Putting the Geographical Analysis Machine on the Internet

Stan Openshaw, Ian Turton, James MacGill, John Davy*
School of Geography and School of Computer Studies*,
University of Leeds, Leeds LS2 9JT

1. Introduction

Currently most of the proprietary GIS software systems lack sophisticated geographical analysis technology. Attempts over the last decade to persuade the system developers to add spatial analysis functionality has so far failed to have much visible impact. The traditional arguments seemingly still apply; viz. no strong market demand, fear of statistical complexity, lack of suitable GIS-able methods, and a deficiency of statistical skills amongst the GIS system developers.

Various solutions to this dilemma have been suggested and tried out; in particular the development of spatial statistical add-ons tied to this or that GIS and the development of standalone statistical packages with either basic GIS functionality or easy linkage to one or more GIS systems. The problem is that these systems mainly serve research needs whereas most of the potential end-users of geographical analysis methods are not researchers in academia but involve the far larger numbers of global GIS workers. Typically these end-users want easy to use, easy to understand, easy to explain methods with all the complexity hidden away from their gaze. They are not necessarily statisticians and may not be trained in any spatial science. Yet these are the potential applications oriented end-users of geographical analysis tools. Somehow user friendly methods need to be developed so that spatial analysis is no harder to apply than any other part of the GIS tool-kit. Unfortunately the potential users have no good idea of what tools they need because they may not know what is available or what is possible. Additionally, there has been considerable

misinformation by the GIS vendors as to what is spatial analysis. The challenge is to foster a spatial analysis culture amongst the end users of GIS.

There are three further difficulties. First, there is not a single dominant GIS so any spatial analysis technology likely to appeal to end-users has to be what might be termed "GIS invariant". It has to be compatible with all of the world's current GIS systems otherwise the majority of users will never be able to use it. Second, most potential end-users may need to be convinced by data trials and tests that a geographical analysis tool is worth using on their data and in their unique operational-institutional-organisational setting. These tests have to be capable of being easily performed since this is probably the principal mechanism for the diffusion of spatial analysis tools. If potential users can obtain at no expense and little effort potentially useful results on their own data then maybe the case for adding spatial analysis functionality to the GIS tool-kit will be greatly strengthened. Finally, any successful methods have to be available in a platform independent form so it can be moved subsequently on to local user systems.

The paper focuses on the use of the World Wide Web as a means of creating a platform independent interface to a particular geographical analysis method as a test of viability. The aim was to take a generic and widely applicable point pattern analysis method (the Geographical Analysis Machine) that is currently little used because it is not available in a platform independent form and convert it into a WWW based tool so that any user with a web browser could use it. Such a system would be GIS invariant since data from any of the world's GIS systems could be sent to it, analysis performed, the results pre-viewed, and then imported back into any local GIS

for further scrutiny. The ideal system would use WWW forms and hypertext to make using it as easy and as automated as possible, whilst providing in depth support and information sufficient to resolve any user questions concerning the application. This strategy would also avoid (or postpone) the traditional problems of porting software onto different hardware and for many different operating systems, and the subsequent problems of software support and maintenance. If there is only one "geographical analysis machine" located on the internet then all these problems are avoided as far as the end user is concerned. Additionally, the physical location and hardware used for this machine could be located virtually anywhere in the world, it could be a parallel supercomputer or a workstation. The only remaining problem is how to support this virtual machine and whether the ultimate owners of the hardware are happy or willing to run a user service on it. The solution here may be to allow the program to be downloaded to a user's local host and to restrict the "free" service to evaluations concerned with testing, evaluating, and prototyping the technology. The principal downloading problem is that the system being downloaded has itself to be portable and platform/operating system independent (e.g. Java), or exist in multiple versions for specific target systems (six species of Unix, NT, windows, OS/2, Mac) or distributed as source code able to compile without difficulty anywhere (i.e. Fortran 77, C, C^{++}).

The paper evaluates these alternatives. Section 3 describes the development of a geographical analysis machine on the WWW. Section 4 describes the performance impact of converting the Fortran code to C and then re-casting it into an object oriented form for C^{++} and then ultimately into Java. Section 5 provides some examples of running the GAM on the WWW and section 6 outlines plans for future developments.

2. A Geographical Analysis Machine

2.1 Mark 1 Geographical Analysis Machine

One way of meeting the design objective for end-friendly geographical analysis tools is to develop totally automated geographical analysis methods. The notion of an analysis machine is very attractive. Put your data into a "geographical analysis machine", click on the "go" icon, and out come the results. Openshaw *et al* (1987) described the development of a prototype Mark 1 Geographical Analysis Machine (GAM/1) for the analysis of point data for evidence of patterns . This was an early attempt at automated exploratory spatial data analysis that was easy to understand. The GAM sought to answer a very simple practical question. Given some point referenced data for a region of interest relating to something of interest to the user **WHERE** might there be evidence of localised clustering if you have no a priori idea of where to look due to lack of knowledge about the data and the possible pattern generating processes? This is a generic exploratory geographical analysis task that is widely applicable to many different types of data; for example, rare disease data, crime data, burst pipes, and traffic accidents.

The GAM algorithm involves the following steps:

Step 1. Read in X,Y data for population at risk and a variable of interest from a GIS

Step 2 Identify the rectangle containing the data, identify starting circle radius (r), and degree of circle overlap

Step 3 Generate a grid covering this rectangular study region so that circles of radius r overlap by the desired amount

Step 4 For each grid-intersection generate a circle of radius r

Step 5 Retrieve two counts for this circle (the population at risk and the variable of interest)

Step 6 Apply some "significance" test procedure

Step 7 Keep the result if significant

Step 8 Repeat Steps 5 to 7 until all circles have been processed

Step 9 Increase circle radius and return to Step 3 else go to Step 10

Step 10 Create smoothed density surface of excess incidence for the significant circles using a kernel smoothing procedure with an Epanechnikov (1969) kernel (see Silverman, 1986) and a bandwidth set at the circle radius; this is used to build-up an aggregate surface of significant excess incidence based on all significant circles found in step 7

Step 11 Map this surface because it is the peaks that suggest where the accumulated evidence of clustering is likely to be greatest.

