Depthmap 4

A Researcher's Handbook

Alasdair Turner, June 2004

Citation: Turner, A., 2004, “Depthmap 4 — A Researcher’s Handbook”, Bartlett School of Graduate Studies, UCL, London.
http://www.vr.ucl.ac.uk/depthmap/handbook/depthmap4r1.pdf

Revision 1 15-Sept-04

This handbook is for Depthmap versions 4.06r to 4.09r

Draft Version without figures and appendices

Abstract

This document gives an overview of the features available in Depthmap version 4. Depthmap is primarily a computer program to perform visibility analysis of architectural and urban systems. It takes input in the form of a plan of the system, and is able to construct a map of 'visually integrated' locations within it.

In addition, the most recent version of Depthmap now supplies a range of configurational analyses which come under the umbrella term of 'space syntax'. Space syntax analyses examine the relationships between components of space; each analysis starts with a representation of the spatial components, then makes a graph of these components, and finally analyses this graph using, for the most part, conventional graph theoretical measures. The analyses available within Depthmap include the original visibility analysis, generation and analysis of axial maps as well as segment analysis. In addition, a plug-in module gives users the ability to perform agent-based analysis, and a 'software developers' kit' (SDK) allows users to program their own graph analysis.

Depthmap was first written for the Silicon Graphics IRIX operating system as a simple isovist processing program in 1998. Since then it has gone through several metamorphoses to reach the current version 4, for the Windows platform. It is designed to run on Windows 2000 and XP operating systems.

This document is still in draft form; as yet, figures are still to be added, as are the appendices. The version of Depthmap it describes (4.06r) is newly released.

Contents

Chapter 1

Introduction

1 A background to Depthmap

The original concept behind Depthmap developed from two strands of thought. One was isovist analysis (Benedikt, 1979), and the other space syntax (Hillier and Hanson, 1984). Benedikt created maps of properties of the visual field at points within plans of buildings. He drew contours of equal visual area within the plan and called the resulting map an 'isovist field'. He believed that these maps would give an insight into how people navigate the actual building. Since closely packed contours would indicate rapidly changing visual field he reasoned that these would indicate decision points within the building. Independently, Hillier and Hanson developed the theory of *space syntax*. They created various representations for the components of space; they then drew maps of these components, and crucially, the relationships of the components with each other. Within the space syntax community, the representation that has become most used is the axial map. The actual derivation of an axial map, as we will see, is quite complex, but essentially it involves drawing a set of lines through the open space of the plan. Hillier and Hanson then created an interesting twist to established theory at the time. They created a graph using the axial lines themselves as nodes, so that each line was considered connected to others that it intersected. From this graph, they calculated how well 'integrated' each line was with respect to all the others in the graph, that is they calculated a measure of the average number of steps it takes to get from one line to any other within the axial map. The integration of axial lines is of particular interest to researchers as it correlates well with the number of pedestrians found to be walking along the axial line (Hillier et al., 1993, and numerous other studies since).

Since Benedikt had theorised that isovist fields would correspond in some way to movement patterns of people and Hillier et al. had shown that relationship between lines through the space does correspond with movement patterns within space, it was decided to combine isovist fields with space syntax to provide a measure of how well integrated isovists themselves are within a plan of an environment (Turner and Penn, 1999). The methodology was later formalised more simply as *visibility graph analysis* (VGA) (Turner et al., 2001). In VGA, a grid of points is overlaid on the plan. A graph is then made of the points, where each point is connected to every other point that it can see. The *visual integration* of a point is based on the number of visual steps it takes to get from that point to any other point within the system. Various graph measures, not just integration, may be made: the idea was that all possible occupiable locations within the built environment would be categorised by their visual relationships to other occupiable spaces through a continuous map. Due to its providence, it was hypothesised that VGA would give a good indication of how people might interact with space — either moving through it (Desyllas and Duxbury, 2001) — or standing, discussing or generally occupying it [see, for example, Doxa's (2001) analysis of the usage of the National Theatre and Royal Festival Hall in London]. Depthmap was the tool created to perform these analyses, and how to perform them comprises the basis of this paper.

Whether or not VGA actually succeeds in its aims is open to debate. For people movement, a recent study has shown that other methods based on space syntax seem to correlate better with pedestrian count rates in cities (Turner, 2003), although an earlier result from a large department store was promising (Chapman et al., 1999). Regardless of the outcome of the debate, there does appear to be a promising avenue of research for modelling people movement which uses the visibility graph at its core, if not a direct measurement of the graph. Penn and Turner (2002) and Turner and Penn (2002) proposed an agent-based analysis with an underlying visibility graph. In this analysis, 'agents' (automata within a computer representing people) are released into a plan of the environment and navigate using the visibility information directly available to them through the visibility graph. A simple method of path selection which approximates choosing the longest line of sight (Turner, 2005) is applied. The results of the analysis have correlated well with pedestrian movement both within a building environment ($R^2 = 0.77$, Turner and Penn, 2002) and within an urban environment ($R^2 = 0.68$, Turner, 2003). The agent-based analysis proposed by Penn and Turner has been produced as a plug-in module to Depthmap, and a short discussion of how to apply it will be included herein.

There now follows a non-sequitur. Depthmap can also analyse axial maps. There are two independent reasons for this. The first is due to an old argument regarding the definition of the axial map, which, with Hillier and Penn, I wanted to resolve once and for all. The second is due to an idea floated in a paper from 2001, to which I will return in a moment. The argument about the axial map is that while researchers appear to able to draw axial maps easily, they are extremely hard to produce using computer software (Ratti, 2004). Peponis et al. (1998) get very close to a solution. They start with what is called an 'all-line map' (Penn et al., 1997), that is a map of all the lines that may be drawn through the open space of a plan in one of the following ways: (a) connect two convex corners, (b) connect one convex and one reflex corner such that the line can be extended through open space past the reflex corner, or (c) connect two reflex corners such that the line can be extended through open space past both of them. Peponis et al. then reduce these lines using a so-called 'greedy' algorithm to a near-minimal set of lines that complete all topological loops and fully observe the whole system. This would be the solution to the problem, but their algorithm only works with simplified plans. Further, it has been pointed out that a greedy algorithm does not necessarily give a unique axial graph (Batty and Rana, 2002). Thus, in Turner et al. (2004), with Hillier and Penn, I address these twin problems to conceive a suitable reduction of lines that forms a unique graph for any given configuration, from any arbitrary plan. The result looks very similar to a hand-drawn axial map. If further reduced using a greedy algorithm, it is nearly indistinguishable from the hand-drawn map. These algorithms were developed and tested within Depthmap. Once the axial map construction was completed, since Depthmap already performed graph analysis, it was a simple matter to include an additional option to load predrawn maps, and perform axial analysis of them, whether predrawn or generated within Depthmap.

The second reason for the inclusion of axial analysis within Depthmap stems from an obscure research report, where a method of 'angular analysis' of VGA maps is proposed (Turner, 2000). Dalton (2001) introduced a similar method for the analysis of axial maps, which he called 'fractional analysis'. Fractional analysis is an elegant and simple algorithm that weights axial graphs according to the angle of turns within axial map. At the same conference, I suggested a small improvement to Dalton's algorithm: a solution to the *segment problem* within axial analysis (Turner, 2001*a*). The segment problem is one concerned with increasing resolution. It would seem natural to break axial lines down into their constitent street segments between junctions, in order to make an analysis at this scale. The problem is that if you segment an axial map, a previously integrated long line turns into a segregated mass of shorter lines, as you must take a step each time to pass from one segment to another along the length of the line. The segment problem, in fact, is one that would seem to have held back many network models of cities, for, as Hillier (1999) has pointed out, the key to movement within cities appears to be linearity of routes, and the logic of the city appears to be a deformed grid of near orthogonal linear routes. The segmented line breaks the linearity of routes, and fails to distinguish between the orthogonal lines crossing the grid and the continuation of near linear routes. Thus, the answer to the segment problem appears to be to use an angular weighting of connections, so that passing in a straight line has weight zero, while turning 90° costs '1' step, and turning back on the line completely, a 180° turn, costs '2' steps, with a continuous range in between. Recently, Hillier and Iida (2004) have implemented the first true angular analysis of axial maps using this method, and show a host of impressive correlations with pedestrian movement 'gate' counts. Since Iida has written the software to perform this calculation, there would seem to be little reason to include angular analysis in Depthmap. However, there is a further idea alluded to in the original 'angular analysis' paper: that a better analysis than weighting turns continuously might be to weight them in discrete graduations, '0' for approximately straight on, '0.5' steps for near 45°, '1' step for near 90° and so on, although the number of graduations is of course arbitrary (Turner and Penn, 2005). The inspiration for this 'quantised' angular analysis is due both to Hillier's (1999) observation that grids are 'almost' (but not exactly) regular, and Montello's (1991) observation that people tend to 'round' turns in routes to the nearest 45°; therefore, it is reasoned, human movement might correspond to this facet of human perception. The method of analysis has been called 'tulip analysis', after the tulip maps that rally-car drivers use to navigate, and is incorporated into Depthmap, along with standard angular analysis.

The preceding paragraphs cover the broad typology of analysis currently available in Depthmap. This paper goes on to describe how they are applied within the program in more detail, and further elements of the analyses possible. The paper is structured into seven further chapters as follows. Chapter 2 deals with the pre-analysis phase, importing plans to Depthmap, moving around, zooming and so on, as well as how to view different line layers. Chapter 3 then goes onto the core functionality, creating and analysing the VGA graph, and how to transfer the datas into different 'layers', for example, to compare VGA data with observed pedestrian gate counts. Chapter 4 examines axial line map creation, generation and analysis, while chapter 5 examines segment analysis. Chapter 6 explains features principally regarding views, such as how the colour scale works and how it may be changed, but also includes linking between floors. Chapter 7 talks about how to get data *out* of Depthmap, including exporting images and data tables created within Depthmap. The final chapter offers a brief conclusion, and my thoughts on the future of VGA as whole. There are four appendices. The first appendix contains mathematical definitions for the graph measures calculated by Depthmap; the second describes the use of the plug-in agent module; the third outlines the software developers' kit (SDK). The fourth appendix contains a change history from the last major release of Depthmap. Throughout the paper, rather than giving precise usage notes, I discuss why the analysis is done as it is, and sketch the direction I want to take Depthmap in. I want to stress that this direction is not fixed, it requires your — the Depthmap user's — input into how the program should develop.

2 Some general notes on the Depthmap interface

In order to read the text you will need to be familiar with a few basic graphical user interface (GUI) concepts. I will talk of 'the mouse', 'icons', 'clicking', 'windows', 'menus', 'toolbars', 'status bars', 'dialog boxes' and 'cursors', as well as others. 'Mouse', 'icon' and 'clicking' you will have to know before you start, as well as 'windows' (Depthmap has a main window which opens when you run the program, and subwindows that appear when you open or make a new graph). 'Menus' appear on a bar at the top of the main Depthmap window; the toolbar is a row of 'tools' just below the menu bar that you may select by clicking on them with the mouse. The status bar is an information strip along the bottom of the main window. An 'edit box' is a small box which contains some editable text. A 'drop list' looks like an edit box with a down arrow next to it, which you may click to expand a list of options to choose between, predominantly by clicking on the option you require. A 'dialog box' is a box that typically appears when you select a menu option with an ellipsis after it (e.g., 'Open...'). It usually contains an assortment of buttons, edit boxes, drop lists etc. which collectively allow you to set up, for example, a complex analysis. The 'cursor' describes the mouse pointer on the screen. It is usually an arrow, but on selecting certain options it might change it something else, for example, a magnifying glass or a hand symbol.

It is worth noting that there is some logic to the layout of the menus and toolbar. The logic is shaped around the basic unit, or document, handled by Depthmap, the *.graph* file (or simply 'graph file'). A graph file encapsulates everything for your analysis: the plan, the visibility graph or axial graph itself (or both), and the results of the analysis, a set of 'attributes' arranged into columns when listed out. For example, a column of 'connectivity' or 'integration' values. The menus a split into 'file' operations, 'edit' operations (changing the graph itself, copying and pasting), 'tools' for analysis (split further into VGA analysis tools, axial analysis tools and segment analysis tools), a 'view' menu to alter the way you view your analysis results, and a 'help' menu, which is perfunctory and contains little to no actual *help*. There is at least an 'online help' choice that points you to a webpage (it should open it for you) from which you can find this document. The 'toolbar' options are arranged from left to right in the order you will most often use them. On the left, it starts with a few file operations reproduced from the 'file' menu ('new', 'import', 'open', 'save'). These are then followed by a few navigation modes, before moving onto tools you will need to make a VGA graph or to generate an axial map. Finally, to the right of the toolbar, there are a few extra tools which are useful once you made a graph and you are ready to analyse it.

