A GKS-BASED MICROCOMPUTER GRAPHICS PACKAGE FOR URBAN AND REGIONAL ANALYSIS AND PLANNING

Michael WEGENER

Department of Civil Engineering
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113, Japan

Klaus SPIEKERMANN

Department of Spatial Planning
University of Dortmund
Postfach 50 05 00
D-4600 Dortmund, West Germany

ABSTRACT

The graphics package described in this paper provides an integrated programming environment for IBM XT/AT compatible microcomputers based on the WATFOR-77 compiler of the University of Waterloo, Canada, and its implementation of the Graphical Kernel System (GKS) graphics standard. The package consists of a library of more than seventy FORTRAN77 subroutines which form, in the GKS terminology, an 'application layer' between the graphical primitives of GKS and a multitude of planning-related tasks of spatial analysis and presentation ranging from simple line draws and polygon fills to complex three-dimensional transformations of spatial data.

INTRODUCTION

> *"Solving a problem simply means representing it so as to make the solution transparent."*
>
> Herbert A. Simons: "The Sciences of the Artificial", 1969

A large part of all scientific work consists of presenting a problem in a different way. Just as in mathematics a result is derived through a sequence of transformations, so in other disciplines a conclusion follows from an appropriate presentation of facts. In both cases the object of investigation is not changed but brought into a form suitable for new insight.

This is particularly true for urban and regional planning, where diagrams, maps and other graphical representations traditionally are indispensable for demonstration and communication. Unfortunately their manual preparation requires great effort, time and cost.

Microcomputers with their rapidly developing graphics capabilities have the potential for efficiently generating sophisticated graphical representations at little cost. However, there exist no graphics software specifically addressing the needs of

urban and regional planners. As the microcomputer revolution is deeply transforming the world of business and commerce, the world of the urban and regional planner is still sadly lagging behind. Too small a market for profitable software development, the writing of planning software remains largely left to non-professional programmers such as graduate students, researchers or the planners themselves, and more often than not on a minimum budget.

It was with this target group in mind that the program package described in this paper was designed. It provides an integrated programming environment for IBM XT/AT compatible microcomputers based on the WATFOR-77 compiler of the University of Waterloo, Canada, and its implementation of the Graphical Kernel System (GKS) graphics standard. The package consists of a library of more than seventy FORTRAN77 subroutines which form, in the GKS terminology, an 'application layer' between the graphical primitives of GKS and a multitude of planning-related tasks of spatial analysis and presentation ranging from simple line draws and polygon fills to complex three-dimensional transformations of spatial data.

The embedding of graphics functions into the traditional programming language of geographers and planners distinguishes the package from stand-alone graphics packages and makes it particularly suited for immediate visualization of the results of computations such as simulations. Its minimal hardware requirements facilitate its application in planning education and in decentralized work environments.

In the paper an overview of the functions supplied by the subroutine package is illustrated by demonstration programs. In addition, examples of typical applications such as demography, migration analysis, transportation and digital terrain modelling are shown.

THE GKS GRAPHICS STANDARD

There are several ways to produce computer graphics. Each of them has its advantages and disadvantages:

(1) General-purpose graphics packages such as MS-Chart, Harvard Graphics or Autocad do not require the user to have programming skills, but are restricted by their built-in possibilities; no individual designs are possible. In addition, these programs are only good for drawing; any prior processing of the data has to be done in a separate program.

(2) Graphics commands of computer languages such as GW-Basic or Turbo-Pascal permit the immediate visualization of computations, which is invaluable if alone for debugging. However, there is so far no standardization of the graphics syntax of these languages, which greatly reduces the portability of such programs.

(3) Graphics standards are aimed at establishing a standardized interface between any graphics hardware and any computer language. The idea is that the interface translates a graphics command into appropriate hardware instructions depending on

the nature of the graphics device available thus freeing the user from considerations of technical detail. GKS (Graphical Kernel System) is the first international graphics standard. There exist conventions for calling GKS from Fortran, Pascal, Ada, and C. GKS is described in Enderle et al., 1987 and Encarnacao and Strasser, 1988.

The program package described in this paper uses the third approach by applying GKS together with Fortran. The reasons for choosing Fortran are as follows: Fortran is traditionally the computer language of geographers and planners and is continuously being revised in order to keep up with new developments. There are several implementations of GKS imbedded in Fortran available for microcomputers, workstations, minis and mainframes.

Figure 1 shows how Fortran, GKS and the computer hardware work together. The GKS software is a set of procedures, consisting of graphical primitives and hardware drivers for different types of hardware (or 'GKS workstations'). The graphical primitives are called as functions or subroutines from the Fortran program. However, as each of them performs only one elemental function, programming in GKS directly involves a great number of calls. Therefore it is convenient to combine several GKS calls that frequently occur together in a particular application into one macro routine performing a higher-order more complex function. A set of such routines is called an 'application layer' in the GKS terminology.