Note that the original GAM/1 consisted of Steps 1 to 9. Steps 10-11 are the GAM/K version that evolved in the late 1980s (see Openshaw et al., 1988, 1989; Openshaw and Craft, 1991). Many subsequent attempts to apply GAM are based on the GAM/1 rather than the more sophisticated GAM/K version. The kernel density surface is very important as a guide as to where to find localised pattern. The accumulation of evidence of many different scales of pattern analysis is a very important but simple mechanism for filtering out false positives and allows the user to focus their attention on the highest peaks. The choice of significance test depends on the rarity of the data

i.e. Poisson, binomial, Monte Carlo and bootstrap methods can be used. A fuller description is contained in Openshaw et al (1998).

The GAM was very computationally intensive hence the term "machine" because it really needed a dedicated big and fast computer to run it. The first runs on leukaemia data for Northern England took over 1 month of CPU time on a large mainframe computer of the 1980s. Later it was modified to run on various vector supercomputers such as the Cray X-MP. GAM's main claim to fame was its use in the discovery of a hitherto unknown cancer cluster on Tyneside.

2.2 GAM goods and bads

Ten years ago GAM/1 was a mixed blessing! It was praised by geographers as a potentially major development in useful spatial analysis technology. However, it was severely criticised by some statisticians, in part due to their ignorance of the geography of the problem and partly due to potential deficiencies in the statistical testing employed in GAM/1. Additionally, the software for GAM/1 was never distributed since it was not easily run and its dependency on a supercomputer severely restricted its usefulness. GAM/1 did have some good aspects. In particular, it was automated. Prior knowledge of the data which would invalidate hypothesis testing was rendered irrelevant because the method had no prior knowledge, unlike human investigators who once they view a map of the data are instantly in a post hoc hypothesis testing situation. GAM also looked for localised clusters at a time when most spatial statistical methods mainly concentrated on global (whole map) measures of pattern. Additionally, GAM was a search technique and its search for local clusters was geographically comprehensive and it treated every location in the same

way, whereas a deductive approach focused only on those locations where one or more a priori hypotheses were to be applied. GAM was thus unaffected both by prior knowledge of the data and ignorance of where to look for clusters, because it looked everywhere. GAM also incorporated a mechanism for handling data uncertainty arising from imprecision in the location of the point data. However, there were also some technical problems with GAM/1. In particular, there was the issue of multiple testing and various concerns about whether the statistical tests for pattern were appropriate. With time many of these criticisms faded.

2.3 Comparative evaluation of GAM

The next major development occurred in the late 1980s the International Agency for Research on Cancer (IARC) commissioning a study of several different clustering methods, many developed by the critics of the original prototype GAM. The idea was for eight different research groups to apply their preferred method to each of 50 synthetic cancer data sets. These data were specially created to simulate plausible rare cancer clustering patterns but with varying degrees of clustering. The locations of the clusters were known but kept secret from the participants who performed their analyses without any knowledge of the correct results. The results were finally published in Alexander and Boyle (1996), over five years late! It was anticipated that the statistical methods preferred by the critics of GAM would work best. However, perhaps to the surprise of Alexander and Boyle, GAM/K was shown to be the best or equivalent best means of both testing for the presence of clustering within a study region and for finding the locations of clusters; see also Openshaw (1996).

2.4 Reviving GAM/K

The IARC results were sufficiently encouraging to make it important to revive GAM/K. The code for the original GAM/K still runs on the later day version of the Cray X-MP vector supercomputer (the Cray J90). Initially, efforts were made to port the Cray X-MP code on to a Cray T3D parallel supercomputer with 512 processors. Unfortunately the original Cray code was heavily vectorized and unsuitable for a parallel machine. The algorithm had to be reprogrammed from scratch. Its performance now critically depended on how efficiently it performed the spatial data retrieval. The original GAM/1 used a recursive tree structure (a KD-B tree). In GAM/K for the Cray X-MP this had been replaced by a hash based two dimensional bucket search that was heavily vectorized. If nothing was done then the new GAM/K was estimated to require about 9 Days of CPU time on a single J90 processor (or Cray T3D node) to perform a single UK run using census data; although a single run could be completed in less than 1 hour on a 512 processor Cray T3D. However, this was clearly unsatisfactory if GAM was ever to become a widely applicable tool.

Fortunately, it was very easy to make GAM/K run much faster and, of course, modern workstations now deliver performance levels equivalent to a single Cray J90 processor. Subsequent modifications to the spatial data retrieval algorithm used in GAM/K reduced the 9 days to 714 seconds on a slow workstation. The strategy was simply to use knowledge of the nature of the spatial search used by GAM/K to minimise the amount of computation being performed. The effects of the code optimisation were dramatic. A test data set of 150,000 points (equivalent to a UK census dataset) generated a total computation load of 10,498 million megaflops of

arithmetic with the naive method but this was reduced to 46 million megaflops by the revised algorithm. GAM/K was now a practical tool from a computational point of view. It could now be easily linked to any GIS since it no longer needed a supercomputer to run it unless you are interested in the existence of possible multiple testing effects. This involves running the GAM analysis 500 or 1000 times so that Monte Carlo estimates can be made of multiple testing effects. It would seem that this is only worth performing for rare disease data where there are indications of weak levels of clustering.