3 Before you begin: entering your user licence key

When you first run Depthmap, you will be asked both to accept the user licence agreement and to enter your user licence key. This key matches with a specific user name. You will need to enter both the key and your user name exactly as they have been supplied to you. If you do not have a user licence key at this stage, see the Depthmap webpage for details of how to obtain one:

`http://www.vr.ucl.ac.uk/depthmap/`

If in the future you need to update your licence for some reason (for example, to upgrade to an agent licence key), you can do this by choosing 'Update licence...' from the 'Help' menu.

Chapter 2

Preparing to Analyse a Plan

This chapter details how to import a plan for analysis into Depthmap, and basic manipulations that can be applied before analysis begins.

1 Importing a plan

To start, you need to create a Depthmap graph file, to do this, select 'new' from the file menu or the toolbar. You can then import a plan by selecting 'import' from the file menu or the toolbar. Note that you cannot open a graph file directly, the reason being that only graph files themselves may be open, saved or closed.

For a typical VGA analysis, the plans you import must be 2D vector drawings. The vector drawing was an early choice to ensure accurate visual information could be retrieved; whether or not accurate visual information is really required is open to question, but the vector import has remained. The supported import file types are currently[1] AutoCAD's DXF format (the importer was written for version 12, but it can import most 'ASCII' DXF plans), NTF format (this is the format for Ordnance Survey 'LandLine' data, which is easily available to universities within the UK), and something called CAT, which stands for Chiron and Alasdair Transfer Format. The DXF loader has gradually become more complex, but it is still best to import plans with only lines, polylines and polygons. Circles and curves slow analysis significantly and are only approximated within Depthmap. Neither the DXF loader nor the NTF loader import text, you are left with simply line information. However, any layers in your DXF or NTF are preserved, which will be of use later. The CAT format is very simple. It has just two sorts of data, polygon and polyline. The difference is that polygons are closed while polylines are open. A typical CAT file looks something like this:

```
#CAT
Begin Polygon
0.1 0.1
2.0 2.0
3.9 0.1
End Polygon
```

[1]Note, you can also import a MapInfo MIF/MID combination as an axial map. This will be dealt with separately in chapter 4.

As can be seen, writing a CAT file forms a simple way to prepare plans quickly. It would, of course, be quicker to prepare plans directly within Depthmap. Unfortunately, the facility is not included *yet*. One simple reason for the lack of inclusion is scale: it starts off as arbitrary until you have imported a drawing file. I find the CAD package scale setting unweildy, and would prefer a more intuitive method. Another reason is that undoing a line or polygon needs further consideration. It is not quite as simple as removing the drawn line from a list of lines.

You can import more than one file to analyse at a time. You will notice that you are asked either to add or replace the currently imported plans. Adding extra import files is especially useful for OS LandLine data where each NTF covers a 500m×500m tile, although conceiveably you might also want it for DXFs.

2 Using line layers

There are two sorts of layers in Depthmap. There are VGA layers, which will be covered in the next chapter (see page 19), and there are line layers. Line layers are the layers which were originally contained withing your drawing file, be it DXF or NTF (CAT files do not have layers). Depthmap gives you the option to turn these layers on or off. This will be useful later when performing VGA analysis; at the most basic level, you may well want to open doors within your plan to allow vision, and thus the visibility graph, to pass through them. To do so, go to the 'View' menu and select 'Line layer chooser...'. A small window opens, with a list of layers. Above the list is the name of the original file these layers were imported from. If you added more than one file you can flip between them using the arrow keys. Note that Depthmap is not intelligent enough *yet* to sort layers with the same name, but from separate files, into single combined line layers. So if you have loaded 16 NTF tiles and you want to turn off the road centre line layer for each of them, you will have to iterate through all 16 files. I apologise profusely for this, but as with most 'features' in Depthmap, it is one that I have suffered myself. 'Turning on' or 'turning off' a layer is simply a matter of clicking on the eye icon next to the layer's name. You will see there is also a pencil icon. For now, this does nothing. The intention should be fairly obvious: one day would like you to be able to edit layers directly within Depthmap. There is another (hidden) feature on the line layer chooser. If you click on the *name* of the layer, then you will be presented with a dialog box to change the colour of the lines in that layer. However, another idiosyncrasy: note that you cannot return to the default foreground colour (ever) after you have changed it in this way (for details on changing the default foreground colour, see page 34).

3 Moving and zooming

After you have imported a drawing file several options on the toolbar become available. In particular, a set of three 'modes' are activated. For the time being, the select mode does nothing; however, the hand icon allows you to drag the drawing around, and the magnifying glass allows you to zoom in (and out) of the image. The functionality of the zoom tool is copied from Adobe Photoshop. By default you zoom in — you can either click once, or select a region to zoom into. If you want to zoom out, then hold down the 'Alt' key and click; alternatively, right click on the mouse rather than left clicking. The 'Alt' key choice is Adobe's, and it imitates the use of the 'Apple' key on an Apple Mac. In addition to the zoom and pan hands, you can use the arrow keys to pan across the drawing and the '+' and '-' keys to zoom in or out (if you use the '+' key on the main keyboard, you should not need to hold down the shift key, simply press the unshifted '=' key[2].).

4 Setting file properties

You may want to include some extra information about your Depthmap graph file. For example, the geographical location of the site contained within the plan. To do so, select 'Properties...' from the 'File' menu. You will be able to set the title, location and a short description. There are also a few fields that you cannot adjust. The author as well as the organisation is taken from your Depthmap licence entry. The creation date is the date you *first* created the file, and the version of Depthmap is the version *first* used to create it. There is an exception: this information is only available with Depthmap 4.06r onwards; therefore, if an earlier version of Depthmap has been used, or for any reason this information is not known, Depthmap inserts 'unknown' in their place. I realise that it seems inequitable to stop you from changing the author information; the aim is a little gentle persuasion to use properly licensed copies of Depthmap.

[2]On non-English keyboards this feature may not work properly

Chapter 3

Visibility Graph Analysis

You will now be able to import a 2D plan to analyse. The next steps are first to prepare a grid for the visibility graph point locations, then to make a visibility graph from these locations, and finally to analyse the graph.

1 Preparing the visibility graph grid

The first stage of a VGA analysis is to make the grid of point locations for the analysis. To do this, choose 'Set Grid...' from the 'Edit' menu or the toolbar. This allows you to set a grid spacing for your analysis. It also used to allow you to set an offset. For some reason I removed the offset although no doubt in some cases it might still be useful (e.g., comparing two slightly differently drawn maps of the same area). The default grid spacing is a *sensible* choice of spacing for a reasonably fast initial analysis on most machines. The choice is based on the maximum dimension of your input drawing. The system will allow you to choose from ten times this sensible choice to one tenth of it. Both of these extremes are fairly silly. The maximum only has about 3 or 4 grid squares, the minimum will typically have in the region of 1 000 000. Note that there is a theoretical maximum for Depthmap based on internal parameters — a grid 32 000 $\times$ 32 000 locations. After you have set the grid, a grid of lines will be displayed over your drawing plan (to turn the grid off, see page 37).

It is important at this stage to clear up a common misunderstanding: Depthmap analyses *point locations* not grid squares. Although you are shown a set of squares for ease of seeing the points, visibility is always assessed from the very centre of the grid square. You may fill these grid squares using the fill tool or the context fill tool (for details on context filling, see the dedicated section in chapter 3, page 17) to fill an open area with point locations, or the pencil tool to fill point locations one at a time. Each can be undone using 'Undo' from the 'Edit' menu or by pressing `Ctrl+Z`; in addition, you may right click with the pencil to clear a point. Note that the fill tool uses a 'flood fill' technique which is very accurate with respect to visibility rules — remember that although each grey square may seem to overlap a line, the very centre of that square, the point to be analysed, never crosses it. However, whilst the flood fill is very accurate, it is also somewhat cautious. It proceeds one grid square at a time, vertically, horizontally or diagonally from where it starts. If it cannot 'get through' a gap between lines because the centre of the next square is obscured, it stops. Thus, sometimes whole areas which you know to be visible are missed. Simply fill in these areas as well before you analyse. Some people are tempted to fill every gap they perceive with the pencil, but there is no need — although the fill is cautious, the visibility graph construction algorithm looks from centre point to centre point of all the locations, and sees across any gaps.

2 Making the visibility graph

Once you have a set of point locations, you are ready to make the visibility graph. Select 'Make visibility graph...' from the VGA tools menu. This connect each point location to every other point location that it can currently see. As already mentioned, 'gaps' in filled points are of no importance to the algorithm, all that matter are the vector lines of the plan drawing, and then, only the currently viewed lines of the plan drawing. This can be used to your advantage to analyse different types of accessibility of locations and then compare them. For example, in one analysis, you might analyse visibility at eye-level, and in another you might analyse visibility at knee-level (the associated visual fields at these locations are sometimes known as *knee*-sovists to contrast with *eye*-sovists). The knee-level analysis would be prepared by including layers for both eye-height obstructions (such as walls) and other objects that might block permeability (such as tables and chairs). In order to compare the two analyses easily, the eye-level analysis points would still be prepared as per the knee-level analysis, excluding the tables and chairs. However, before analysis, the table and chairs layer would be hidden, and the graph made.

When you choose 'Make visibility graph...' you are shown the option to make a 'boundary graph'. This feature is not yet completed, and should be greyed out. The idea is that objects on the edge of the graph may be attributed with values according to where they can be seen from. Note that after you make the graph some of the earlier options will now no longer be available. You will not be able to change the size of the VGA grid again, and the graph make process itself is irreversible. Make sure you have set up the graph exactly as you want it to be processed before you continue.

Depthmap processes the graph for a time depending on how many points you have made, and how well interconnected they are, before displaying some results on screen. Depthmap does not show the graph itself, as this is typically highly connected and near meaningless, but it does show a result from the construction of the graph: the connectivity of each node (how many locations each node can see). Some people confuse this with a more complex graph analysis, the integration, as it shows a rainbow range of colours on the screen. This is *not* the case. In order to analyse the graph more fully you must go on to choose 'Run visibility graph analysis' from the VGA tools menu.

For the time being you just have a small range of immediately available measures: the connectivity (displayed), isovist maximum radial and isovist moment of inertia. The word 'isovist' is a misnomer, as an isovist is technically the polygon visible from each node. In fact, the 'isovist' maximum radial calculated by Depthmap is the distance to the furthest visible point location from each node, and the isovist moment of inertia is calculated using the points rather than the isovist continuous isovist polygon. The moment of inertia, by the way, refers to how easy an object is to spin. The lower the moment of inertia, the more easily an object will spin. An ice-skater spinning will often start out with their arms and legs stretched out, which gives them a high moment of inertia and means they will spin slowly; they then bring there arms and legs in, lowering the moment of inertia and making them spin faster. Whether or not this is meaningful in the context of a spatial analysis is of course entirely open to question! Appendix A gives details of the calculations involved. To switch between the displayed attributes, the top of the graph window has two drop lists which become active once the graph construction is complete. The left-hand drop list, which I will call the layer selector, should simply say 'Visibility Graph Analysis', and allows you to switch between VGA layers (see section 5 of this chapter, page 19). The right-hand drop list lets you swap between the attributes available, and I will call it the attribute selector. You may change the attribute names if you want to by clicking on its name on the drop list and typing over it.

Tip: If you hold the mouse over a point location, Depthmap will display the value of the current attribute for that point location.