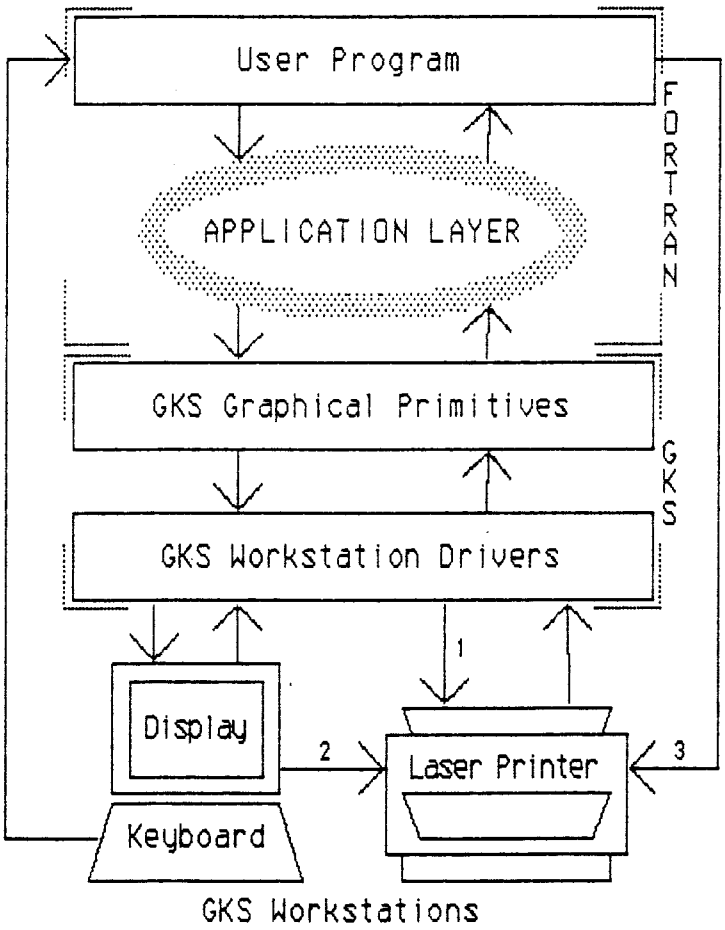

Figure 1. GKS and Fortran.

The program package described here is such an application layer. It consists of over seventy subroutines written Fortran77 using the Fortran calling conventions for GKS. It should with minor modifications run on any computer for which a Fortran compiler and a GKS implementation exist. However, presently only the WATFOR-77 compiler developed at the University of Waterloo, Canada, for IBM XT/AT and compatibles (Coschi and Schueler, 1985, 1986) and its implementation of GKS, WATCOM GKS (Yach, 1986), have been tested.

The WATFOR-77 compiler for IBM XT/AT and compatibles supports the full Fortran77 standard, has a good editor and an excellent debugger, and produces code that is among the fastest in the field (Voglewede, 1987). It can make use of an arithmetic 8087/80287 coprocessor. Normal operation of the compiler is compile-and-go, but it can produce executable load modules which, however, are quite large.

The present Version 1.3 of WATCOM GKS corresponds to 0a, the lowest performance level of GKS. There are no pointing devices such as mouse or light pen nor are there segments by which one can manipulate designated picture elements. Device drivers exist for CGA, Hercules, EGA and VGA displays and 9-pin matrix printers. Printer output can be generated in three ways (Figure 1): Output to a 9-pin matrix printer can be produced by the existing GKS driver (1) or as hardcopy (Shift-PrtScr) from monochrome or Hercules displays (2). Output to a 24-pin matrix printer or a laser printer is presently possible only by sending the display memory as hardcopy to the printer by a user-written driver (3). This driver does not utilize the much higher resolution of these printers and should therefore as soon as possible be replaced by a true GKS driver (1). All illustrations in this paper were produced on a laser printer using this method.

THE MACRO LIBRARY

The main objective in developing the program package was to provide routines for the graphical tasks most frequently encountered in the preparation of research papers, research reports or student theses in the field of urban and regional analysis and planning requiring as little programming by the user as possible and using the least expensive and most commonly available hardware. These objectives led to the following design principles:

- Self-contained procedures. Unlike in GKS where there are separate routines for setting parameters such as line type or area fill pattern, all parameters necessary for a routine are set explicitly in the calling statement.

- Restraint in using color. Although today color monitors are widely available, color printing is still expensive. Therefore techniques which distinguish graphical elements by line type, line width or shading pattern rather than by color were preferred. However, use of color with color monitors is possible.

- Use of raster techniques. For simplicity, heavy use was made of the possibilities of raster technology to overwrite and to OR, XOR or AND picture elements. Consequently, the procedures using such techniques are not suitable for pen plotter output.

Figures 1 to 5 show output of demonstration programs written to illustrate the graphical techniques made available by the subroutines. Figure 1 contains the basic elements: the default window (a), point/line drawings such as dots, lines, polylines, circles, ellipses, etc. (b), markers (c), area fills and shadings (d), thick lines (e) and fonts (f).