3. A WWW implementation

The next stage is to consider using simple scripts, HTML forms and the common gateway interface (CGI) to develop an interface to the GAM. The idea was that the user merely had to send their data to GAM and the results would be sent back to them. The WWW can be used to provide a standard GUI with help, buttons, check lists etc. that can be run through any browser.

The first form the user encounters (figure 1) provides a series of boxes for the user to enter the names of their data files. These files are then uploaded to the CCG web server. The mechanics of this transfer are negotiated between the server and the user's browser.

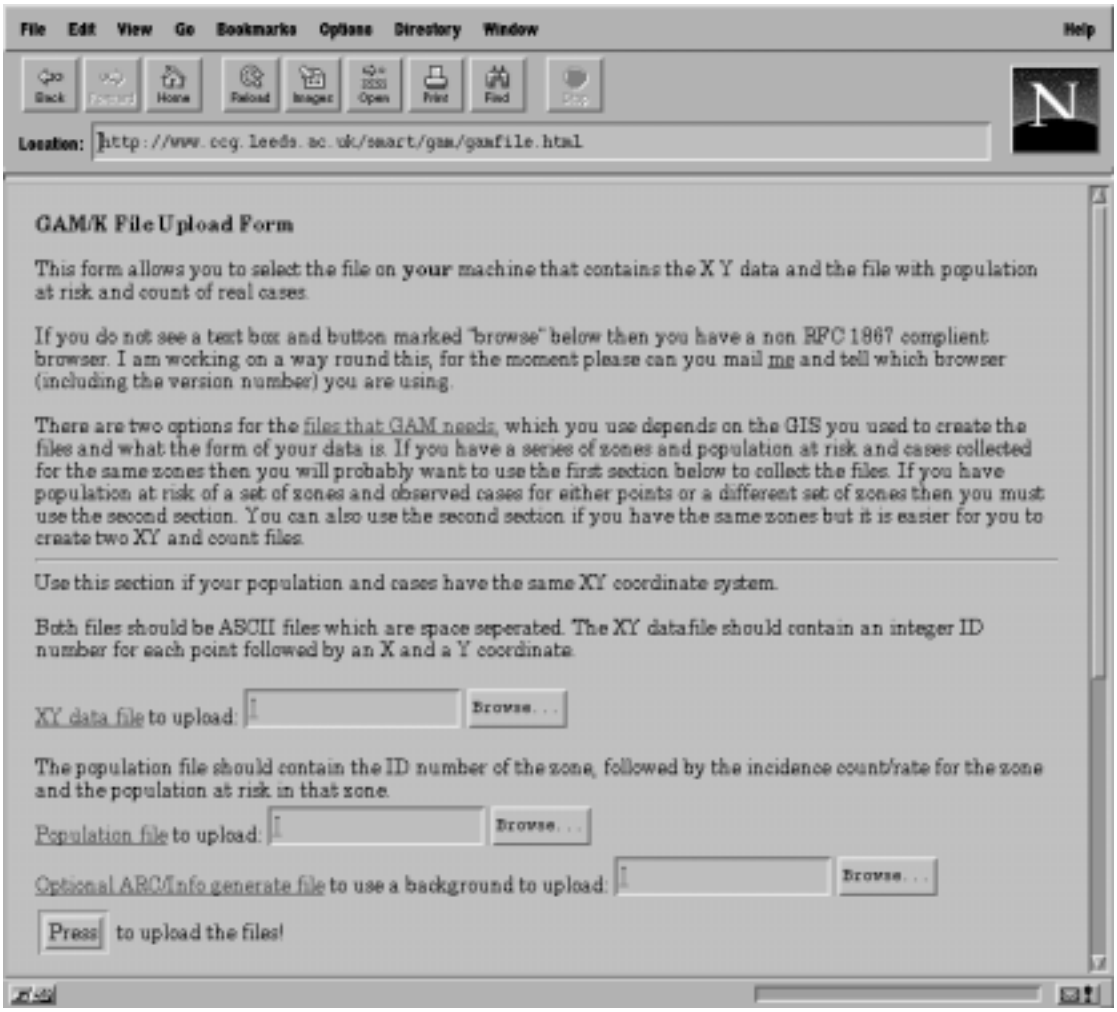

Figure 1: The file upload page

The user is then presented with an HTML form (figure 2) that allows them to specify the parameters of the run. The form uses a combination of type in boxes, pull down menus and check boxes. The browser makes use of the local interface so that the menus look like other menus of the user's graphical interface.

