings has been standardised before conversion to TP form. Orientation-independent measures, and those with a variety of other properties, can be defined but tend to be less efficient to compute. Examples are discussed in Main(1981).

3.3 Structuring the database

We will now outline a method by which the equivalent of the tree structure described in 2.3 might be generated for use with STPs. We assume that the entire DB (of N shapes, say) has been converted to STP form and stored as a random-access disc file of N records. The first requirement for generating the tree is a definition of what we mean by the 'centroid' of a group of STPs. We can define the centroid as the STP that results from averaging corresponding tangent angle values over the whole group, i.e.

$$T'_i = \sum_{j=1}^{J} T_{ij} \quad ,i=1,\ldots S$$

where S is the number of sampled points and J is the size of the group. As with distance calculations, this averaging process can be carried out efficiently by vector operations. (Although T' has not in fact been generated by sampling any particular TP, we refer to it as an STP since it is of precisely that form and can be treated by a clustering algorithm as if it were an STP.)

Armed with definitions of distance and centroid we are now in a position to choose a clustering algorithm to sort the original N STPs into progressively larger groups, forming a tree structure as in Fig.1. A wide variety of 'nearest centroid sorting' algorithms that rely only on definitions of distance and centroid, are available and are described and compared in Anderberg(1973).

Repeated applications of the chosen algorithm will give rise to a tree structure such as appears in Fig.1. The STPs of the group centroids, being of the same form as those of the original outlines, can be stored with them in the same random-access file, giving rise finally to a DB of

$$N + \sum_{i=2}^{L} G_i$$

records, where G_i is the number of groups in the ith level of the tree and L is the number of levels. The structure of the tree can be stored economically in a different file, using pointers to records in the STP file and making the order in which records are stored in the latter file unimportant.

A word of caution. Where N (the number of original outline shapes) is large, it is imperative that the number of distance calculations required to construct the tree is at most a function of N and not of any higher power of N. Even using fast vector arithmetic the time taken to compare two STPs is not insignificant, and the process of tree generation would otherwise quickly become unmanageable for quite moderate values of N (a few hundred, say). Although many nearest centroid sorting algorithms do satisfy this requirement even they usually require some function of ($G_i \times G_{i-1}$) distance calculations at the ith level of the tree, and the G_i themselves may be large near the top of the tree. This may require in practice that the average group size is kept large (i.e. G_i kept small) at least at the upper levels of the tree, by placing suitable constraints on the clustering algorithm.

3.4 Searching the database

Having structured an outline shape DB in the same form as has been described for text DBs, all of the possible search strategies available for the latter carry across directly to the new situation - we need only replace the relevance measure by a shape similarity measure. The query, which in the case of a text DB would typically take the form of combinations of exclusion and inclusion conditions placed on words and phrases, is replaced by a 'query shape' which we wish to compare with material in the DB. This, of course, requires that the query shape be digitised and converted to STP form before the DB is searched.

The effectiveness of a given DB structure and search strategy can be estimated using analogues of the precision and recall parameters in 2.4 obtained simply by replacing 'relevant' by 'similar' in the definitions. A particular shape, x say, from the DB would be regarded as 'similar' to the query shape q if, in the notation of 3.2,

$$D'(T_x, T_q) < K$$

for some threshold value K.

3.5 User interaction

In order to interact effectively with a graphics program of any type, a VDU capable of graphical display is almost a necessity, and the following remarks will assume that such equipment is available.

It was pointed out in 3.1 that the process of transforming a two-dimensional curve into tangent profile form is reversible. The recovery is precise in the case of a TP but less than precise in the case of an STP. The imprecision is most noticeable where the original curve was closed (as is usually the case with artefact outlines) and the beginning and end points fail to coincide when the curve is regenerated from its STP. A good display can nevertheless be achieved, albeit somewhat artificially, by initially calculating the extent of the mis-match and then redistributing the error uniformly around the curve as it is displayed. The outlines in Fig.2 have been generated in this way.

Using this procedure we can display on the VDU screen the artefact outlines stored in the DB, by transforming the STP records back to cartesian form, and this will often be the most effective way of informing the user of the result of a search. More far-reaching possibilties, however, exist. It can be shown (Main,1981) that any two-dimensional curve transforms to a unique TP <u>and vice versa</u>. Thus each <u>centroid</u> STP held within the DB can be <u>transformed</u> to a curve that is a form of 'average shape' of all the outlines in the group it represents, and an example appears in Fig.2. We therefore have a means of showing the user what is happening as a search progresses up the tree, by displaying the group centroids of those branches the search algorithm is following. To take the idea of interaction a stage further, one can envisage that the user might be allowed to <u>choose</u> which branch to follow on the basis of this display.

A number of points should be made about displaying group centroids in this way:

i) The average of the TPs of two closed curves does not, in general, transform back to a closed curve. This is a mathematical consequence of the way TPs are defined and is not related to conversion to STP form. Thus group centroids of closed curves will be subject to a mis-match which is over and above, and unrelated to, that caused by conversion to STP form. No extra action need be taken, however, provided that the centroids are displayed using the same error redistribution technique described above.

ii) When presented with a visual display of group centroids the user should be made aware that what he is looking at does not correspond to any real artefact but that the centroids represent only an intermediate stage of the retrieval process. The method seems nevertheless to be a useful tool for bringing out in immediate visual form, the essential characteristics that distinguish groups.

iii) The fact that group centroids can be displayed at all is a consequence of the fact that complete outlines are being stored and manipulated throughout. It is not easy to see how such a powerful visual impression of group characteristics could be provided in text DB systems, or even in outline shape systems where the information has been degraded to a series of measurements.

4. Concluding remarks

The foregoing discussion is intended to make some general points about how techniques already employed in the design of a certain class of text DB system can be carried across to the design of a graphics DB system for the storage and retrieval of complete outlines. To give some substance to the discussion we have shown how a particular storage format for outline shapes can be employed in this context. It is not intended to suggest that this shape representation, nor the distance measure described, are necessarily the best choices for any particular application - a variety of other methods could be used within the same general framework. The interested reader is referred to e.g. Duda and Hart(1973). However, the tangent profile format and associated distance measures have at least been quite thoroughly examined in the context of artefact shape and their strengths and weaknesses identified (Main, 1981). It will also be clear that this paper has not attempted to suggest answers to the more technical decisions that would need to be made when setting up a practical system, such as how to best choose the clustering algorithm, distance measures and threshold values, sampling interval for generating STPs, and tree search algorithms. Such questions can only really be answered with reference to constraints such as database size, available hardware, response time required and the complexity of the shape information with which the system is expected to deal.

References

Anderberg,M.R.(1973) Cluster analysis for applications Academic Press, New York.

Duda,R.O. and Hart,P.E.(1973) Pattern classification and scene analysis Wiley and Sons, New York.

Leese,M.N. and Main,P.L.(1983) 'An approach to the assessment of artefact dimensions as descriptors of shape' Proc. Conf. on Computer Applications and Quantitative Methods in Archaeology (ed. Haigh,J.) pp. 171-180, University of Bradford.

Main,P.L.(1981) A method for the computer storage and comparison of the outline shapes of archaeological artefacts, PhD thesis(CNAA), May 1981.

Martin,J.(1977) Computer data-base organization (2nd edition), Prentice-Hall, New York.

Salton,G.(ed.) (1971) The SMART retrieval system, Prentice-Hall, New York.