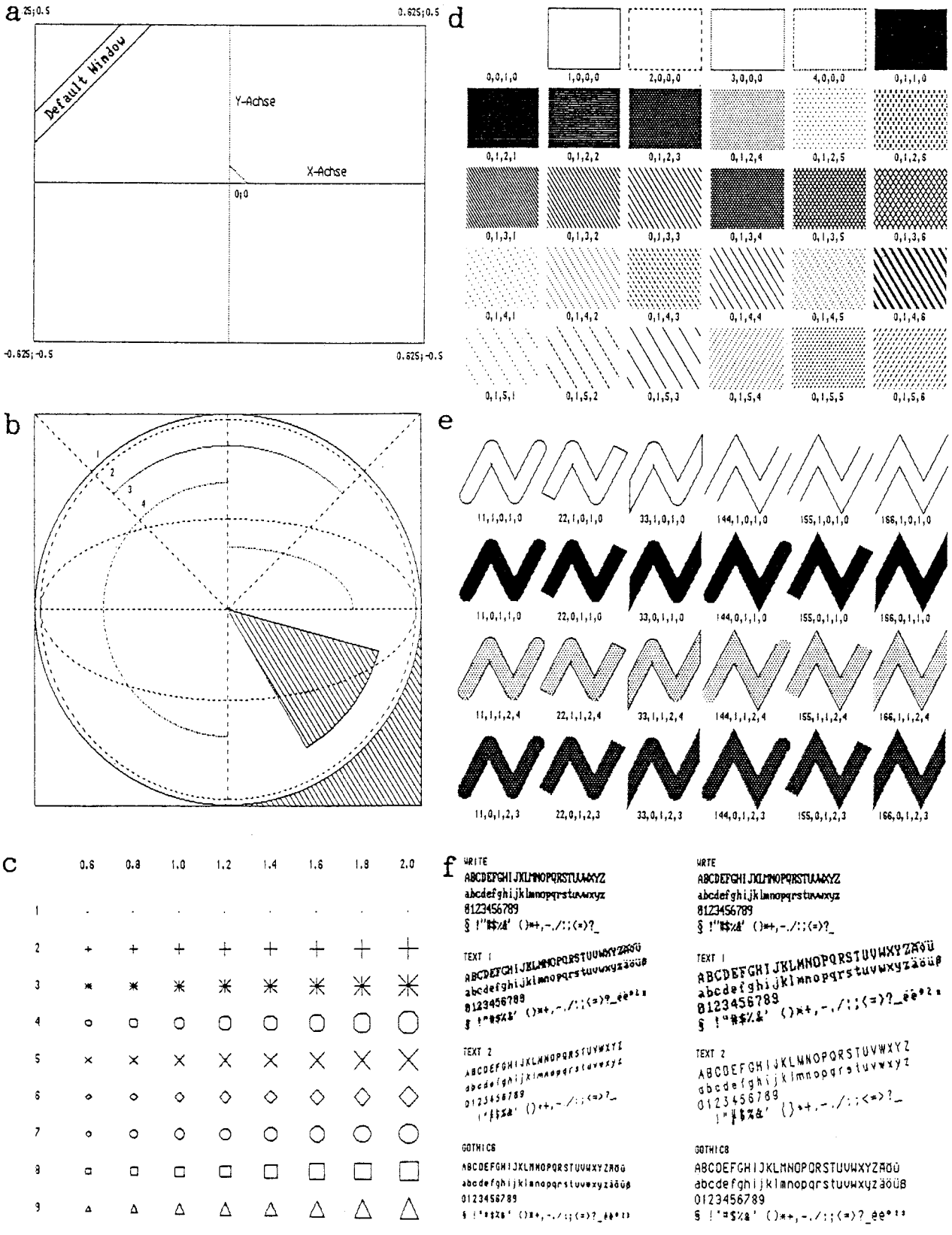

Figure 2. Basic elements of the macro subroutines: the default window (a), point/line draws such as dots, lines, polylines, circles, ellipses, etc. (b), markers (c), area fills and shadings (d), thick lines (e) and fonts (f).

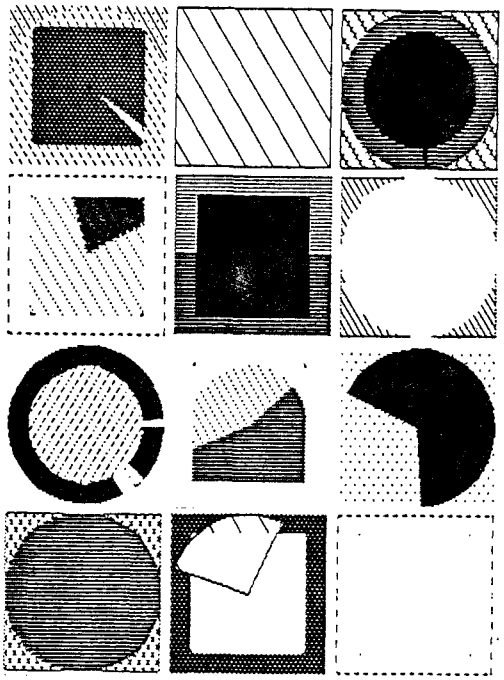

Figure 3. Boxes, frames, disks, wedges and rings.

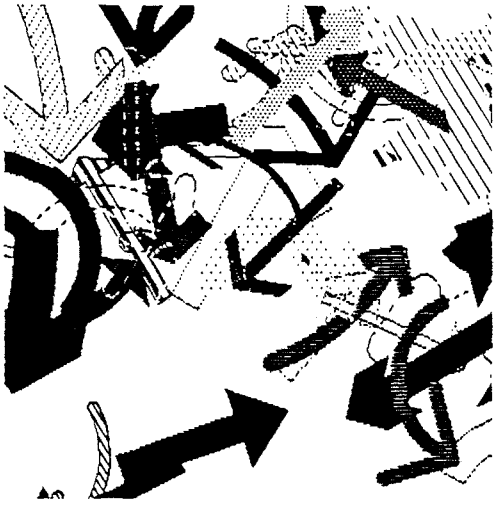

Figure 4. Just arrows.

Area fill subprograms exist for arbitrary polygons as well as for boxes, frames, disks, wedges and rings, which when overlaid can produce a large variety of patterns (Figure 3). One subroutine draws all sorts of arrows (Figure 4).

A separate group of subprograms deals with three-dimensional representations. The general idea is to establish a three-dimensional coordinate system or 'workbox' in which it is possible to draw using 3D-versions of the point/line and area fill subroutines familiar from two-dimensional drawing. Simple algorithms for hidden line removal are available. Figure 5 shows the three-dimensional workbox as wire-frame (a), as cube with hidden lines removed (b) and as open box with a ramp inside (c). Figure 6 compares two ways to represent surfaces: The raster scan technique (a) permits the axonometric representation of mathematical functions of x and y with hidden line removal. If the data are available for regularly spaced grid points, any surface can be displayed three-dimensional in true perspective (b).

Other macro subroutines not illustrated by the demonstration programs include printer and video control macros, macros to read and write bit images ('pictures') from the display to disk and vice versa, macros to generate device independent graphics files ('metafiles'), and a number of helpful utilities facilitating sorting and recurrent geometrical calculations.

The two following pages show the full set of macro subroutines and their calling syntax in the form of reference cards (Figure 7). The parameter names follow Fortran conventions, i,e, names beginning with i, j, k, l, m or n indicate a parameter of type INTEGER. Parameter names printed in bold indicate arrays, underlined parameters are output parameters.

The explanation of the function(s) performed by the macros and the meaning of their parameters are kept to a minimum. A full explanation of each subroutine and of the demonstration programs as well as numerous application examples are contained in Wegener and Spiekermann (1989).