Once you have made the graph you are strongly advised to save the file, using 'Save' from the file menu or the toolbar (or `Ctrl+S`). On the subject of saving, one issue that crops up again and again is graph size. Can I make it any smaller? The answer is: yes, but not without slowing Depthmap down significantly. There are two concerns: the 'in memory' size of Depthmap with a graph (how much memory Depthmap uses when it has the graph open), and the size of the graph saved to disk. Both the 'in memory' size and the saved to disk graph are compressed in a number of ways, but are still typically large. In fact, they are both compressed in the same way. A graph file could be compressed further when it is saved to disk, but third party 'zip' programs specifically designed for compressing files will do the job better. So, if you want to make to make the saved graph size smaller, use a third party zip program to zip up the graph file (Windows XP is supplied with a zip program as an integral part of the operating system). As for 'in memory' size, it depends on how detailed you make your graph. I believe the real question is: how detailed does your analysis have to be? The mean depth of any system (see page 14) is suprisingly consistent across a range of scales. Try a few larger scale analyses first, and see how fast the mean depth of the whole system converges[1]. If your mean depth has converged significantly, then there is no need to continue to increase your file size. I realise that the wording of this paragraph is slightly vague — what, you may ask, does 'converged significantly' mean? The reason for this vagueness is the general problem of using a 'pixelated' system to represent a continuous whole. Like many other analyses, the convergence of mean depth is chaotic. As you increase detail, you might just get a resolution that sees through a specific gap that

[1]Unfortunately, seeing the average mean depth for the system used to be easy in Depthmap, but the feature has been disabled. See page 38 for how and why the status bar data has been modified, and page 39 for how to export data from Depthmap.

suddenly reduces the depth of your whole system by connecting to otherwise separated areas. Or even by shaving a corner here and a corner there, you might achieve the same result. So ‘convergence’ is a fuzzy issue. The bottom line is to be sensible, and to understand what you are trying to achieve with your analysis. In particular, read Turner et al. (2001) for a more detailed discussion of this problem.

One final problem that might occur is that very occassionally Depthmap may appear to see through walls. This is almost certainly not the case: the problem will most likely be that your plan drawing has very small gaps between pieces of built form. For example, one line does not quite meet another when they have been drawn. To avoid this problem you should use ‘snap to’ or similar on your CAD drawing package to make sure all lines, particularly where they meet at ‘T’-junctions, are properly joined to one another.

3 Finding step depth from a visibility graph node

One thing people often want to see is the ‘isovist’ from a point location. As described above, an isovist is the polygon which contains all the visible area from a particular location. *Depthmap does not calculate isovists.* It is a piece of graph analysis software, not a geometric analysis tool. However, you can see an approximation to an isovist if you choose to calculate ‘step depth’ from the current location. Every node one step away from the your selected point location is directly visible from that location, i.e., all those nodes are within the isovist of the point location. With a high density grid, this approximates to the isovist, although you will typically see ‘dotted’ rows emanating from a location where the visibility cone goes through a small gap. These occur because the point centres of these nodes are just visible through the gap, while those nearby, seemingly in a straight line from the gap, are just obscured.

To construct a step depth calculation, select a point (or range of points) using the arrow cursor. If you want you can even choose separate areas of points by holding down the `shift` key while you select points. Next choose ‘Step Depth’ from the VGA tools menu, or alternatively click on the ‘Step Depth’ tool on the toolbar, or press `Ctrl+D`. Depending on how large your graph is, this may take a little while. In order to make the approximate isovist more apparent, you might want to change the colour range displayed — see chapter 6, page 34. The step depth itself may be viewed as the number of turns (plus one) that it takes to get from the current location to any other location within the plan. Everything directly visible from the starting location is at depth one, everything visible from that at depth two, and so on throughout the plan. We call this the *visual depth* of each point. Note that the visual depth is a measure of the ‘shortest path’ through the graph. You could turn umpteen times to get to any location within the graph, but on the shortest path from one node to any other, you make as few turns as possible.

Tip: If you want to record multiple step depth calculations from many different locations, you can rename the ‘Visual Step Depth’ attribute. For example, you might change it so that it includes the point it was take from: 'Step Depth from 10,3' (to rename attribute columns, see page 11). Now when you next select the step depth tool, Depthmap will create a new 'Visual Step Depth' column rather than overwriting the old one.

In addition to the standard visual step depth, you can also choose to find the 'metric step depth'. Rather than using a step depth based on the number of visual turns from one location to another, the metric step depth uses a *weighted* version of the visibility graph. In this version, to move from one location to another costs not one step, but the metric distance from one location to another. If you select a point and choose this version of step depth, Depthmap will calculate the shortest path through the weighted visibility graph (difficult to follow in text, but simply in practice) to each other point location, and record this distance. This is recorded in the attribute called 'metric step shortest-path length'. It will also calculate the angle through which you have to turn to get to that point by that route, and record it in 'metric step shortest-path angle'. Note that the angular units used in Depthmap have changed recently: before Depthmap 4.06r the shortest-path angle was given in 'radians'; it now uses the same convention as Hillier and Iida (2004) use for angular analysis of segment maps, where 1 represents 90°, 2 180°, etc. (see page 26). The reason for the inclusion of the extra angular measure is Penn's (2003) study into how paths might be cognised by people. One further metric depth measure is available: if you select a single point to start from (rather than a selection of a set of points), Depthmap will also record the straight-line distance to each point from this origin (i.e., the crow's path distance), which may be useful to compare with the shortest metric path through the visibility graph (again see Penn, 2003).

It is imperative to realise that the metric measure works through connections within the visibility graph. Therefore, it will cross void space just as the visibility measures do. In order to force the metric path round objects you must include those objects when you make the visibility graph. Further still, some counter-intuitive results may appear around object, for example, the same 'dotted' points produced by the visual step depth also appear on the metric path, as these are directly on a line from the initial location, whereas to reach others, slight turns will be necessary. In rare cases, the metric path may even double-back around a corner to get to points it has bypassed. If you find this situation, think about the visibility graph behind the metric analysis to understand what is going on.

4 Analysing the visibility graph

Once you have made your visibility graph, you will be given the option to 'Run visibility graph analysis...' on the VGA tools menu. Selecting this option opens a dialog box with two further options currently available: to perform a visibility analysis of the graph or to perform a metric analysis of the graph. These two further options are described in the following sections, as well as a method of reducing the analysis time for visibility analysis, context filling. The chapter concludes with a section on how to transfer the results in summarised form to 'layers' within the visibility graph.

4.1 Visibility analysis

Within the visibility analysis you have the possibility to make both a 'global' analysis of the graph and a 'local' analysis of the graph. The global measures are all based on preparing shortest paths from each node, through the visibility graph, to all other nodes (see the discussion about shortest paths in the previous section). The local measures are based on the relationships between each node and the nodes directly connected to it. The reason the two are split is that different sorts of visibility graph may take much longer to analyse for either global or local measures. For example, if the graph is compact with lots of open space (e.g., an open plan building), then the local measures will be slow while the global measures will be faster. Conversely, if the graph is of a street network, where each node has relatively few connections, the global measures will be slow while the local measures will be faster. Many people simply check both boxes, but, before you start, ask yourself if you really need all the analyses provided.

After the measures have been calculated, each measure is given a name beginning 'Visual'. This is to distinguish VGA measures from both metric and axial measures, which are calculated later. The global measures of the graph include 'mean depth', 'node count', 'integration' and 'entropy'. As with the other attribute columns, you can switch between them using the attribute-selector drop list at the top of the graph window, and change the names if you desire by clicking on the drop list name.

Mean depth is calculated for each node much like the step depth. The shortest path (i.e., the fewest number of turns) through the visibility graph is calculated to each other node within the graph. These are summed and divided through by the number of nodes in the graph (minus the node we are considering). Note that if the graph is divided into two separate sections which cannot see each other, then the number of nodes is simply the number in *that section* of the graph. The count of nodes itself is put in the 'node count' column.

'Integration' is a measure which has seen much mention in the space syntax literature; it is defined in Hillier and Hanson (1984). The measure is essentially a normalised version of the mean depth, and it is important because it has been found to correlate well with pedestrian movement 'gate' counts, as remarked in the introduction (Hillier et al., 1993). 'Normalisation' is a procedure to make different systems comparable with one another, typically by forcing a value in the range 0–1. Hillier and Hanson do this first, to produce what they call the *relativised assymetry*. However, they then divide through by a number called the d-value (see appendix A for the calculation), which is meant to cater for the fact that as axial map graphs grow, they also become less integrated due to the way the lines intersect. Thus, a small system always looks more integrated than a large one. Depthmap simply takes the d-value and blindly applies it to VGA graphs, the result is called ‘Integration (HH)’. The application of a normalisation designed for completely different circumstances may seem a little naïve, but the reasoning was that at least the measure is clearly defined, and if you say integration for VGA or axial analysis, then you know you are talking about the same quantity. de Arruda Campos and Fong (2003) examine the application of the d-value to VGA graphs in more detail and suggest that instead a normalisation using a different number proposed by Hillier and Hanson, the p-value may be more appropriate for VGA graphs. Although Depthmap does not yet normalise by p-value, it does provide another integration variant proposed by Teklenburg et al. (1993), which it labels ‘Integration (Tekl)’. Teklenburg et al. were also interested in the normalisation of axial maps, but the scaling they use is more generic, and certainly simpler, using a logarithmic scale. Not enough work has been done at the time of writing to see if this method is a more appropriate normalisation, or, indeed, if any normalisation of depth is a sensible idea for such graphs.

There is a final pair of global measures calculated by Depthmap, called ‘entropy’ and ‘relativised entropy’. These are described in Turner (2001*b*). The reason for their introduction was that Depthmap seems to prioritise open spaces or wide streets in terms of integration, where as people seem to cognise space on a much more one-dimensional level (see, for example Penn, 2001; Kuipers et al., 2003). A measure of entropy is a measure of the *distribution* of locations in terms of their visual depth from a node rather than the depth itself. So, if many locations are visually close to a node, the visual depth from that node is assymetric, and the entropy is low. If the visual depth is more evenly distributed, the entropy is higher. Relativised entropy takes account of the *expected* distribution from the node. That is, in most cases, you would expect the number of nodes encountered as you move through the graph to increase up to the mean depth and then decline afterwards. The technicalities of entropy and relativised entropy, which are based on Shannon’s measure of entropy and information theory, are described in appendix A. Within the text here, it is worth mentioning that there is a spatial interpretation problem for both entropy and relativised entropy, and that is that entropy is low for uneven distributions, whether these are close *or* far away from the current node. That is, a doorway which suddenly opens onto a street, and a set of streets that finally opens onto a few doorways may well have the same entropy, while they would see very different usage.

For global measures there is also the option to set a ‘radius’. The radius corresponds to the radius commonly used in axial analysis. If it is set to the default, ‘n’, then the global analysis is applied as described above. If the radius is instead set to an integer number (1,2,3 etc.) then the global analysis is restricted to path lengths up to that number. So, for radius 2, the mean depth is calculated by summing the length of the paths to all nodes one visual step away (simply the number of nodes it can see), and the length of the paths to all the nodes one step from there (i.e., all the nodes at step depth two multiplied by two); Depthmap then stops, as all paths up to length two have been discovered, and the total is divided by the number of nodes encountered (all those at step depth one, and all those at step depth two) to give the mean depth. You will note that Depthmap appends 'r2' to the name of the measures for radius 2 (or 'r3' or 'r4' for radii 3 and 4) in order to distinguish them from the radius n measures.

The local measures calculated by Depthmap are 'clustering coefficient', 'control', and 'controllability'. Clustering coefficient is a measure introduced by Watts and Strogatz (1998) to help assess whether or not a graph is a 'small world' or not. A 'small world' is one that has tightly clustered groups of friends (though Watts and Strogatz's point was they may be much more generally applicable than just to people), but surprisingly low mean depth, that is, the number of steps to get from one person (e.g., a particular Australian sheep-herder) to any other person (including every Innuit as well as every Mexican) through links of mutual friends is surprising low. What, you are probably thinking, has this got to do with visibility graphs of spatial systems? The answer is that the clustering coefficient appears to give an idea of the 'junctionness' of locations[2], and how the visual information is changing within systems, dictating, perhaps, the way a journey is perceived and where the decision points come within it (Turner et al., 2001). There are, however, caveats and the reader should refer to Llobera (2003) for arguments against this interpretation. Notwithstanding Llobera's contentions, Doxa (2001) provides some examples of how the clustering coefficient might be used to help understand the use of public buildings.