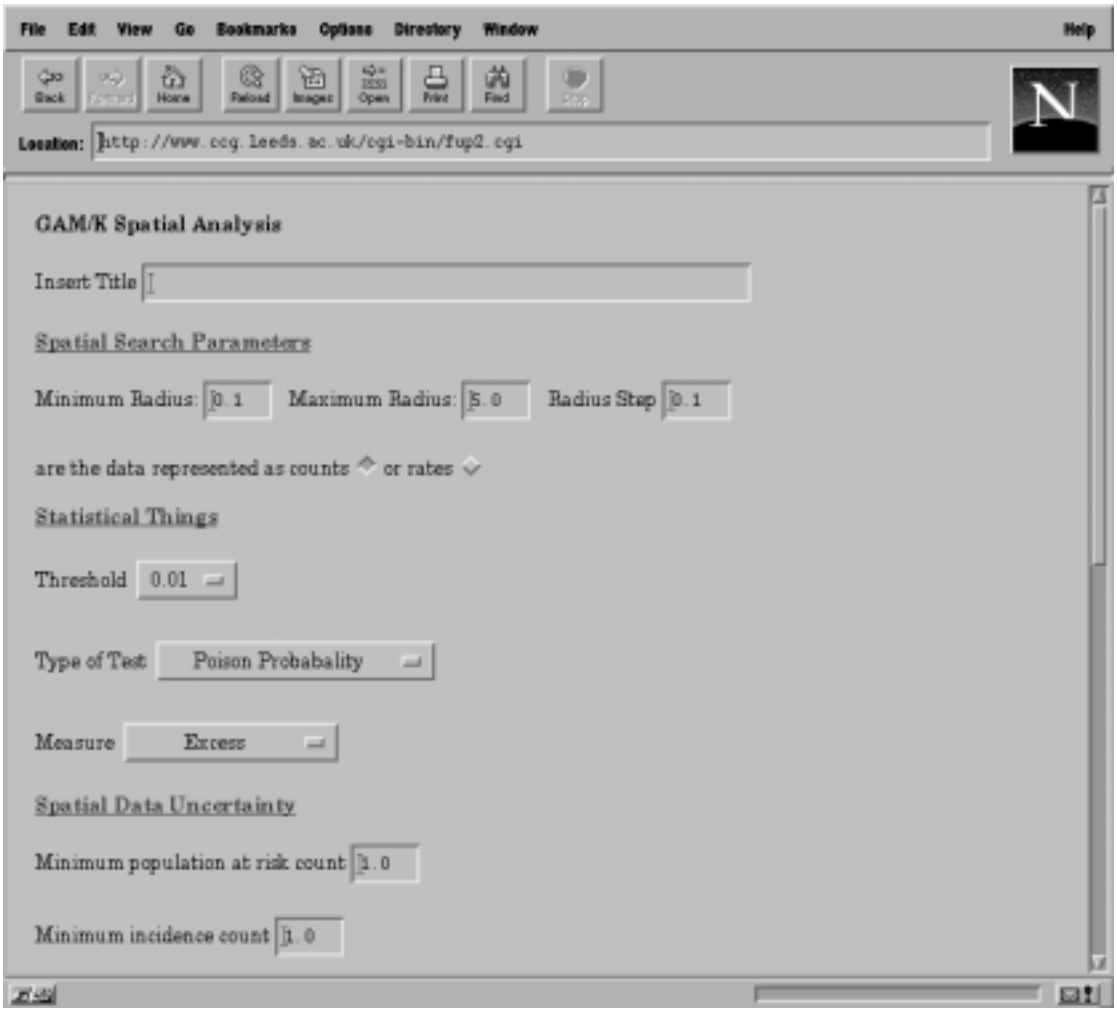

Figure 2: Parameters Page

The user then proceeds to run Gam on the CCG web server via a CGI Script. GAM produces its output as an Arc/Info ASCII grid file. The final CGI script converts this to a graphics interchange file (gif) that can be displayed in the user's browser, in exactly the same manner as a normal image (figure 3).