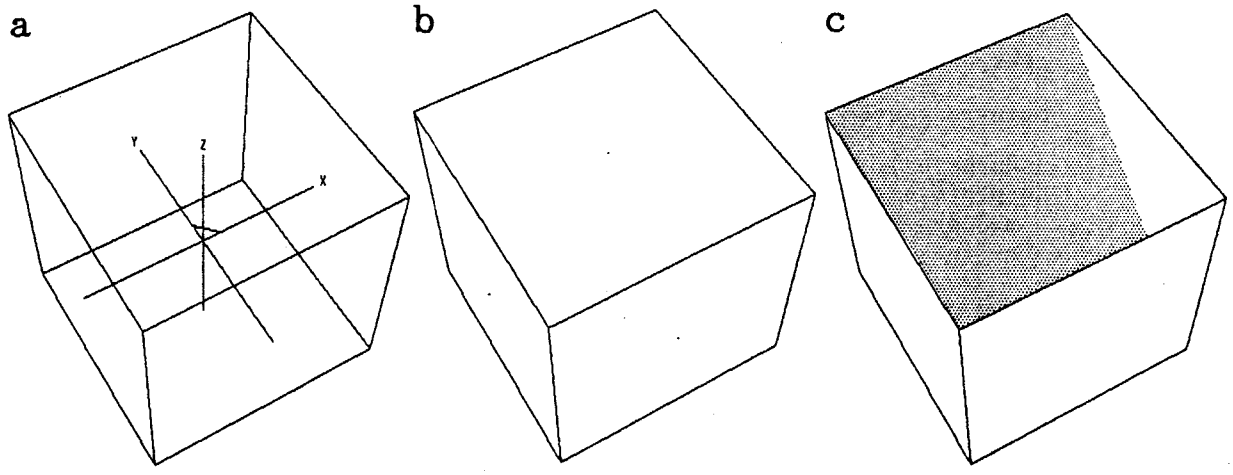

Figure 5. The 3D-workbox as wire-frame (a), as cube (b) and as open box with a ramp inside (c).

APPLICATION EXAMPLES

Figures 9 to 12 show examples of graphical representations produced with the subroutine package in a variety of research and student projects. For all representations individual programs calling the subroutines were written, no effort was made to standardize these programs for a larger class of problems.

Figure 9 contains an example of an age pyramid (a), a time series diagram with different line types (b), a three-parameter phase diagram (c), a Lorenz curve (d) and two applications of the grid plotting program of Figure 5, one to sampled data (e) and one showing a two-dimensional parameter surface from a calibration experiment (f).

Figure 10 contains examples of maps: a base map showing only zone boundaries and names (a), a map of population densities as grey-scale map (b), a map showing net

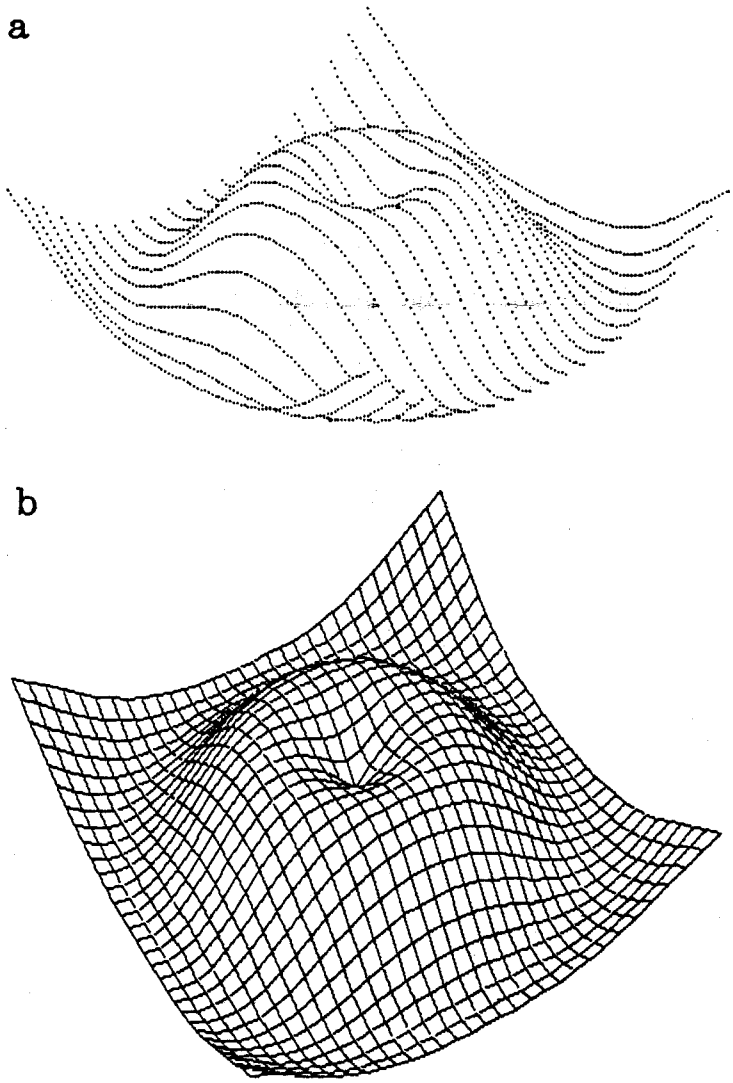

Figure 6. Three-dimensional representations of surfaces with raster scan technique (a) and with grid plotting routine (b).