Control is a measure which again comes from Hillier and Hanson (1984), while controllability is a measure proposed in Turner (2001*b*). As the names suggest, control picks out visually dominant areas, whereas controllability picks out areas that may be easily visually dominated. For control, each location is first assigned an index of how much it can see, the reciprocal of its connectivity. Then, for each point, these indices are summed for all the locations it can see. As should be obvious, if a location has a large visual field will pick up a lot of points to sum, so initially it might seem controlling. However, if the locations it can see *also* have large visual fields, they will contribute very little to the value of control. So, in order to be controlling, a point must see a large number of spaces, but these spaces should each see relatively little. The perfect example of a controlling location is the location at the centre of Bentham's panopticon. This point can see into every cell, but the spaces inside each cell can see relatively little (back out to the centre and perhaps across it, but no more). By contrast, the cells themselves lack control, as they can see relatively little, but the locations they link to (in the centre) link out to many locations. Controllability is much easier to describe: for a location it is simply the ratio of the total number of nodes up to radius 2 to the connectivity (i.e., the total number of nodes at radius 1). Applied to the panopticon example, it would seem to operate in a similar manner to control. Each of the cells is highly controllable, as the area of visual field is small compared to the area viewable from the centre to which it connects, while the centre is less controllable, as it links only to the cells within its field, and they add little extra visual field. However, the panopticon is, of course, a contrived example. In reality, some spaces can be both con-

[2]A more naïve answer is: to determine if building or urban visibility graphs small worlds. O'Sullivan and Turner (2001) discuss why it is futile to attempt to classify landscape visibility graphs as small worlds or not. Similar arguments hold for visibility graphs of urban and building systems.

trollable and controlling, and others uncontrollable and uncontrolling. It would seem an interesting avenue of research to see if anything further can be made from these measures — for example, to look at locations where crimes are committed, where a robber will want to control the victim, but at the same time be uncontrollable by the forces of law and order.

4.2 Reducing the analysis time by context filling

The problem with analysis of a large visibility graph is that it takes a long time. This may be especially inconvenient if what you are really interested in only forms a small part of the graph. For example, you want to analyse the plan of a building, but you want to put the building in the context of the outside urban grid. In order to speed up the analysis somewhat and retain a good analysis of the inside space, Depthmap allows you to fill 'context' space. To do so, you would first fill your building (and only your building) with standard, normal, points. This is a little tricky, as the points will flood out of the building into the open space. You might handle this in one of two ways: (a) have a separate doors layer that you can turn off before you analyse (remember, the purpose of this section is to analyse both the outside and inside of a building *together*), or (b) block the entrances using the pencil tool (be careful only to fill points where the centre location is in open space) and then fill within the building, as per normal, using the 'fill' tool. Now that you have the area you want to analyse filled with normal points, you can fill the outside area with context points. Select the 'context fill' tool from the toolbar and fill. The context points will have a bluish tinge whereas the ordinary points are grey.

Once the grid is prepared and filled you can make the visibility graph as normal (see page 10). At first you will see no difference, all the points will have connectivity values filled in, regardless of whether or not they are normally or context filled. It is only when you perform an analysis of the graph that you will see a difference. Depthmap only analyses 'evenly numbered' context points (i.e., only 25% of the context points). The alert reader will notice that this will have serious implications for the integration analysis: Depthmap is effectively combining two graphs at different scales. In order to prevent any problems, Depthmap in fact still calculates mean depth to all points in the graph, be they normally filled or context points. The only difference is that only evenly numbered context points are used to calculate onward paths. The algorithm is speeded up signficantly due to the way Depthmap calculates mean depth using what is called a 'breadth-first search', and the amount of time this algorithm takes to work is called '$O(n^2)$'. That is, as you increase the number of points to analyse, the time Depthmap takes to calculate global graph measures increases according to the number of points *squared*. Using only a quarter of points for onward search to other points reduces this to about $n \times n/4$. Further still, since, for most analyses you will perform, most points are context points, and only 25% of them have their mean depth calculated reduces the time still further — to about $n/4 \times n/4 = n^2/16$ — that is, a near sixteen-fold increase in efficiency.

The use of context fill does lead to some bizarre quirks in Depthmap. On the analysis side, because only 25% of points are used in the context area, Depthmap sometimes does not continue an analysis through a gap between buildings where you might expect it to (simply because the context graph is much rougher than the full analysis graph). Therefore, it is sometimes useful to try a step depth calculation (see page 12) to check that all areas of the context map may be reached. A second quirk is that when you select one area and then try to select another, 75% of the points (the unanalysed points) remain highlighted. This is an unfortunate problem with Depthmap's redraw algorithm. Because Depthmap does not think the points are analysed (they have no attribute values), it thinks they cannot be filled with points, and therefore, that they cannot be selected. Do not worry. In fact, Depthmap *has* made the new selection, it has just failed to redraw properly. You may also find this problem with other analysis types — whenever a point does not have an attribute value assigned to it (e.g., if you try a step depth calculation where two systems are unconnected). It is on a list of things to sort out.

4.3 Metric analysis

As well as visual analysis of points, Depthmap may also make 'metric' analyses of points. It prepares metric analyses in exactly the same way as it prepares indivual metric step depth values (see 13). You should also note once more the intricacies of how these depths are calculated. Remember that the metric calculations are based on the underlying *visibility* graph, so they pass through 'void' space just as the visibility graph does, and that it may be necessary to 'double back' through the visibility graph to get to some points. Also remember that from Depthmap 4.06r onwards angles are recorded in the range $0 - 4$ for $0 - 360°$ (see page 26) rather than in radians.

The metric analysis is as follows: for each point, Depthmap finds the shortest metric path through the visibility graph to each other location. It sums these and finds the mean. It also finds the mean angular deviation encountered on each of these paths, as well as the mean straight-line or crow's fly distance to other points. Each of these analyses is preceded with the word 'metric' in the attributes table, so it is easy to distinguish them from the visibility analyses of the graph. The metric mean shortest-path distance (or more simply, metric mean depth) does not ever seem to produce results that are particularly interesting — it focuses on the centre of the system. Of more interest to analysts would probably be 'metric choice', that is, the number of times a location is encountered on a path from origin to destination. This as yet is not recorded, as the calculation becomes extremely complicated, however, it appears metric choice might actually give some indication of pedestrian movement flows, if only in segment graphs (Hillier and Iida, 2004). Another possibily interesting measure is the angular-deviation on the metric shortest-path routes. As discussed when describing metric step depth, Penn (2003) suggested this measure, and showed how it might vary for individual routes. It still requires further investigation and theoretical input as to whether or not it provides a meaningful average measure for the system.

Before leaving metric analysis, it is worth pointing out that it does encounter difficulties if you decide to 'merge' between floors (see chapter 6, page 32). If you jump from one area of a graph to another, both angular and straight-line distances will become confused. The straight-line distance is just calculated from the geometric graph, while the angular distance is logged at the jump, and then only incremented when you again turn a corner away from the new, merged, node (that is, all of the merged nodes connections are considered as being at angle zero from the merged node).

5 Visibilty Graph Layers

One feature we have not used yet is the left-hand drop list at the top of the graph window, the 'layer selector'. If you click on it you will see that as well as 'Visibility Graph Analysis' you can view 'Gates', 'Attractors', 'Generators' and 'Boundaries'. Of these, only the gates are fully implemented at this stage, and I will describe its use in detail below.

Attractors and generators are really intended as supplementary layers for the agent plug-in, and their inclusion is a hangover from the integrated agent plug-in. In future, I plan that plug-in modules should make their own layers if they require them, allowing modules to add as many layers as they need and freeing them from the restrictions of my naming conventions (see appendix C for details about how to code your own plug-in modules). The boundaries layer is slightly different in that it shows nodes on the edge of the visibility graph which intersect built form. The boundaries layer is intended to be used alongside the 'make boundary graph' feature (page 10) when it is finished.

The 'gates' layer is designed to facilitate the comparison of visibility graph values with values you might have collected from observed pedestrian behaviour in the corresponding physical location. The name implies a specific kind of space syntax observation, a 'gate' count, where the numbers of pedestrians passing through a notional 'gate' (typically, an observer placed on the pavement), although the layer could be used for counts of anything: stationary people within an area, people in conversation, and so on. To use it, first make 'gates'. Switch to the gates layer view, and select a region of points that you wish to comprise the layer. For my own experiments, I usually select a fairly generous region of points around the gate (Turner, 2003). To make the gate, either select 'Add Layer Object' from the edit menu or the toolbar, or press `Ctrl+G`. Note that you cannot undo 'Add Layer Object' as yet, so be careful to select the correct area before pressing `Ctrl+G` and save frequently. By default, the layer viewer will show an ID number column for the layer objects within it. Provided your drawing plan is displayed in enough detail, a text ID number will appear at the centre of your gate (if not, zoom in further to see the text; see page 37 to turn the text off).

Once you have made some 'gate' objects you can transfer VGA attribute values to them. Switch back to viewing the VGA and select 'Push Values to Layer' (again, either from the edit menu or the toolbar). If you now view the gates layer again, you will see that whichever VGA attribute value you were viewing has been transferred to the gates layer. Each layer object takes the *average* value of the points that comprise it for the VGA attribute you were viewing. Note, in the future I would also like it to be able to transfer the maximum or minimum values to the layer.

In addition you can change the attribute column names for the layer as usual, but further, if you wish, you can add new attribute columns (for example, to record data from your own observations of the area) and also delete them by using the 'Add Column' and 'Remove Column' buttons to the right of the attribute-selector drop list (note that you cannot remove or rename the ID number column). There then remains the question: how do you enter your own attribute values into the new attribute column? This is dealt with in chapter 6, in the section on showing the text and tables (page 37). You can even use this facility to change the values of the attributes you pushed into the layer from the VGA analysis.

Chapter 4

Axial Lines: Axial Map Generation and Analysis

Generating unique axial maps, as originally described by Hillier and Hanson (1984), is impossible. Each researcher will inevitably draw slightly different maps, as will a computer program. However, the graphs of the lines produced are remarkably consistent, and now it seems that a 'greedy' algorithm will soon draw identical graphs to those that researchers draw (Turner et al., 2004). So it might seem unnecessary to start this chapter with a discussion of how to load hand-drawn maps. It seems unnecessary, but it is not. In order to get a the computer program, Depthmap in this case, to draw a good axial map, you have to produce a good plan for it to work from. You have to decide whether or not to include street furniture, how to work around roundabouts, and so on and so forth *before* you analyse your system. You might argue that we should be able to make these decisions upfront, but how do we sketch new designs, change old ones, work according to what we know is the true situation rather than what the cartographer, viewing from 1km up, has drawn? The hand-drawn axial map has mileage yet, and you will probably find it easiest to draw first and then look into automatic generation later. It is also important to know that Depthmap has both a slow and a buggy implementation of the algorithm I describe with Hillier and Penn, because it suffers from what is called a 'floating-point error' (see section 3 for details). I hope both these issues will be resolved soon, but for now, the hand-drawn map remains superior. After demonstrating how to load a hand-drawn map, this chapter devotes two sections to automatic line generation, first all-line maps, and then fewest-line maps, before turning to the analysis of all these types of axial maps.

1 Loading a hand-drawn axial map

There are two ways to import a hand-drawn axial map into Depthmap. The first is to import a NTF, DXF or CAT as usual and then to 'convert' it to an axial map. To do this import the axial map drawn in one of these formats and select 'Make Axial Map from Line Layers' from the axial tools menu. All the lines from visible line layers will be included in the axial map (if you really want to, you can even convert a building plan to an axial map). As with making a visibility graph, the axial map is coloured up according to connectivity (the number of lines each line intersects with) as soon as the graph is made. This *does not* mean an axial analysis has been performed. In order to analyse the axial map you will need to run axial line anaylsis, see ahead to section 4 of this chapter (page 25.

Tip: As with visibility graph points, you can view the value of the currently displayed attribute by holding your mouse over a line and leaving it still for a moment. Sometimes this is difficult for short lines: however you place the mouse it just will not show the value of the attribute for the line. If you are having a problem, zoom in.