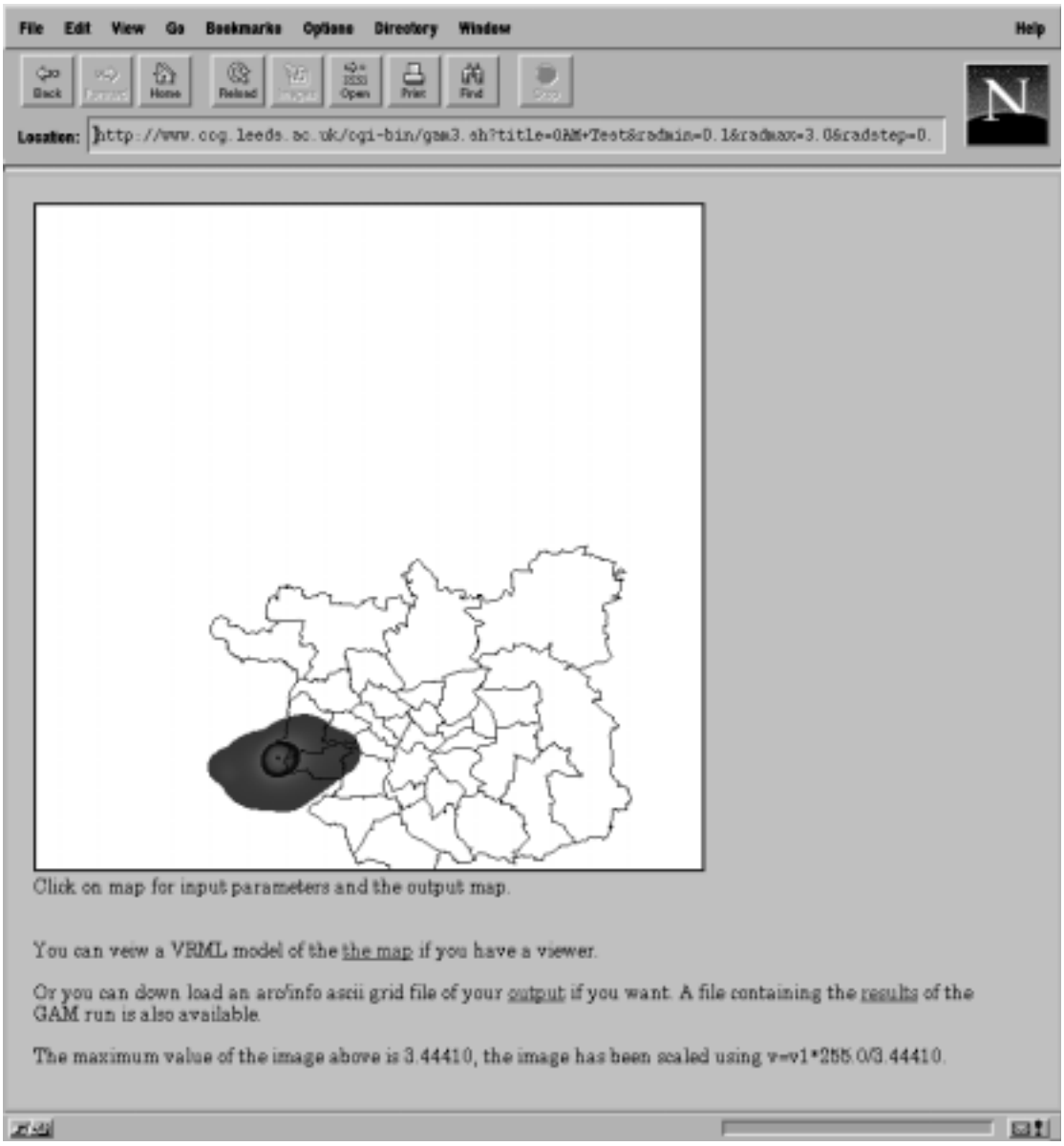

Figure 3: Output from a GAM run

Other options allow the user to add a simple vector layer (e.g. administrative boundaries, roads etc.) to the output map so that hotspots can be more easily located. The user can then down load the ASCII grid file for import in to Arc/Into or Arc View. Alternatively the grid file can be converted to the virtual reality modelling language (VRML) and viewed in "3-dimensions" to allow users who are unfamiliar

with cloropleth maps determine which is the highest peak. If you wish to try out GAM on your data then the URL is http: /www.ccg.leeds.ac.uk/smart/intro.htm.

4. From Fortran 77 to C^{++} to Java

GAM was originally developed in Fortran 77 and its performance has been substantially improved over several years, largely through highly optimised spatial search routines. Despite its great antiquity Fortran code has some advantages. The latest F90 compilers greatly extend the capabilities of Fortran 77 and narrow the gap between Fortran and C. However, if the aim is to squeeze maximum execution time performance out of a code then Fortran 77 code run on a F90 compiler is still best. The simple structure of Fortran allows highly efficient loop optimisation albeit at the expense of programmer productivity. The problem with C and C^{++} is the lack of optimising compilers that can cope with the nature of these codes, whereas Java is even worse because it is currently an interpreted language although just-in-time compilers and other tools may soon narrow the performance gap. However we are entering an era where programmer time is far more expensive than machine time. At present and certainly in the future the use of structured programming methods and modular code development techniques will become more important. Portability will become an overriding concern to even program speed.

The Fortran 77 to C conversion was very easy and could have been performed by automatic software but was used as a learning experience to understand the algorithms prior to object orientation. Current needs, however, called for a re-engineering in Java, on three counts. Firstly this enables distribution of a completely

portable and platform independent version to be freely available. Secondly, a modular object-oriented design facilitates GAM's future deployment in a range of client-server-based spatial analysis scenarios which may well become useful as GAM becomes embedded in more complex and smarter pattern search mechanisms able to explore higher dimensional data spaces. Finally there are plans to add animation to GAM and this is by far most easily done in Java.

The translation process proceeded in three stages. First the Fortran code was translated to a structurally equivalent version in C^{++}, using only a procedural subset of the language. This was a straightforward task which could well have been automated. Second, the program was re-engineered in C^{++} to an object-oriented form.

Four main classes were used. They are as follows.

(1) a *Database* class holds the (pre-processed) data with co-ordinates, incidences, and population at risk, making it available to arbitrary numbers of (read-only) search routine.

(2) a *Search* class encapsulates the main spatial search. It was not sensible to modularise at a finer grain since this would lose the benefits of the various optimisation employed.

(3) A *SignificantTest* class manages the application of statistical significance tests.

(4) A *DensitySurface* class manage the final output from the search as a surface (of user-controllable resolution), whose peaks indicate regions of potential further interest.

The final stage was a straightforward translation from C++ classes to Java. Clearly the re-engineering process gains flexibility at the cost (in the short term) of performance. However, because the re-engineering was at a fairly coarse grain it retained the high performance properties of the search algorithms originally developed in Fortran. As a result the loss of performance was likely to be far less than would have been the case if GAM had been object oriented at the circle level. Experiments were therefore carried out to assess this performance impact. Table 1 shows the execution times (in seconds) for different sizes of data set. Times include the input of the initial date from files and the output of the final density surface to a file. All results were obtained on a Sun Ultra 30 workstation. Standard Fortran and C++ compilers were used, with the highest level of optimisation. The Java version used Sun's JIT (just-in-time) compiler.

Table 1. Execution times for GAM in various languages on Sun Ultra 30 workstation

Programming Language	**9,681 pairs of X,Y**	**71,911 pairs of X,Y**	**370,397 pairs of X,Y**
Fortran	1.88	10.14	517.6
C++(procedural)	2.61	12.35	493.6
C++ (0-0)	2.90	15.13	537.6
Java (JIT)	8.24	27.77	921.5

Note: times based on mean of 10 runs

Given the maturity of Fortran optimising compilers, the performance loss in the other versions is unsurprising, and indeed compares quite well with what was anticipated. In particular, the performance loss with C++ of moving from a monolithic to an

object-oriented code structure is small but this reflects the coarse granularity of the object-oriented code. The performance of Java could be further improved through the use of a native compiler when they become available. However, in all cases the execution times are sufficiently small as to suggest that GAM is now a very practicable proposition and can be easily applied to many different problems. The (100% pure) Java version gives the basis for a portable down-loadable version of GAM. Ideally this would be in the form of an applet invoked from within a Java-enabled WWW browser. Unfortunately it is not quite that easy, security restrictions within browsers prevent applets reading or writing the local filestore. For this reason we make GAM available as a Java application in the form of a Jar archive file, which can be downloaded from the GAM web site. After manual extraction of the compiled class files can be run using a standard Java interpreter or JIT.