GKS Macros 1

GKS CONTROL MACROS

igks	Initialize GKS
cgks	Close GKS

PRINTER CONTROL MACROS

askprn(ips)	Inquire printer status byte
setprn(m,n)	Initialize matrix printer
m=0 n=1/2	alpha 12/17 cpi, 6/8 lpi
m=1 n=1	graphics 120 dpi, 9 lpi

VIDEO CONTROL MACROS

cls	Clear alpha screen
clear	Clear graphics screen
askvideo(m,nc)	Inquire video mode
m=7/15, nc=80/90	alpha/graphics 80/90 cpl
setvideo(m)	Set video mode
setcolor(ic)	Set colour code
inverse(n)	Invert graphics screen
setattr(ir,ic,ia)	Set screen attribute at ir,ic
ia=1/7/8 (129/135/136 blinking)	underline/normal/black
ia=9/15/112 (137/143/240 blinking)	high-underline/high/inverse
askcur(ir,ic)	Inquire cursor position
setcur(ir,ic)	Position cursor at ir,ic
cursor(ic)	Set cursor attributes
setwrap(m)	Set line wrap

OUTPUT CONTROL MACRO

pmode(pm)	Set plot mode
pm='r'/'x'/'a'/'o'	replace/XOR/AND/OR

POINT/LINE MACROS

getdot(x,y) /REAL function/	Inquire pixel
putdot(x,y,ici)	Draw point
ici=0/1	black/white
putmk(x,y,mt,scf,ici)	Draw marker
mk=1/2/3/4/5/6/7/8/9	./+/*/o/x/◇/o/□/△
line(x1,y1,x2,y2,lci)	Draw line
lci=0/1/2/3/4	black/solid/--/../.-
pline(n,**x**,**y**,lci)	Draw polyline
circle(x,y,r,lci)	Draw circle
ellipse(x,y,rx,ry,lci)	Draw ellipse
circarc(x,y,r,al,bt,lci)	Draw circular arc
ellarc(x,y,rx,ry,al,bt,lci)	Draw elliptical arc

AREA/FILL MACROS

paint(x,y,ici,ist,jst)	Fill area
polygon(n,**x**,**y**,lci,ici,ist,jst)	Draw/fill polygon
ist=0/1/2/3	hollow/solid/pattern/hatch
jst=0/1/2/3/4/5/6	see **demo3**
box(x1,y1,x2,y2,lci,ici,ist,jst)	Draw/fill box
frame(x1,y1,x2,y2,d,lci,ici,ist,jst)	Draw/fill frame
disk(x,y,rx,ry,lci,ici,ist,jst)	Draw/fill disk
wedge(x,y,rx,ry,al,bt,lci,ici,ist,jst)	Draw/fill wedge
ring(x,y,rx,ry,d,al,bt,lci,ici,ist,jst)	Draw/fill ring
tline(x1,y1,x2,y2,d,m,lci,ici,ist,jst)	Draw/fill thick line
tpline(n,**x**,**y**,d,m,lci,ici,ist,jst)	Draw/fill thick polyline
m=x.. 0/1	ball/flash
m=.xx 1/2/3 (4/5/6)	round/flat/vertical (open)

Figure 7. GKS Macros reference card.

TEXT MACROS

keybd()	Test keyboard buffer
inkey(n,chr)	Read keyboard buffer
getchar(ir,ic) /CHARACTER function/	Read character at ir,ic
putchar(ir,ic,chr)	Write character at ir,ic
write(x,y,txt)	Write text at x,y (ROM font)
wrte(ir,ic,txt)	Write text at ir,ic (ROM font)
text(x,y,ift,al,ht,xpf,spf,txt)	Draw text at x,y (GKS fonts)
gothic6(x,y,spf,txt)	Draw text at x,y (GOTHIC6)
gothic8(x,y,spf,txt)	Draw text at x,y (GOTHIC8)

PICTURE MACROS

getpic(x1,y1,x2,y2,mpic,lpic,**pic**)	Read picture from screen
putpic(x,y,**pic**)	Write picture to screen
savepic(fn,**pic**)	Save picture to disk
loadpic(fn,**pic**)	Load picture from disk

METAFILE MACROS

ometa(fn)	Open GKS metafile for output
cmeta	Close GKS metafile
rmeta(fn)	Read and execute GKS metafile

3D MACROS

workbox3d(vd,al,bt,scf,m)	Establish/draw 3D workbox
window3d(xl,xr,yl,yr,zb,zt)	Set 3D window
trans3d(x,y,z,xp,yp)	Transform 3D coordinates to 2D
putdot3d(x,y,z,ici)	Draw 3D point
line3d(x1,y1,z1,x2,y2,z2,lci)	Draw 3D line
pline3d(n,**x**,**y**,**z**,lci)	Draw 3D polyline
polygon3d(n,**x**,**y**,**z**,lci,ici,ist,jst)	Draw/fill 3D polygon
box3d(x1,y1,z1,x2,y2,z2,lci,ici,ist,jst)	Draw/fill 3D box
dsort3d(m,np,**ip**,**id**,n,**x**,**y**,**z**)	Depth sort of 3D polygons
surface3d(z)	Draw 3D surface of function z
grid3d(n,m,xa,xe,ya,ye,**z**)	Draw 3D grid of points **z**(n,m)

DATE/TIME MACROS

date(iy,im,id,iw)	Inquire system date
time(ih,im,is,ihs)	Inquire system time

RANDOM NUMBER MACROS

rnd() /REAL function/	Random number generator
setrnd(i)	Initialize **rnd**
i=n/0	seed n/seed from **time**

SORT MACROS

isort(n,**iz**,**ind**)	Sort INTEGER numbers
rsort(n,**rz**,**ind**)	Sort REAL numbers
csort(n,**sf**,**ind**)	Sort CHARACTER fields

GEOMETRY MACROS

angle(x1,y1,x2,y2)	Angle of line in radians
sll(x1,y1,x2,y2,x3,y3,x4,y4,n,xs,ys)	Intersection line/line
slc(x1,y1,x2,y2,xm,ym,r,n,xs,ys,xt,yt)	Intersection(s) line/circle

DOS MACRO

dos(cmd)	Execute DOS command or program
cmd=command	command or program name

Figure 8. GKS Macros reference card (continued).