Sometimes, if you are working fast, you may accidently select the 'Make Axial Map from Line Layers' option again. An awful thing happens: the axial map disappears! What has happened is that after Depthmap makes an axial map, it *hides* all the active line layers automatically, so that you can see the axial map lines you have just constructed beneath them. If you try to make an axial map *again* then there will be no line layers displayed, and Depthmap will make another, blank, axial map. I should really enable a check to see that you have some line layers displayed, but unfortunately, I have not completed this yet. To get back to the old map, use the layer-selector drop list at the top left of the graph window. You will have created two 'User Defined Maps'; select the first one to see your original map once more. There is no delete facility for axial maps, so you will be left with an extra blank map.

The second way to create a hand-drawn map is to import one directly from MapInfo. MapInfo can export MIF/MID file combinations, that Depthmap can import as axial maps. To do so, create a new graph file and select import. In the import dialog box, change the file type from 'Drawing files' to 'MapInfo axial map'. Now load the MapInfo file. It may take a little while, as it connects your file into an axial map directly, without creating line layers. As with the DXF, NTF or CAT loaded map, it is not yet analysed, and merely shows connectivity. It is given the name 'MapInfo Map' in the layer-selector drop list.

MapInfo MIF/MID file combinations have a coordinate system included. The coordinate system may often be 'non-Earth', i.e., flat, but in some cases it may be an 'Earth' coordinate system, i.e., the lines it describes lie on the surface of a sphere. Note that Depthmap *ignores* the coordinate system and just draws the lines as though they are on a flat surface (although it does actually retain the coordinate system information, and if you later export the lines as a MIF/MID file combination, the original coordinate system will be saved with the file).

2 Generating an all-line map

The alternative to drawing an axial map by hand is to generate an axial map from your open space geometry. This is done in two stages: first, generate an 'all-line' axial map, and second, reduce this to a 'fewest-line' axial map (see section 3 of this chapter). The fewest-line map is what tends to be drawn by researchers, the all-line map is a first stage. However, it is a map that might be considered interesting in its own right (Penn et al., 1997), and it can be analysed just as any other axial map can be. To generate the map, you must show Depthmap where the open space is (much like you must also implicitly show Depthmap the open space for a VGA analysis). To do so, select the

'Axial Map' tool next to the 'Set Grid' on the toolbar, and then click in open space. The all-line map will be made in three stages. A preprocessing stage tidies up any loose ends of lines in your map. The processing stage then goes through connecting all pairs of intervisible corners within the space, as per Penn et al.'s algorithm. If both corners are convex, a line is added between them, if one is reflex or both are reflex, then a line is only added if it can be extended past the reflex corners. The final postprocessing stage connects the lines into an (unanalysed as per usual) axial map. The map is called 'All-Line Map' within the view-selector drop list.

Sometimes you may find that you wait, and then no all-line map appears. There are couple of problems that can occur: firstly, Depthmap may not find a first corner from which to start the axial map. I am not sure why this happens, and it happens so rarely as not to be a problem. If this is the case, then you should be able to click elsewhere and the map will be generated (note that you will need to reselect the 'Axial Map' tool as it is automatically deselected in order to prevent you accidently clicking it again). The second problem is more difficult to solve: Depthmap needs a system where the corners are properly connected to make the axial map. Tiny gaps, too small to see on most software will cause Depthmap to think there are no convex or reflex corners, only ends of lines, and it will fail to produce an axial map. As with performing a VGA analysis, you *must* ensure that all corners of buildings and walls are properly connected in the CAD software you use to draw your plans, using a facility such as 'snap to' or similar.

Note that your spatial system does not have to be bounded, it can consist of polygons that open out, like a real urban system, rather than enclosing all the space within them. If not, Depthmap puts a margin around the system — a horizontal margin of 25% of the original width and vertical margin of 25% of the original height is added — and any axial lines protruding from this expanded region are cropped to fit within it (as otherwise they would head off to infinity). Note that the size of this margin may have implications for which lines within your system will intersect.

On a large system, all-line map generation can be extremely slow. This is because it checks every corner to every corner in the system to find out if they can see each other. Although an obvious algorithm to use, this is another algorithm which takes $O(n^2)$ time to complete. There are other, faster, algorithms to check the intervisibility of points (for example, the one that creates the Depthmap visibility graph), but as the all-line generation code was only recently developed as a proof of concept, I have not had time to switch to a more efficient algorithm as yet.

3 Reducing the all-line map to the fewest-line map

As with the code to generate an all-line map, the code to reduce an all-line map to a fewest-line map is essentially a proof of concept. With Hillier and Penn, I was trying to prove that it *could* be done, rather making a system where it *would* be done. I still believe that drawing axial maps by hand is preferrable for real uses; however, Depthmap contains the code to prove that you can draw an axial map analytically. There are two sorts of fewest-line map generated by Depthmap. If you select ‘Reduce to Fewest Line Map’ from the axial tools menu, both are generated for you (it causes little processing overhead to make both at the same time). As with making the all-line map, there are several phases. Preprocessing constructs some visual and topological relationships between elements within the space, while processing checks to see which of these lines fulfil these visual and topological relationships. In fact, processing try to find the minimum number of lines that fulfil the relationships. The relationships are that all parts of the system must be visible from the set of remaining axial lines, and all islands of built form must be contained within the set of remaining axial lines (Turner et al., 2004). The processing code is about the most difficult to predict in terms of the time it will take of any analysis in Depthmap. You will often have seen the ‘Please Wait’ dialog by now, with an estimated time to completion. Note that the time to completion will often rise quite dramatically within the processing phase.

The two results that Depthmap produces are the ‘Fewest-Line Map (Subsets)’ and ‘Fewest-Line Map (Minimal)’. The subsets version, which uses ‘subset reduction’ to remove lines with subsets of connections of other lines, is guaranteed to produce a unique axial graph for any configuration, while the minimal version, which removes lines in an attempt to get the fewest lines that both complete all islands and see all of the system, is not guaranteed to make the same reduction as someone else’s program might (for details, see Turner et al., 2004).

The map is slightly different from the graph. In the map, you must draw the exact same line, for example, in a road, you might draw two crossing lines and choose to remove one or the other. So does Depthmap, but in certain situations, this choice is arbitrary for both subset reduction and minimal fewest-line maps. This is not a very big problem. In fact, the graph for the minimal version is also practically unique, and you should be able to use either for most analysis — so long as you record which you use, and use it consistently, there should be no scientific problem with your decision. The minimal tends to look much more like the sort of map a researcher would tend to draw, while the subset version is technically the more correct (Turner et al., 2004).

There is one slight difference between hand-drawn axial maps and generated maps. People draw axial lines going into long dead ends, while the algorithm in Depthmap deletes these lines (and often draws just a tokenary line into other convex areas, which does not cross the space as a line drawn by a person would). Hillier and Penn (2004) describe a set of lines which pass through the longest dimension of convex spaces as a set that forms strong coverage; we have not thought of an algorithm which reproduces them elegantly and consistently in all cases. It is an outstanding question, and left open to the reader to find an answer.

There is also a bug in Depthmap’s algorithm. It is called a ‘floating-point error’, and occurs when the computer approximates a number to a certain number of decimal places. This particular error leads to Depthmap sometimes thinking it has not isolated all islands from each other when in fact it has. In these cases it manages to draw an *extra* line that it does not require in the fewest-line axial map. I have not narrowed down exactly which circumstances lead to this error yet. It is extremely rare, but you should be warned that it does happen.

4 Analysing the axial map

Before you analyse the axial map, two attributes, connectivity and line length will be available. Two types of analysis may be performed on axial maps to obtain further attributes: the step depth from a line (or set of lines) and a global integration analysis.

To calculate the step depth, first select a line with the arrow cursor. To select more than one line, hold down shift while clicking. In some cases, just as with holding the mouse over the attribute value, short lines may be difficult to select. Once more, zoom in if this is the case. For the time being, selecting ranges of lines by dragging the select cursor have not been enabled. When the line or lines have been selected, choose 'Step depth' from the axial tools menu or the toolbar. The step depth, how far each line is away from the root line (or lines) in terms of the number of changes of line, will be displayed. Note that the root line is at step depth '0', the lines connected directly to it are

You may also calculate global integration measures. To do so, select 'Run Axial Line Analysis...' from the axial tools menu. There is just one option: to reduce the radius. Just as with VGA analysis, the default 'n' will calculate measures to all lines reachable from each starting line. The measures also mirror the VGA measures: entropy, integration (both 'HH' for Hillier and Hanson d-value normalisation, and 'Tekl' for Teklenburg et al.'s normalisation), mean depth and node count. In addition, there is a measure called 'intensity', which was invented by Hoon Park. Intensity is the entropy relativised by the number of nodes in the system (see appendix A for full details). Park thought it would distinguish between 'local centres' in systems, and is currently working further on the theory. For further information about the other measures, please refer back to chapter 3. Local measures (such as control and controllability) are not yet included for axial maps, and clustering coefficient requires further consideration. In its current form, clustering coefficient is unlikely to be useful for analysis of axial lines, as 'clusters' of interconnected streets rarely exist in a typical deformed grid. However, Jiang and Claramunt (2004) suggest an interesting measure which may tell us something about how tightly structured a grid of lines is — they call it 'k-clustering' — effectively a form of 'radius 2' clustering coefficient.

Chapter 5

Segment Analysis

In order to make a segment analysis, you will need to have first created an axial map by whatever method (see chapter 4). Depthmap breaks this axial map into segments and connects it together as a network. Section 1 of this chapter describes how to make this network of segments, or *segment graph*.

The segement graph is relatively uninteresting from the local point of view — segments are linked typically across junctions, so that, in general, their connectivity is just 3 or 6 — however, when the segments are examined globally according to the average amount of turning it takes to get to any other line within the system (an *angular analysis* of the segment graph) urban environments take on results much like integration analysis, but on a finer scale.

Note that although you may think of segment analysis and angular analysis as interchangeable descriptions, they are not. Depthmap calls any analysis of a segment graph 'segment analysis'. Angular analysis is just one segment analysis that may be performed, just as a segment graph is one form of graph that angular analysis can be applied to. Depthmap also includes one other segment analysis, 'tulip analysis', and in the future it is intended that some form of metric analysis of segment graphs should be included. The angular analysis and tulip analysis of segment graphs are described in the second section of this chapter.

The way angular connections are made in the segment graph follows Hillier and Iida (2004). If a segment carries straight on to another segment then the angular turn costs 0 steps. If the segment turns 90° to get to another the segment, the angular turn costs 1 step. Finally, if the segment has to double back in the opposite direction (i.e., a turn of 180°), the angular turn cost 2 steps. In angular analysis, an intermediate turn is interpolated on this scale from 0 to 2. The 'cost' of turning a certain amount is called a 'weight' in graph theory: each connection has an associated weight, and the segment graph is called a weighted graph.

1 Making the segment map from an axial map

Use the layer-selector drop list at the top left of the graph window to choose the axial map that you would like to make into a segment map. From the segment tools menu select 'Make Segment Map from Axial Map...'. You are given two options in the accompanying dialog box. ‘Do not remove axial stubs’ means that the segmenter algorithm will simply chop up the axial map into segments as you might expect and connects them into a network. The second option ‘Remove axial stubs less than 25% of line’ is introduced due to a facet of hand-drawn axial maps (and also now generated maps). In order to ensure that the connection between axial lines is made, researchers join lines together with their ends overlapping slightly. Depthmap calls these ends ‘axial stubs’. The problem with them is that they cause an unnecessary angular turn to be introduced away from the rest of the system; that is, so if you are finding the step depth from a line, you always have to make an additional turn at the edge of the network of segments down one of these axial stubs. Although no work has been done on the differences between analysis with and without stubs as yet, the facility is there should removal prove useful. The stub removal version of the algorithm chops of any ‘stub’ detected to be less than 25% of the length of the original axial line (the reason for this cut-off is so that lines which are deliberately drawn as extending into closed convex polygons are retained). The cut-off for the length of stub you choose to remove can be anything between 1 and 50%. A few tests have shown that anywhere up to about 33% still chops line. Once the segment map creation is complete, it will appear in the list of axial maps in the layer-selector drop list.