5. Case Study of GAM analysing crime data for a US city

The US Department of Justice Crime mapping and research centre created a crime data set which GAM has been used to analyse. The data was supplied at the US census street block scale of the outlying counties of a US city. There were two types of crime residential bulgaries and street crimes.

5.1 Residential crimes

The obvious population at risk for residential crime is the census population although this could be adjusted to reflect socio-economic covariates. This was not done here. Figure 4 shows the hot spots detected by GAM in the residential crime data. There are two large hot spots (labelled A and B in the figure) which is where attention should be concentrated. Hot spot B is the larger while hot spot A is strong, its location in a "peninsular" means that edge effects may be responsible for some of the excess, circles located near the edge of the zones may take in crimes located in the edge zones, but attempt to draw population from the missing areas of the map leading to false excesses. It may be interesting in the future to repeat the analysis with central area of the city included in the study area since this boundary is clearly not a low crime area as with the outer boundaries of the study area.

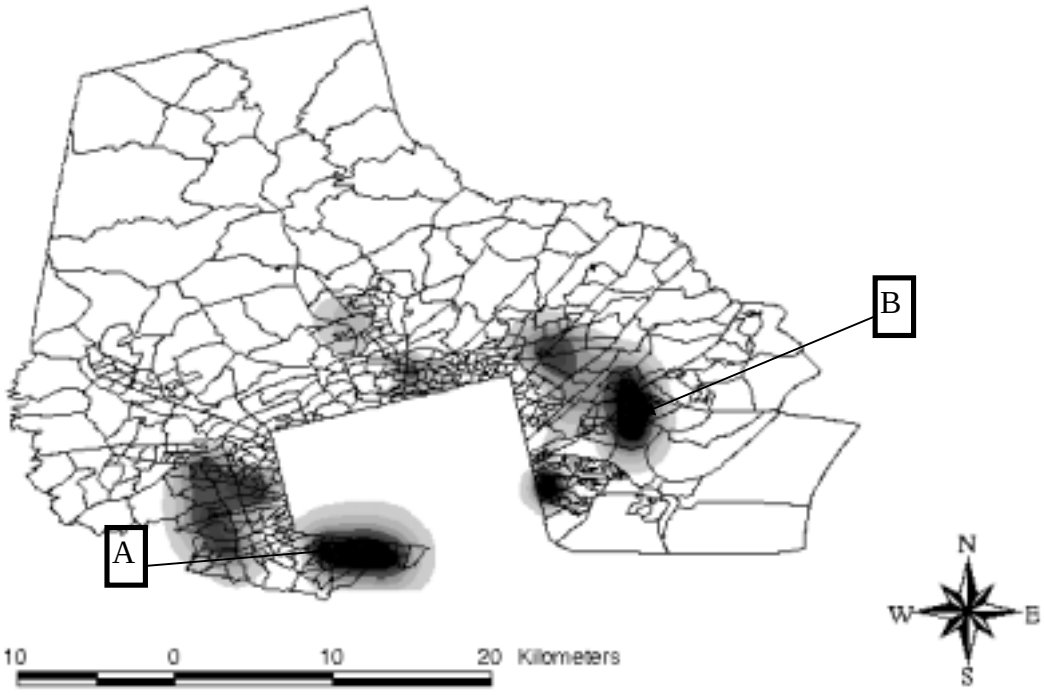

Figure 4: Residential crime hot spots, population at risk census night population

5.2 Street crimes

There are two possible populations at risk: census night time population and the length of street in each polygon. We investigated both, since it is relatively easy to calculate the length of street in each block group using an intersect command in the GIS. Figure 5 shows the hot spots detected if resident population is used as the population at risk. Again there are two large hot spots detected however in this case they are both situated well back from the edges of the map.