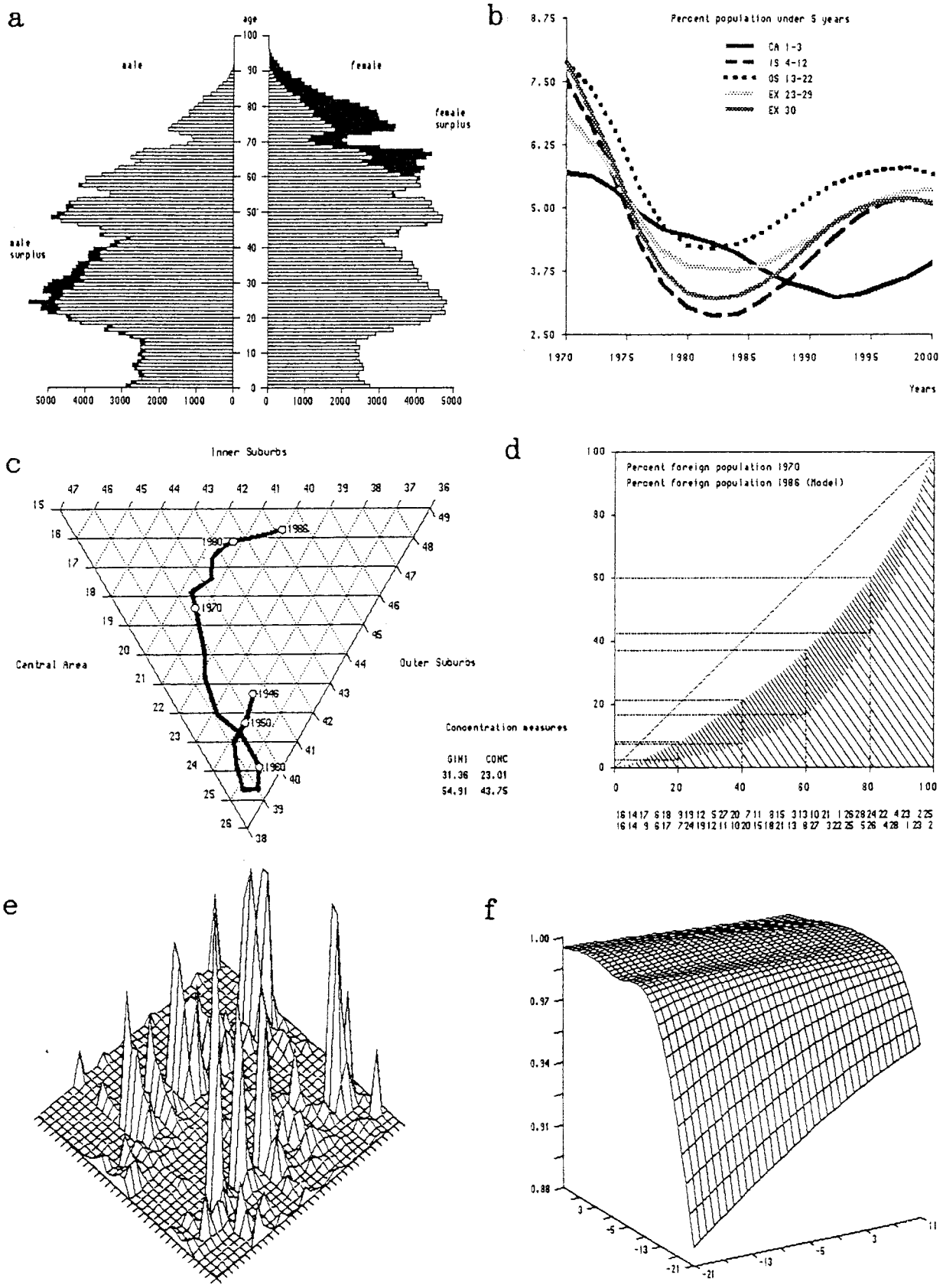

Figure 9. Application examples: Age pyramid (a), time series diagram (b), three-parameter phase diagram (c), Lorenz curve (d), grid display of sampled data (e) and of two-dimensional parameter surface (f).