As with the VGA and axial analysis, once you have made the segment graph, a connectivity attribute is displayed. Because segment graphs are connected through both physical connections and angular turns to other lines, you are given two options for connectivity, ‘Connectivity’ (the straight-forward connectivity as per axial analysis) and ‘Angular Connectivity’ (the cummulative turn angle to other lines). The standard connectivity is shown by default, and two other attributes are recorded: the ‘Axial Line ID’ and the ‘Segment Length’. The axial line ID records Depthmap’s internal reference number for the axial line that the segment was generated. You may want this when you export values from the segment map to compare axial and segment analyses (see the section on exporting data in chapter 7, page 39). Segment length should, I hope, be self-explanatory.

If you have started from an ordinary hand-drawn or generated axial map, and you have chosen not to remove axial stubs, both the connectivity attributes should look the same and equally uninteresting. Standard connectivities for axial stubs are 3 (at the end of the stub, carry straight on, turn left, or turn right) and for intermediate segments are 6 (straight on for both ends, and turn left and right for both ends). Similarly, the angular connectivities should be 2 for an axial stub: if carry straight on costs 0, and according to basic geometry, if a line intersects another, the sum of corresponding angles (the left and the right turns) comes to 180°, or a cummulative weight of 2. For the same reasons, the angular connectivity of any intermediate segment should be 4. If however you chose the removal of axial stubs, these rules are broken at the edge of the network of segments, and so, while the centre of the network should remain constant with a cummulative angle of 2, the edges will have varying connectivities.

Tip: You can make a road centre-line segment map by first loading in road centre lines from NTF. Make an axial map from the road centre-lines (see page 21), and then choose 'Make Segment Map from Axial Map...' from the segment tools menu.

Before moving on to the analysis of the segment graph, it is worth noting why you need to go through the stage of an axial map. Why not go directly from a DXF (or particularly a road centre-line graph) to a segment map? The answer is that I wanted you to be able to *unlink* certain segments from others easily. See page 33 for full details of how to unlink axial lines; for now, note that if a road bridge crosses another road, the road-centre lines or axial lines will coincide on the map. In order to represent the situation as available to the user of the environment at ground level, you will need to unlink them, symbolising the fact that they cannot move from one of the lines to the other. This is less easy to do for segments than for axial lines. For segments, you would have to unlink between four segments that join to the ends of each other. In the case of the axial or road-centre line, you simply have to unlink a single line from another single line. Hence, you are asked to unlink in the axial map, and then convert to a segment map. *You cannot link and unlink the segment map itself.* Instead, when you make a segment map from an axial map, the unlinks are preserved. In fact, it also tries to preserve links. I have not tried it but I suspect this will cause a crash on many systems, as the resolution it finds is to try to extend linked lines until they cross each other to find the angle between them.

2 Analysing the segment map

As with VGA and axial analysis, you may perform a step depth calculation for a segment map. Select a segment with the arrow cursor (the same provisos for line selection apply as with ordinary axial maps — if you find it difficult to select the segment, zoom in), and then select 'Step Depth' from the segment analysis tools or the tool bar (or press `Ctrl+D`). Since the purpose of generating a segment map is to perform angular analyses, the step depth calculation used is based on the weighted segment graph, and creates an attribute called 'Angular Step Depth'. The step depth follows the shortest angular path from the selected segment to all other segments within the system, and the angular path length is recorded on the line (if there are two unconnected systems within the segment map, this does not matter, the unconnected system will simply not be shown). Remember that in the angular scale used, '1 step' is 90°, and that angles are cummulative. Thus, if you turn left 45° and walk for a segment length then turn *right* 45° you have made two turns of 45° totalling 90°, exactly the same as if you had made on 90° left or right turn.

Currently just two further segment graph analyses may be performed: angular analysis and tulip analysis. To access them select 'Run Segment Analysis...' from the segment tools menu. You will be presented with a dialog in which you can choose 'Standard Angular' analysis or 'Tulip Analysis'.

Standard angular analysis works out the cummulative angle to all segments within the segment map and gives three new attribute columns: 'Angular Mean Depth', 'Angular

Node Count' and 'Angular Total Depth'. The node count is the number of segments encountered on the route from the current segment to all others. The total angular depth is the cummulative total of the shortest angular paths to them, while the mean depth divides this through by the angular node count.

There is an issue that I would like to mention here: whether or not the angular mean depth is a good relativisation or not. I would argue (and have argued) that it is not (Turner, 2001*a*). It is not good because if you chop a segment in half (for example, in the link / unlink problem we were discussing) then the angular mean depth *changes*. My argument follows that the very reason to introduce segments is to solve the segment problem. If you then say that different segmentations with the same angular path length are relativised to have different values then, it seems to me, that this defeats our original purpose. One of the advantages of angular analysis is that I believe it will bring space syntax into line with more conventional analyses of space, by allowing switches of representation. The two that I have argued seem most promising are road-centre lines and 'medial axis transforms' (Turner, 2003). I return to this topic in the conclusion as a matter for further research. However, notice that if the methodological representation in a road-centre line map tends to smooth corners through many segments and in the axial map representation it does not, then it is better to use no relativisation (and simply look at total angular turn) than to use angular mean depth. In Turner (2001*a*) I suggested relativisation by the total weight of all the angular connections in the system. I am not sure if this is any more helpful really than not relativising at all, although it would be useful to do further research in this area.

The second option on the dialog box is 'Tulip Analysis'. As described in the introduction, tulip analysis is named after the maps rally drivers use to navigate by. These maps are simply a list of diagrammatic turns. So the driver (in fact, the navigator next to the driver) see 'left turn' or 'right turn', sometimes with a degree of the turn. For example, a hairpin bend might have a different symbol to a slight kink in the road. Again from the introduction, there is some justification that these representations might not be peculiar to rally drivers. Montello (1991) suggests that we *remember* turns by category, and Hillier (1999) suggests that our constructive process leads to a set of turns by category. Further, our common experience, would lead us to notice that we often fail to remember almost-rectilinear systems, following paths that we think are orthognal when in fact they are not (Hillier, 2003). Could it simply be part of the human condition that turns are categorised rather than remembered explicitly? If so, could an analysis of categorised turns lead to a better correlation with human movement patterns than one without categorised turns? These questions are unanswered[1], but at least Depthmap allows us to investigate the latter one. Montello finds that turns are approximately discretised into $45°$ units, so it might be sensible to start off with this sort of categorisation — one 'bin' for a turn of less than $22.5°$, one for $22.5° - 67.5°$, one for $67.5° - 112.5°$ and so on — five bins in total. Depthmap actually thinks of a segmented circle, so it calls this an 8 bin tulip analysis. Any turn of less than $22.5°$ is forgotten, and given a value

[1] I should say, these questions are unnanswered as yet. I hope to have completed some work on tulip analysis results in the near future (Turner and Penn, 2005).

of zero, $22.5° - 67.5°$ is rounded to 45°, and given a value of 0.5, reusing the standard angular analysis scale provided by Hillier and Iida (2004). This categorisation might be a sensible start, but would others be more sensible? Depthmap gives the ability to segment to any degree within the range 4 to 1024 bins. You might ask why such a high number. The answer is that 1024 bins will of course approximate standard angular analysis. However, to its advantage, the algorithm for calculating tulip analysis is much faster than angular analysis, as the computer does not have to spend time deciding if an angle is exactly this or exactly that, and whether or not this angle is just less than that one or not, it just throws them into the appropriate bin. My results on this are quite advanced: for any reasonably sized system, mean angular depth calculated by tulip analysis at 1024 bins is almost identical to mean angular depth calculated exactly (Turner and Penn, 2005).

At the moment Depthmap uses a weighted radius to restrict analysis to a smaller area of the graph. This conflicts with Hillier and Iida's use of radius (who use number of segments), but it is consistent with the argument of segment count. In particular, it is noticeable that for small radii the measure provides little information for the angular mean depth (which tends to approximate the radius). To find differentiation, you must look at either the angular total depth or angular node count. That the average of these two values is roughly the same for all nodes indicates that, for segment maps made from axial maps, relativisation is not as much of an issue as I have made out — the angular total depth must be proportional to the total node count. An interesting observation, as that further implies a strong correlation with the axial view of the world, and Hillier's faith in the linearity of routes.

Chapter 6

Colours, Views and Manual Linking

This chapter is dedicated to the functions of the 'view' menu. Most are straight forward, but you will also find an important feature, linking and unlinking of axial lines or merging of VGA points under this menu. In addition, this chapter looks at how to change the foreground and background colours, the colour scale (and how Depthmap applies the colour scale), viewing options when you have both axial and VGA maps, and various 'show' options (grid, text and table). The chapter concludes with a discussion of the values shown on the status bar at the bottom of the Depthmap application window.

1 Manual linking and unlinking of graph nodes

Sometimes you will want to analyse a system over many layers, or a system where one road crosses another over a bridge. Since Depthmap can only analyse two-dimensional plans there is no automatic way to do this. You will need to import combined plans into Depthmap which have the multiple levels drawn separately. Note that for VGA, you can seed Depthmap points in entirely different locations using the fill tool. Unfortunately, the all-line axial seed does not work in the same way yet: if you click in a different location, it will simply regenerate the all-line map for the new location and *overwrite* the old all-line map. So, for the time being, if you would like to model multiple levels in an axial-map, you must hand-draw it[1].

There is an important distinction to be made between linking together axial lines and linking together VGA points. In VGA the points are *merged* together. For the purposes of graph analysis they are considered one point. It takes zero visual (or, indeed, metric) steps to go between them. The merge is also purely for global analysis of the graph (including calculating step depths). If you merge points it does not affect the values of connectivity for the merged points: each merged point still retains its own original connectivity in the 'Connectivity' attribute column. Local measures ignore the VGA merge and act as if the points were separated. By contrast, axial linking is much more intuitive: if you add a link, the connectivity of both newly linked lines is incremented by one, and the lines act as if they had always been linked. Below, the procedures

[1]Feasibly, you could generate the all-line maps for both levels, make fewest-line maps for both and export the lines, then reimport and link

for linking both VGA points and axial lines are described. There is no linking (or unlinking) enabled in the segment map (for details see page 28).

To link either VGA points or axial lines, you must first show the links. To do this, go to the 'View' menu, then select 'Show' and finally 'Links'. I am not at all happy with this placement of the menu item, lost as it is among other view options, and would be interested if anyone could let me know a more logical (and easier to find) place to put it.

1.1 Merging VGA points

Merging VGA points together is relatively straight forward. I will first describe how to do it, and then the rationale behind merging, and some notes on how I intended it to be used.

Before you can merge points, you must have made the visibility graph. Seed your levels with points, and make the graph as usual. Then, as described in the introduction to this chapter, change the view so that the links are shown by selecting 'Links' from the 'Show' menu under the 'View' menu. Two things should happen. The display of the graph should dim slightly, and a new mode button should now become available on the toolbar: 'Join'. To merge two points select this tool and click on one of the points you want to merge. The tool should change cursor to so it shows a '2', indicating that it requires a point to merge to. Click on the second point and a green line should appear connecting the two merged points. To unmerge a point, click twice with the join tool on the same point. A few problems with this approach become apparent as you use it:

1. You cannot merge a point with than one other point, as if you try to add another merge point, the original merge location is overridden. I will deal with this more fully when discussing the rationale in a moment.

2. You cannot change to another mode, to, for example zoom in or out, or pan the window with the hand-cursor, as the merge tool returns to its initial state when you swap back to it (which you may want to do if the points to be merged are separated by some distance). To overcome this second problem, you can use the arrow keys to pan or the '+' and '−' keys to zoom in or out. Remembering the initial state is a little tricky, as merge uses the standard select tool underneath its functionality, thus if you were to swap away and select some points, then return, the tool could easily be confused.

3. You cannot merge multiple sets of points together. This is true, and does stem from the one-to-one nature of merging. It is a similar problem to cutting and pasting of cells in a spreadsheet program. How do you ensure the range of cells is correct for both cut and paste. Further, specific to Depthmap, how does Depthmap know that you want to retain (or change) the orientation of a selection when you go to the set to merge. Of course, these are not hard problems to solve, but they need a little thought on the interface side.