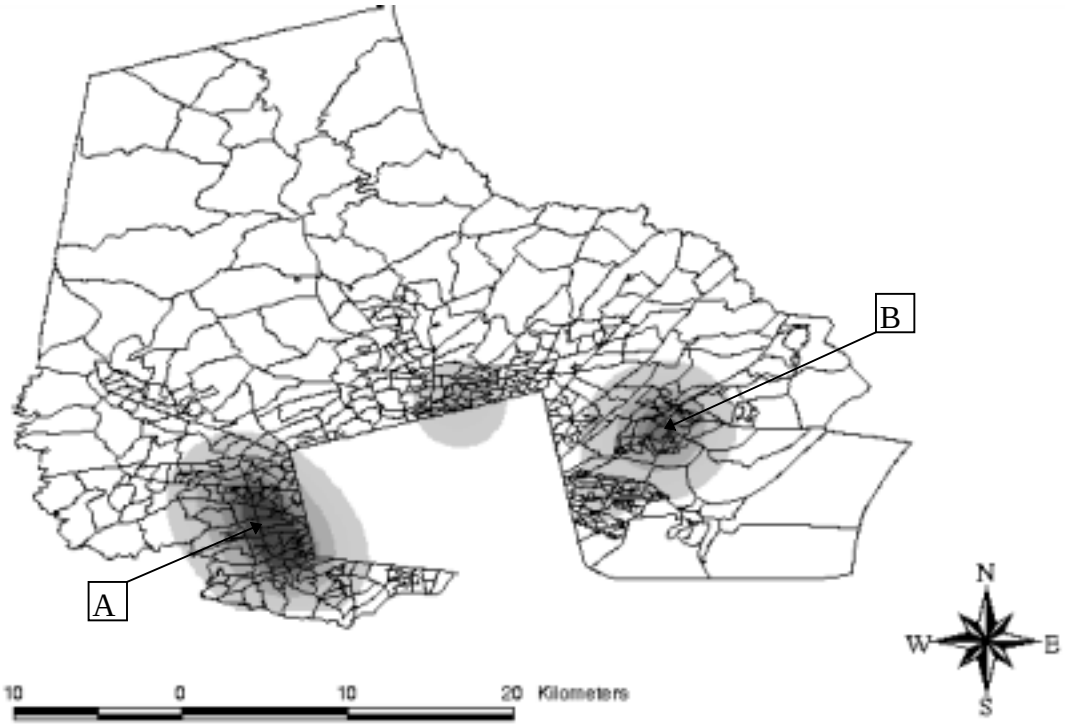

Figure 5: Street Crime Hot spots, population at risk census night population

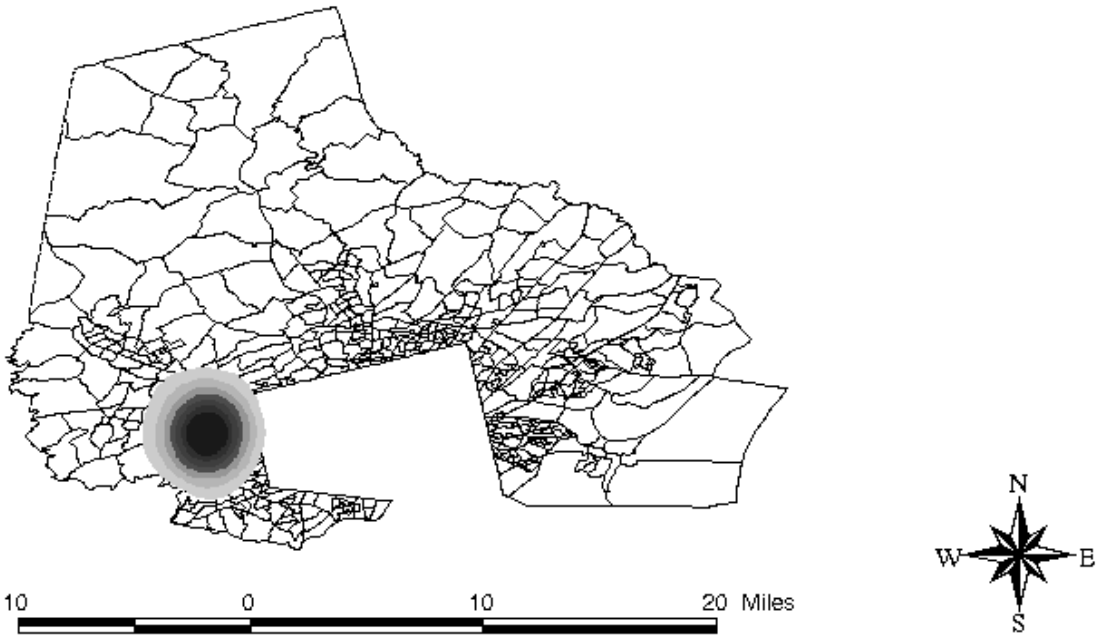

Figure 6: Street crime hot spots, population at risk street length

Figure 6 shows the pattern of hot spots detected by GAM when the length of street in each block group is used to define the population at risk, here only a single hot spot in the south west of the map is detected.

A residual question is how do you know when a hot spot that appears on the map is not a real hot spot at all? There are four possible answers. One is to feed randomised crime data into GAM and see what sorts of hot spots are found. Typically, they will be weak pathetic things that are not "highly peaked"; but how high does a peak have to be before you get excited about it? The answer is data dependent and qualitative. It also depends on what you plan to use the results for. The safest and simplest strategy is merely to look for the highest peaks. The second answer is to expend lots

more compute time on multiple testing by Monte Carlo simulation. This will indicate how easy it is to obtain results as extreme as those being observed by running GAM on multiple sets of randomly generated crime data sets with similar incidence to the observed data. The third solution is to keep for training purposes the results for differing degrees of clustering and use them to train the user to discriminate between the massively interesting and the rubbish. The strength of GAM is that it is a visual method of analysis. It is meant to suggest and create new insights in an almost artistic and qualitative kind of way. The fourth solution is to use GAM purely as an automated procedure and re-install the simple expert system that GAM/K had in 1990 before it was decided to use the human eye-ball instead. It is important to remember that there are many different causes of hot spots, many of which are related to the quality of the data being analysed. No machine based procedure can at present detect "bad data" for you; you have to do it and 100% automation would so reduce the value of human inputs as to render these spurious causes of clustering totally invisible to the end-user.