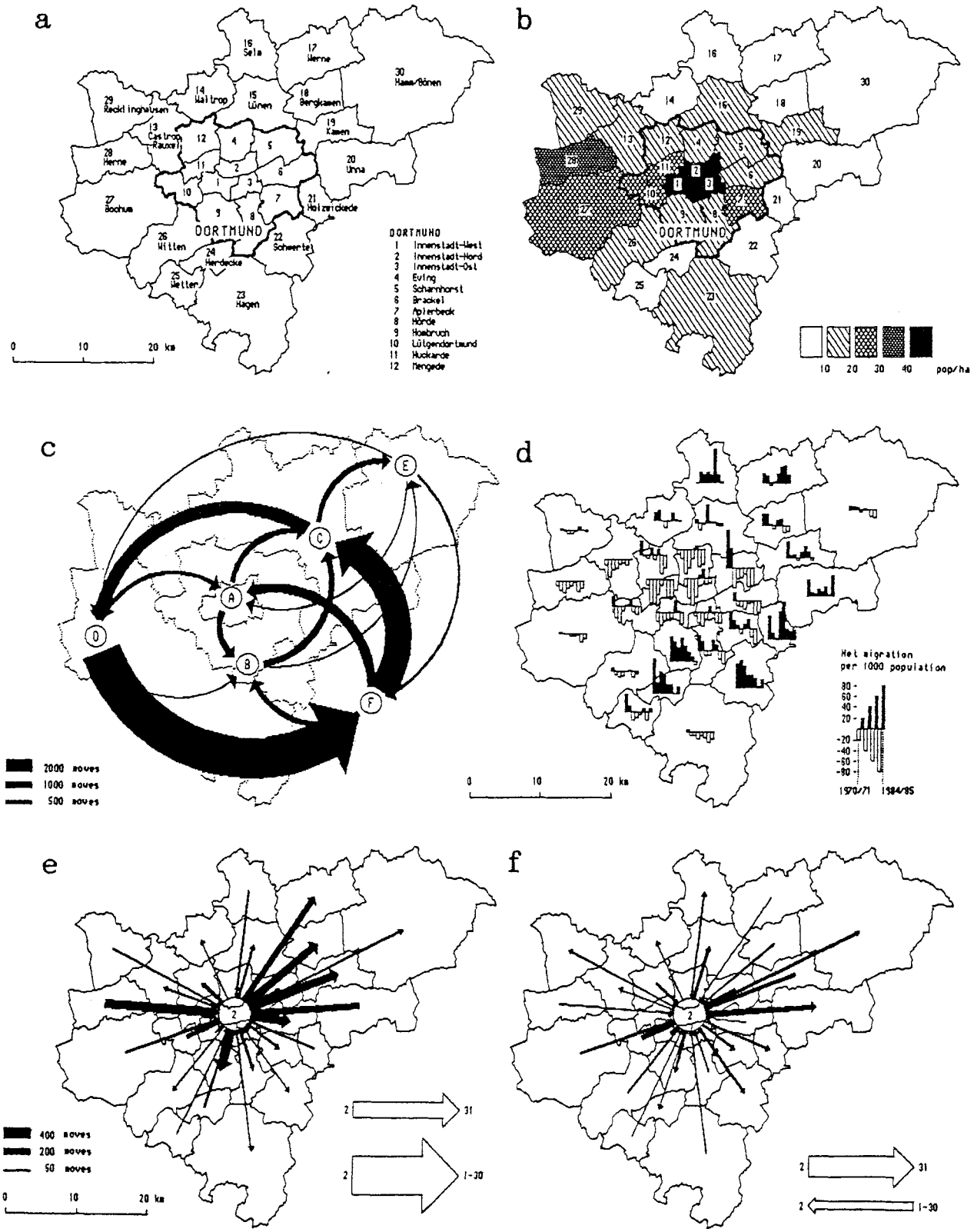

Figure 10. Application examples: maps of the urban region of Dortmund, West Germany, including a base map (a), a grey-scale map of population density (b), maps showing net migrations as arrows between zones (c) or for each zone as positive and negative bars (d) or comparing net migrations for one zone in different years (e and f).

migrations as arrows between regions (c) or as zonal balance in the form of bars indicating migration gains or losses (d). The last two maps show net migrations flows for one particular zone for two different years (e and f).

Two final examples show the output of programs designed to support transport network analysis and digital terrain modeling. Figure 11 shows a section of the Dortmund transportation network (a) and the same section with buffers along each link (b). Figure 12 shows the interpolation of a surface from irregularly spaced observation points through triangulation (a), contours (b and c), grid points (b) to three-dimensional representation (d) using ACM algorithm 626 (Preusser, 1984). These two applications also demonstrate the restrictions imposed by the limited resolution of the microcomputer displays available.

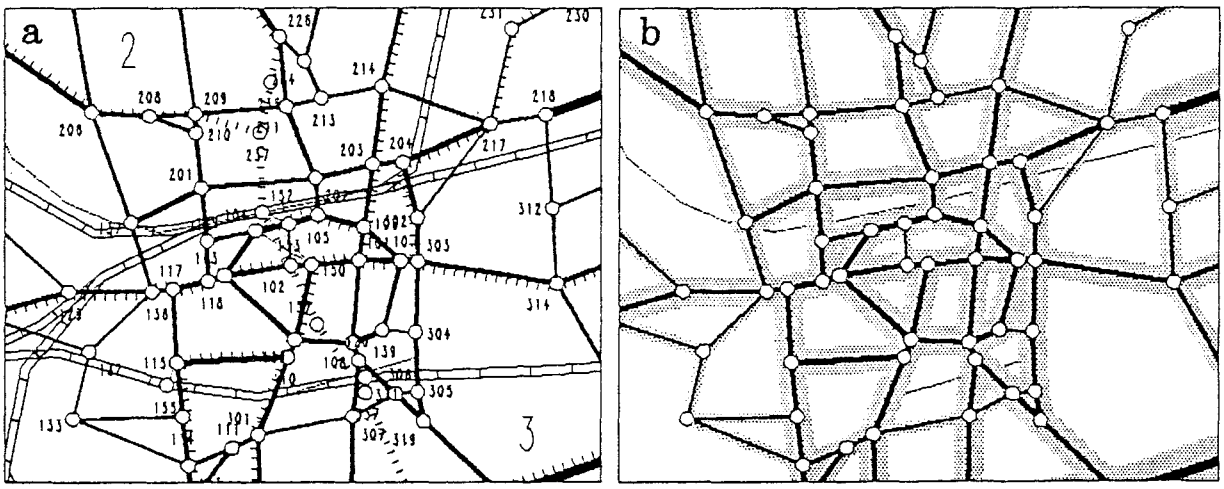

Figure 11. Network analysis: a section of the Dortmund transportation network (a) and with buffers along each link (b).

CONCLUSIONS

The present paper has demonstrated the feasibility and potential of generating sophisticated computer graphics for urban and regional analysis and planning using the GKS graphics standard embedded in Fortran on commonly available low-cost microcomputer hardware.

Its minimal hardware and software requirements facilitate the application of this or similar program packages in planning education and in decentralized work environments or in countries where more expensive hardware and software is not available.

The limits of the hardware, on the other hand, in particular the low resolution of present microcomputer displays, seem to restrict its application to relatively small planning problems, although real-life planning problems are usually large and complex. It remains to be seen if there is enough demand for analytical tools at the level of complexity addressed in this package.