The rationale for merging (rather than explicit linking as we will see with axial lines) is that other methods become very complex. In the merge, the global analysis functions in Depthmap simply regard the merged cells as co-present at the same physical location, and continue to analyse from there. If you consider how visual fields actually combine over three dimensions, it is much harder to work how to join them. Up an escalator you can see none of the floor above, looking down, you can see much of it. In other cases, a lift say, there is a distinct (vertical) jump before the visual field is reconnected, and on a winding staircase, you might see a portion of the staircase in each direction but no more. The viewer height suddenly becomes an issue. In geographic information science, where viewsheds are constructed, viewer height *is* consider, but viewsheds cannot go underneath each other. How tall is the actor, does it matter, should we handle visual possibility of seeing heads of people, and so on? Although these questions are *answerable*, they all require research. Until that research is done, it seems better to stick with the most simple method. The merge method is intended to work with points that overlap in space. You might merge all the points on a straight staircase, with the same staircase drawn in the plan for the floor above. Or on a dog-leg staircase, just a few points on the mezzanine landing. That is, you merge points that are intervisible in a transitionary space.

Although the merge does contribute its own obstacles to usability, it can still be used creatively to test ideas about how spatial systems work over multiple floors. Here is one example:

Tip: How to do lifts. As I have suggested, lifts require a vertical transition. They certainly do not merge directly into the space. A further problem is that lifts can join more than two floors simultaneously (in 2D space), yet merge requires you that you only merge two points together. In order to cope with this, when I make graphs, I assign a few points outside all space. These points will all naturally see each other, and therefore be one step apart. I then merge each lift point with each intermediate point. The result is a one step relationship to any other floor within the building.

1.2 Linking and unlinking axial lines

Linking and unlinking of axial lines is much more straight forward to think about than merging in VGA. If you want to link between floors, then you can add an extra link between two axial lines. As with VGA, you must first make your axial map using whichever method (note that segment maps cannot be linked), then switch the view so you are showing the links. Select 'Links' from the 'Show' menu under the 'View' menu. The axial lines colour should dim. Now select the 'Join' tool from the toolbar. To link two lines that are not already link, first select on and then the other. The connectivity for each line will be incremented. The link is considered like any other connection between lines, as a one-step change to get from one line to the other. A green line should appear linking the centre points of the two lines. Just as you can link two lines you can also *unlink* two lines. For example, if your map has a road bridge over another road that has been drawn as two overlapping lines. To unlink, first select one line and the *right click* on the second line. If the lines were previously linked, then a small red circle should appear where they intersect. Note that you can undo either of these operations by applying the opposite. To unlink lines that you have manually linked, simply manually unlink them again (the green line linking them should disappear).

If you link or unlink an axial map, those links and unlinks will be carried through to the segment map. Note that while it is perfectly reasonable to create unlinks in this way, linking cannot be guaranteed to work and Depthmap may crash as it tries to calculate the angle between the extensions of the two linked lines, which it requires to make the weighted segment graph (see making the segment map on page 28).

2 Changing the background and foreground colour

You can change the default foreground (the default colour for imported lines) and background colour for Depthmap graphs using the 'Foreground...' and 'Background...' options from the 'View' menu. These will permanently set your graph background and foreground colours, so they will remain the same next time you enter Depthmap.

Note that if you override the default foreground colour for a particular line layer in the line layer chooser (see page 7) then you will be unable to reset the line colour for that layer back so that it automatically picks up the default foreground colour. Whatever colour you choose for the layer will continue to override the default.

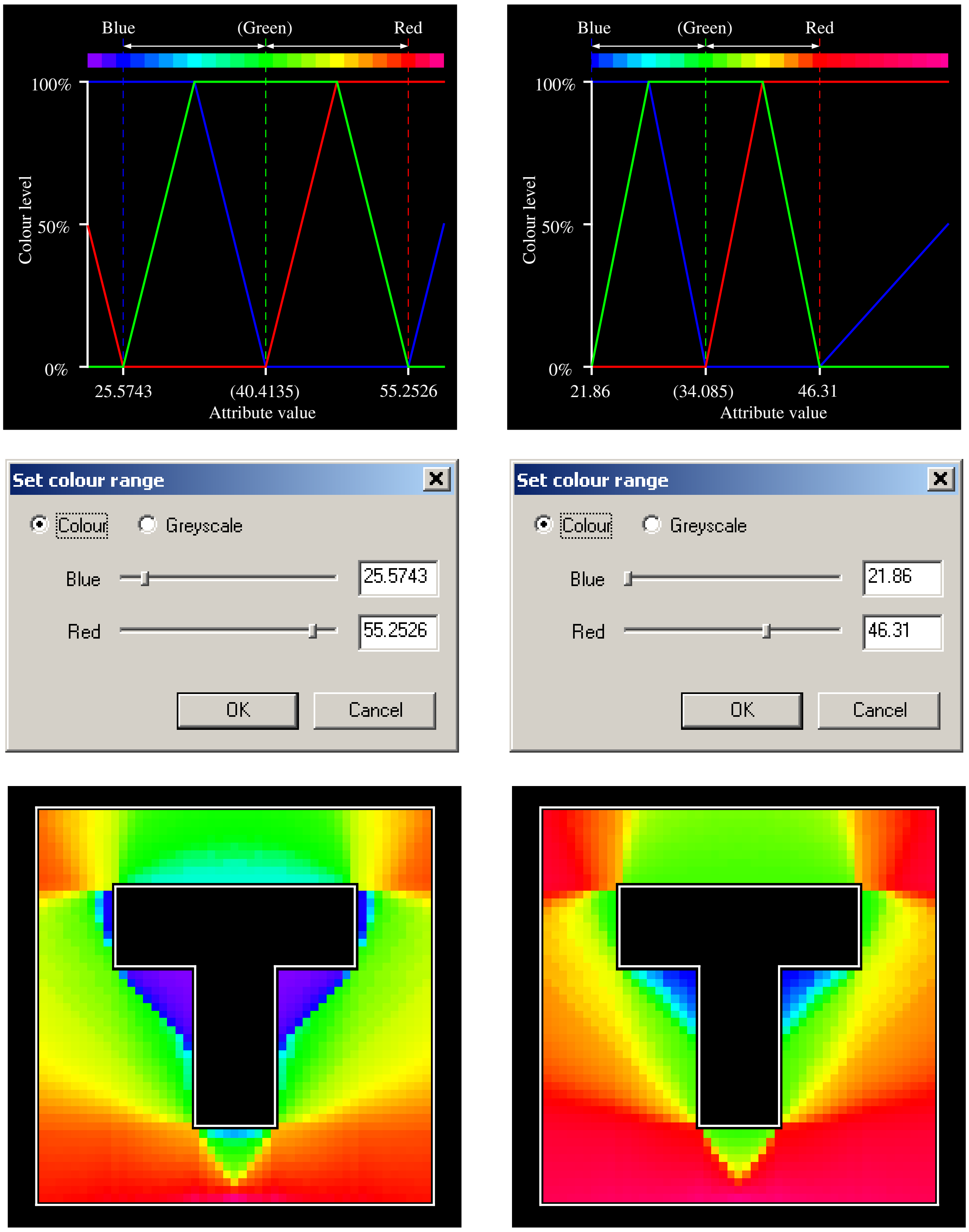
3 Setting the colour scale

By default, Depthmap displays attributes with a colour scale which runs from a blue tinged magenta for the very lowest value, to blue (through cyan) to green (through yellow) to red, and up to a red tinged magenta, for the very highest value. The scale is approximately continuous rather than discrete, that is, every value has a different colour assigned to it[2]. Each point or line's value is translated to a colour on this scale. The way the scale is allocated can be imagined by thinking of a rule where '0' corresponds to the lowest attribute value and '1' to the highest attribute value. By default, 0.0 is bluish magenta. This slowly changes to pure blue at 0.1. Between 0.1 and 0.5 the blue is slowly faded out and green faded in, so the midvalue is pure green. Similarly, between 0.5 and 0.9 green is faded out, and red faded in, so at 0.9 the value is pure red. This then changes to a reddish magenta at 1.0. Figure 3 demonstrates how the colours are blended to form the colour scale.

You can change this default scale by selecting 'Set Colour Range' from the 'View' menu. The bluish magenta and reddish magenta are pinned to the lowest and highest

[2]In fact, this is not quite true, Depthmap uses a scale of 192 colours where available on the computer you are using, so values will be grouped together into one of these 192 colours

[COLOUR PLATE]



Set colour range

[● Colour] ○ Greyscale

Blue 25.5743

Red 55.2526

Set colour range

[● Colour] ○ Greyscale

Blue 21.86

Red 46.31

Figure 1: How the colours are blended for different attribute values to create thhe colour scale used by Depthmap. The left column shows the default range for an example attribute. The right column shows how the range changes when the sliders are moved. Note that green always falls midway between the blue and red sliders.

values, but you can slide the value that is pure blue and the value that is pure red. The numbers to the right show the exact value that is blue or red. You can set these manually if, for example, you want to set a range outside that available by moving the sliders, or you want to assign a colour to a value exactly. The 'green point' is always midway between the blue and red values. If you wish, you can switch the colour scheme so that red corresponds to low values and blue to high values (if you make this switch, the magenta values also switch so the scale is in fact simply reversed). If you push the sliders to the ends, then the magenta colours are lost and the high and low values become red and blue. You can switch the colour scale to a grey scale if you prefer (for example, if you need to publish the image in a black and white format — see chapter 7, page 40 for how images can be extracted from Depthmap). In the grey scale version, there is no equivalent to the magenta ends of the scale, it simply runs from black through grey to white according to where you place the sliders for black and white.

The colour range setting does suffer a few slight problems. If the numbers are very large, the sliders appear to get stuck at the left-hand side of the range. You can override the slider positions by typing exact values into the blue and red value edit boxes. I do not have an explanation for this, but I will look into it. The sliders also become stuck at the left-hand side if there is no variation. This is perhaps more expected as Depthmap has no clue how a range should be assigned accross a single value. Again, the blue and red value edit boxes may be used to override this 'feature'. A second problem experienced is that Depthmap does not remember the colour scale you applied if you switch to another attribute and then switch back again. This, like the irritation of being unable to sort layers by name rather than file right at the beginning of this document, is one of those 'features' I have suffered with for a long time. A final irritation, which I cannot quite track down, is that the colour scale sometimes loses its numeric values altogether. You might see `-1.#IND` appear in the boxes. This means 'minus infinity', and once more, it can be overidden by typing into the edit boxes for the red and blue values.

4 Viewing VGA and axial maps together

If you make a visibility graph, and then also generate an axial map, you will see that the visibility graph is dimmed, while the axial map displays in full colour. The idea was that you might want to make a quick comparison between VGA layers and the axial graph attributes. Conversely, if you make an axial map and then make a visibility graph, the axial map will dim, but also, the VGA grid squares will become smaller. This, again, is so that you can have some idea of what the axial map looks like behind the VGA analysis.

You can switch back to the original view of either map easily. To do so, from the 'View' menu use the 'VGA Layers' or the 'Axial/Segment Layers' submenu to choose to hide either one of the maps. You can also use the menus to hide all the maps if you choose.

In addition, there is the option to 'Send to back' the foreground layer and 'Bring to front' the background layer when both are displayed simultaneously.

5 Showing the grid, text and tables

In this section, we will cover three remaining options on the 'Show' submenu of the 'View' menu: 'Grid', 'Table' and 'Text' ('Links' are dealt with separately above, see section 1 of this chapter).

When you make a VGA grid, or when you open a VGA graph file, a grid will be displayed over your plan so you can see where grid points will be or are filled. You can switch this grid on and off by choosing the 'Grid' option on the show view menu. It is somewhat irritating that this grid is displayed every time you reopen the graph, despite the fact you may have switched it off. The problem is, it is a user interface display option, and not stored as a flag with the VGA graph, and so the user interface displays the grid by default with any graph. Depthmap could turn off the grid for all graphs, but then it would not helpfully display when you are filling.