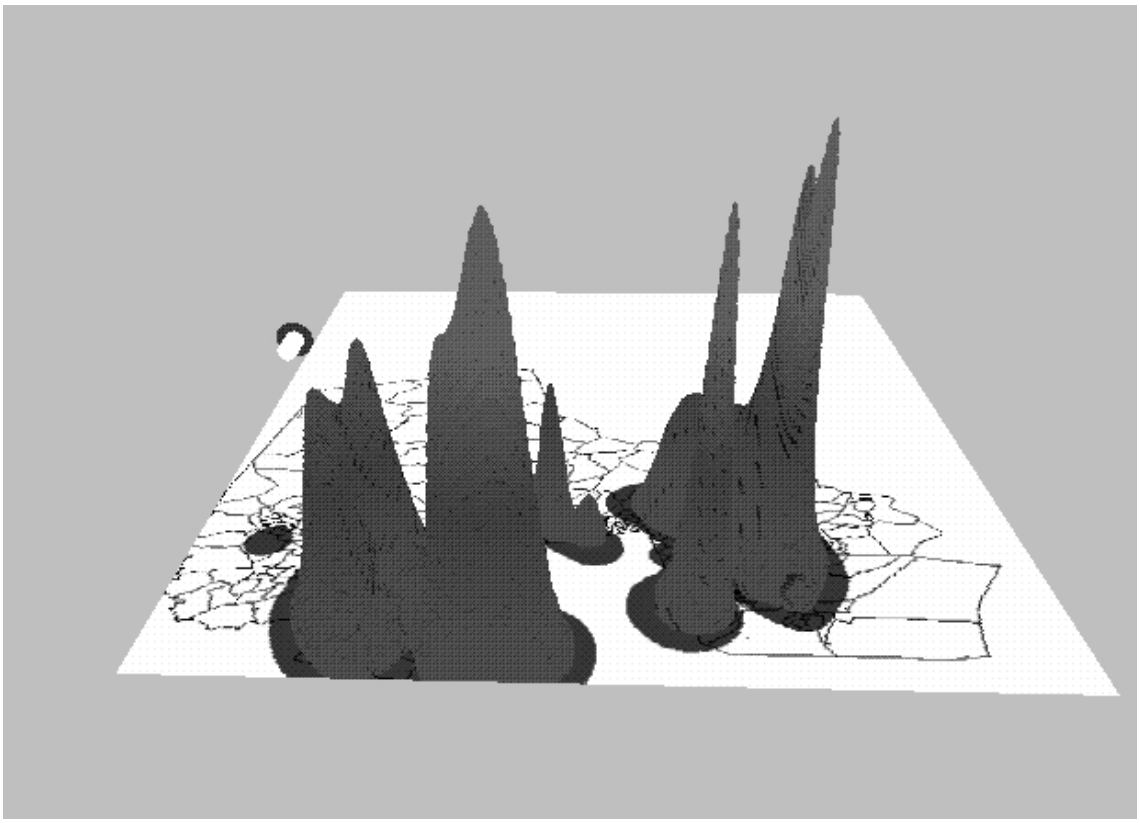

Figure 7: A VRML model of residential crime hot spots.

The on-line version of GAM allows users to view results in their browser, download the results for later analysis in a GIS or move around a VRML model of the results, an example of this is shown in figure 7.

6. Conclusions

The paper has demonstrated a viable approach to producing a distributed internet based approach to geographical analysis relevant to GIS applications. The same approach is being applied to other geographical analysis tools developed at Leeds with the Geographical Correlates Exploration Machine (GCEM, see Openshaw et al 1990) expected on stream soon. Other more animation oriented tools (such as

MAPEX, see Openshaw and Perree, 1996) is also planned. The Internet neatly avoids all the problems associated with vendor specific alternatives and has created a genuinely GIS invariant open geographical analysis formation. The GAM now exists as a virtual machine but the critical question is for how long? The notion of MIDAS national data servers is well established but the idea of obtaining analysis via a similar route is more novel but hopefully if could be added to the existing services. Whether it is worth the effort of using Java to create 100% portability for a 2-4 fold loss of performance is open to debate. A simpler alternative would be to create a portable source code version rather than two or three Unix versions plus NT and leave it at that. In the not too distant future as hardware costs continue to fall relative to the software, the idea of buying a geographical analysis machine (hardware plus software) will not appear as absurd as it once was.

Acknowledgement The research performed here was sponsored by ESRC via grant R237260.

7. References

Alexander, F. E., Boyle, P., 1996 Methods for Investigating localised Clustering of Disease

IARC Scientific Publications No 135, Lyon, France

Epanechnikov, V.A., 1969, 'Nonparametric estimation of a multidimensional probability density', Theor. Probab. Appl., 14, 153-158

Openshaw, S., Charlton, M., Wymer, C. and Craft, A.W., 1987, 'A mark I geographical analysis machine for the automated analysis of point data sets', Int. J. GIS, 1, 335-358

Openshaw, S., Charlton, M., Craft, A.W. and Birtch, J.M., 1988, 'Investigation of leukaemia clusters by the use of a geographical analysis machine', Lancet, I, 272-273

Openshaw, S., Wilkie, D., Binks, K., Wakeford, R., Gerrard, M.H. and Croasdale, M.R., 1989, 'A method of detecting spatial clustering of disease'. In: Crosbie, W.A. & Gittus, J.H., eds, Medical Response to Effects of Ionising Radiation, London, Elsevier, p.295-308

Openshaw S. & Craft, A. 1991, 'Using the Geographical Analysis Machine to search for evidence of clusters and clustering in childhood leukaemia and non-Hodgkin lymphomas in Britain'. In: Draper, G., ed., The Geographical Epidemiology of Childhood Leukaemia and Non-Hodgkin Lymphoma in Great Britain 1966-83, London, HMSO, p109-122

Openshaw, S., 1996, 'Using a geographical analysis machine to detect the presence of spatial clusters and the location of clusters in synthetic data', in F. E. Alexander and P. Boyle (eds) Methods for Investigating Localised Clustering of Disease IARC Scientific Publication No 135, Lyon, France, p68-87

Openshaw, S., Turton, I., Macgill, J., 1998, 'Using the Geographical Analysis Machine to analyse long term limiting illness data', Working Paper, School of Geography, University Of Leeds

Silverman, B.W., 1986, Density Estimation for Statistics and Data Analysis, London, Chapman & Hall