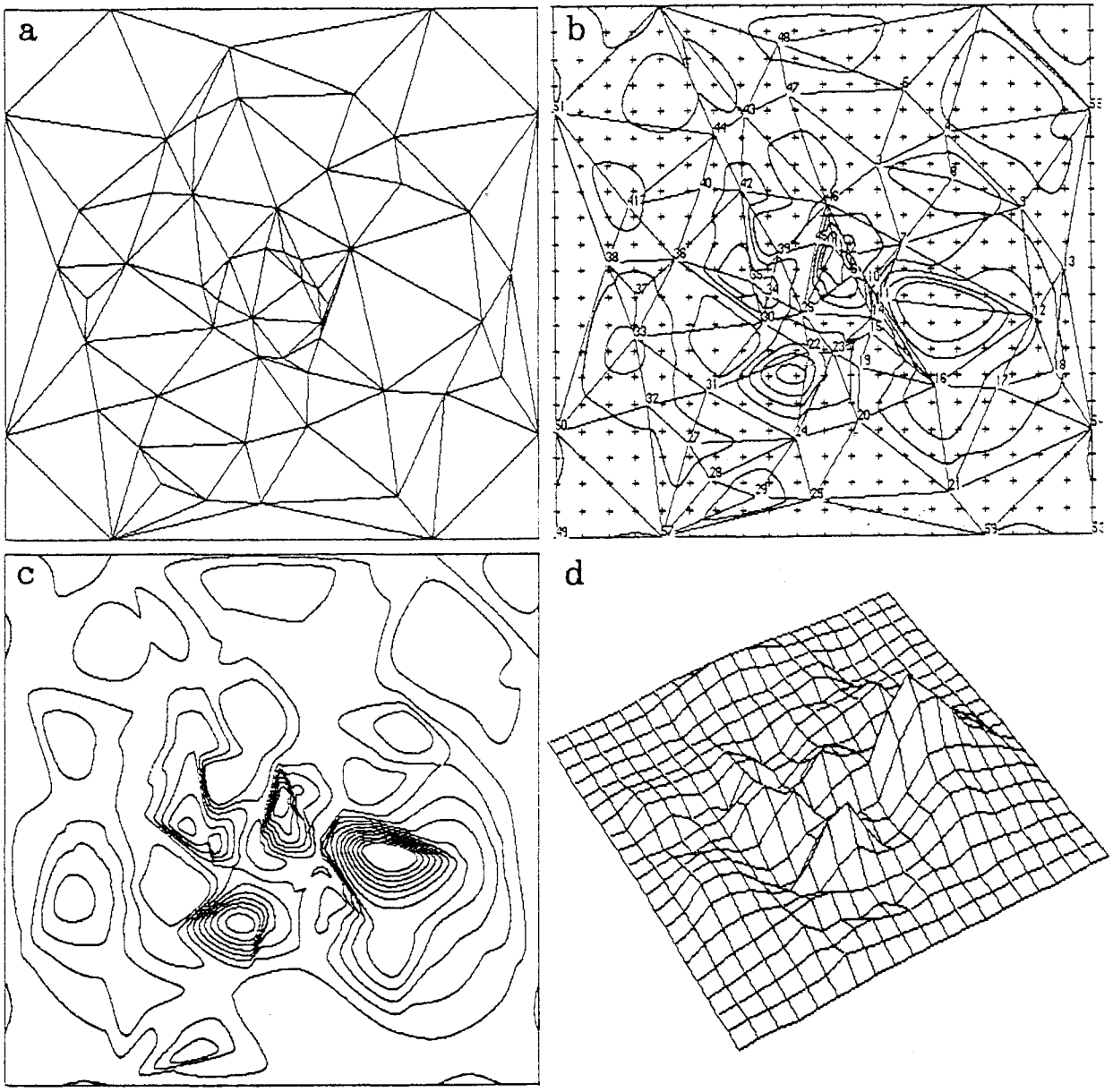

Figure 12. Digital terrain modeling: interpolation of a surface from irregularly spaced data points through triangulation (a), contours (b and c) to three-dimensional representation (d).

Another point which is not at all clear is whether a graphics packages requiring computing skills has a future in the professional practice. This question is of course related to the lack of specialized planning software in the commercial market. It is probably safe to say that programming skills will be required from the urban analyst at least as long as the market fails to offer high-quality planning software that is easy to use and still retains the flexibility necessary for the ever-changing tasks of the planning practice.

A final question is related to the possibility of writing software by people without the expertise of modern software development. It may well turn out that, even if the original idea for the software is a good one, the tasks of achieving and main-

taining high standards of reliability and of continually updating the software to keep up with the state of the art and the changing hardware environment, are beyond the capacity of non-professional programmers.

REFERENCES

Coschi, G. and Schueler, J.B. (1985): WATFOR-77 User's Guide IBM PC with DOS. Waterloo, Ontario: WATCOM Publications.

Coschi, G. and Schueler, J.B. (1986): WATFOR-77 Language Reference. Waterloo, Ontario: WATCOM Publications.

Encarnacao, J.L. and Strasser, W. (1988): Computer Graphics. München: Oldenbourg (in German).

Enderle, G., Kansy, K. and Pfaff, G. (1987): Computer Graphics Programming. GKS - The Graphics Standard. Berlin/Heidelberg/New York: Springer.

Preusser, A. (1984): ALGORITHM 626. TRICP: A Contour Plot Program for Triangular Meshes. ACM Transactions on Mathematical Software Vol. 10, No. 4, pp. 473-475.

Voglewede, J. (1987): FORTRAN Perspectives. PC Tech Journal, June 1987, pp. 92-109.

Wegener, M. and Spiekermann, K. (1989): Mikrocomputergraphik: Eine Unterprogrammsammlung für FORTRAN und GKS (Micro Computer Graphics: A Subroutine Library for FORTRAN and GKS). Berlin/Heidelberg/New York: Springer (in German).

Yach, D. (1986): WATCOM GKS Tutorial and Reference. Waterloo, Ontario: WATCOM Publications.