Table is a somewhat mysterious operation at first. If you display it with a visibility graph or an axial map, a bar appears on the left hand side of the screen, with a cross and tick icon, two apparent columns, 'ID' and 'Value', and, at least on my display, a line of text which reads 'Not avail...'. If you have smaller fonts set on your computer, you may see the whole of this text: 'Not available'. However, you will see that it loses the 'Not available' text if you view a VGA layer, such as the 'Gates' layer (see chapter 3, page 19 for more information about what VGA layers are). If you have actually added some gates (again, see the discussion from page 19), then you will also see a list of ID numbers appear for each gate. If, further still, you have pushed some values to the layer, then you should see a table of ID numbers for each gate and values next to them. You will also see that an edit box next to the cross and tick icons has become active. This edit box will allow you to change the value of the currently selected attribute column for layer objects. To do so, you must first select the layer object for which you want to change the value. You can select a layer object either by clicking on a row in the table view, or by clicking on one of the cells comprising the layer object itself. Layer objects are highlighted with a light blue colour rather than the usual light yellow. As usual, you can shift select to highlight more than one layer object (either in the table view or the map view), or drag select layer objects in the map view. Click on the edit box to activate it and type the new value you want into it. To change the value for all selected layer objects, click the tick icon. The cross icon does nothing. There is no undo for this operation, but it is quite easy to push values across to the layer again if you need to. Why you would want to change pushed values is another matter; in reality, you would probably want to use this feature to add new columns of values for observed gate counts.

There is a slight idiosyncrasy with the VGA layers select algorithm; it gets a little confused between selection of layer objects and selection of points. Layers need to allow you to select points, in order to let you create new gates. However, if you select a point when you have a layer object selected, sometimes it will clear the layer object selection, but other times it will fail to clear the selection, particularly if you have been using the table view to select layer objects. It is in a relative backwater of the user interface, but I hope someday I to make this more consistent.

There is one more viewing feature you should be aware of: ‘Text’. As with the ‘Table’ view, ‘Text’ viewing is only available with VGA layers currently. By default text is displayed for VGA layers, showing a numeric value for the currently displayed attribute. The *size* of this text is determined by the current zoom factor for the drawing plan. Note that if you zoom out too far, the text will not be displayed.

6 Information displayed on the status bar

As of Depthmap version 4, the status bar at the bottom of the Depthmap application window has changed slightly. From left to right, the three boxes show:

1. The number of nodes (either points or lines depending on the type of graph) in the graph. The number of points is particularly useful when you are filling a VGA grid with points (see chapter 3, page 9), so the number of points is also displayed at this stage, before the graph has been made. If you already have an axial map displayed, you will have to hide it (see section 4 of this chapter) while you are filling the grid in order to see the number of points, otherwise Depthmap will show the number of lines in the axial map.
2. The size of the imported drawing (maximum width $\times$ maximum height).
3. The current location of the cursor within the graph (x, y). The number is displayed to six significant digits.

The third box used to show some summary information about the currently displayed graph attribute. The current Depthmap is in flux. The display has been removed partly because it was difficult to interpret, partly due to the introduction of the display of attribute values when you hold the mouse over a point (tips, page 11) or line (tips, page 21), and partly because it is sometimes helpful to know the precise location of a value you are viewing. As for the flux, I intend to introduce a more comprehensive attribute viewer, along the lines of the table view, so you can select a range of points and see their average or maximum or minimum values alongside the corresponding values for the whole system.

Chapter 7

Exporting Results and Pictures

So far, we have found out a lot about how to analyse drawing plans, but nothing about how to get these results into presentations, statistical packages or geographical information systems (GIS). This chapter looks first at how to export data to other systems, and then how to export pictures from Depthmap.

1 Exporting analysed data from Depthmap

Data can be exported in two formats, either as a text file, or a MapInfo MIF/MID file combination. To export, simply choose 'Export Data...' from the file menu. Depthmap will export data for the currently displayed VGA layer or axial map, as chosen using the layer-selector drop list. For example the 'User Defined Map' or the 'Visibility Graph Analysis'. The default file name you are given reflects the data you will export, so for a graph file called `Graph1`, Depthmap will ask you if you want to export the data to a file called `Graph1_User_Defined_Map` or `Graph1_vga` as appropriate.

The default export data format is a 'tab-delimited' text file, which will be given a `.txt` extension when you export it. Tab delimited means that each column of values in the export file is separated by a `TAB` character. The tab-delimited text file format is default for importing files into Excel, DataDesk and a number of other analysis programs.

The file consists of a line of header information, followed by a line of data for each VGA point, data layer, axial line or axial segment. Whichever type of data Depthmap is exporting, it starts with a few default columns, and then prints out the column names as they appear in the attribute-selector drop box. For VGA points and data layers objects, the default columns Depthmap exports are an ID number ('ID') followed by 'x' and 'y' values. For VGA points, the 'x' and 'y' are simply the point location; for data layer objects, the centre of the points comprising it is used. For axial lines and segments, the default columns are ID number ('ID'), the first end of the line ('x1', 'y1') and the second end of the line ('x2', 'y2').

Depthmap hold point locations very accurately, so the 'x', 'y' and 'x1', 'y1', 'x2', 'y2' columns are all printed out to 12 significant digits (that is, the number is rounded after the 12th digit). However, column attributes are not held using such high accuracy, principally because there could be so many of them — when you have 100 000 points, the storage of columns becomes an issue. Depthmap prints out these columns to seven significant digits.

Depthmap can also export MapInfo files in MIF/MID format. To do so, change the file type in the export dialog box so that it reads 'MapInfo file' instead. Depthmap will export to files for the data, one with a `.mif` extension, and the other with a `.mid` extension, which may be imported into MapInfo. Depthmap performs two versions of MIF/MID export, one for VGA points and one for axial or segment lines. Note that it does not support export of VGA layers yet. If you originally imported a MIF/MID file to make an axial map (see chapter 4, page 1), then the coordinate system and bounds will be retained and exported with the data. Further, if you export an axial map with exactly the same lines as were originally imported from the MIF/MID file, Depthmap outputs any data that was in the MIF/MID files (note, if you change the lines, for example, by making a segment map, Depthmap cannot link the original data to the new). Included in the MIF/MID combination exported by Depthmap are the geometry (lines for axial lines, points for visibility graph analysis) and a default 'Depthmap ID' column (to distinguish it from any ID that MapInfo may have added). If there was no original MapInfo MIF/MID import of line data, then Depthmap uses a 'Non-Earth metres' coordinate system. That is, by default it exports points and lines as if they were on a flat plane, and as if the units were 'metres'. The units may not, of course, be correct: if you load from a CAT file, there are no units in any case, and in a DXF file, I do not know where the units used are recorded.

2 Exporting pictures from Depthmap

There are various ways to export the screen from Depthmap to other applications, or simply to the printer. They are listed here from 'most reliable' to 'least reliable'. All use vector graphics (i.e., the lines will print smoothly if imported into the correct software).

2.1 Copy screen

You can copy the contents of the screen to 'paste' into another application by selecting 'Copy Screen...' from the edit menu, or pressing `Ctrl+C`. Copy screen copies directly what you see on the screen to the Windows 'clipboard' as vector graphics. The vector graphics format is internal to Windows, it is called a 'Windows metafile' (WMF), and it can be read by most applications that support vector graphics. For example, Microsoft Word or Powerpoint, or Adobe Photoshop. However, some graphics applications may have problems reading it, as Microsoft regularly update the WMF format, and Depthmap uses a relatively new version that some older programs may not support fully. In particular, some programs insist on bringing the background to the front (see also printing below).

2.2 Export screen

You can export the current display to a file using the 'Export Screen...' command from the file menu. Export screen exports one vector graphics format: 'encapsulated postscript' (EPS). EPS is a standard publishing format. It is actually very similar to Adobe's PDF format, and Adobe products work well with this type of file. However, I have not maintained the export as well as I should have done: not everything you see on screen is included in every EPS file. What is worse, I do not know what is missing at the moment! The best way is to try the export and load into your software. Note that many programs do not have full vector graphics capability when importing EPS, so, e.g., some versions of Microsoft Word, do not load the EPS and instead print some text onto your screen.

2.3 Printing screenshots directly from Depthmap

Finally, you can print directly from Depthmap using 'Print...' from the file menu. However, what you print seems to come out at 'screenshot' size, i.e., small. Further, some printers fail to print the background until *last*, so your lines and points are obscured by it.

Chapter 8

Conclusion

While I was writing this document I was asked: "Why does a researcher's handbook contain a conclusion?". I suppose I want to tell you *why* Depthmap exists, and thus what that means for how it will be developed in the future. In no small part, I hope that many readers will be helping to develop Depthmap through their own analysis, programmed using the software developers' kit (SDK). As for Depthmap as it stands, I hope I have shown you not only how to use it, but some of the reasons for why it works the way it does.

Depthmap was originally programmed as my own research tool. To do whatever research I was interested in. Of course, as Depthmap has developed, I have had time to research some aspects more than others. What you see of through interface is often restricted by the current state of our research. For example, in chapter 5, I talked at some length about the relativisation of angular analysis. How angular analysis will eventually be relativised needs further research, so for the moment there is no algorithm included to do anything more than a basic angular mean depth, despite the reservations I expressed about it. The user interface itself, too, could be better, but we should remember that the interface is also a hostage to research direction. If we are suddenly interested by one sort of analysis, another sort gets put on hold. This is exactly what as happened with the addition of axial and segment analysis, while VGA has stayed very much as it is. There are explicit problems, of course; the index lists eleven, and no doubt there are many more. I hope to tick these off one at a time, no doubt you, the user, will be adding more as I do so! Which brings me to my next point: this document marks somewhat of a departure for me. I am currently engaged in writing my PhD, and I have engineered a lull in Depthmap development for us both. With the addition of an SDK, I hope that the more computer literate users will be able to write their own analysis. I too can turn my attention to develop on the SDK, for my PhD thesis is based on agent-analysis. The current agent-analysis is tacked onto Depthmap in a rather ugly way. I would like to rewrite the agent-based analysis now, so it can do everything I have been trying to do for my PhD, and at the same time, I hope I can give you the user interface to agents that they deserve. Furthermore, through this approach I can concentrate on writing a plug in to Depthmap, while Depthmap itself remains stable. I hope this document will allow you to work with Depthmap while I am engaged in this task, and that Depthmap will become more stable still as you do so. You have a handbook, and you have an SDK. Good luck to further analysis!

Finally, there are a few questions I would like to bring back to the fore. "Does VGA work?" is perhaps the greatest of them. My feeling is perhaps not for predicting people movement, but for informing us about the relationships between spaces and understanding the underlying principles of space, most certainly. For example, Hillier (2003) has used a comparison of metric and visual Depthmap measures to help elucidate his theory of space. I talked of Doxa's (2001) research on the National Theatre and Royal Festival Hall in London, but there is also further research in over a range of scales: Fong (2003) asks what a 'mega-mall', while Cutini (2003) shows how VGA measures can be used to extract the idea of the town square from the form of space itself. And then there are questions of research to do, too numerous to ennumerate in full: there is *k*-clustering (Jiang and Claramunt, 2004), there is comparing medial axis transforms and road centre-line maps (Turner, 2003), and there are many more. Some will require the use of the SDK to answer, but many others do not. Often a little lateral thinking is all that is required. It is always a joy to a programmer to see their program abused and used in a way it was never intended for. I am glad to say that I have been frequently surprised by how Depthmap has been abused (e.g., see Penn's (2003) wonderful 'shape of space' diagrams, analysed in Depthmap and then twisted in DataDesk). I hope that there is enough freedom within Depthmap to produce many more phenomena, and perhaps, to understand them too.

Appendix A

Graph Measures

This appendix is currently in preparation.

Appendix B

Agent-Based Analysis

This appendix is currently in preparation.

Appendix C

Software Developers' Kit

This appendix is currently in preparation.

Appendix D

Minor changes to Depthmap

Depthmap version 4.0624r

This version includes make boundary graph, angular analysis, and enabled import for modules written using the SDK.

Throughout, the 'ID' attribute label has been changed to 'Ref'. This is due to a problem importing text files with a column named 'ID' into Microsoft Excel.

Dephtmap version 4.07r

Some minor bugs with axial line generation have been fixed.

Dephtmap version 4.08r

Updated SDK library.

Dephtmap version 4.09r

Depthmap now loads plug-ins automatically; there is no need to use the import facility. Simply place the plug-in in the same folder as the Depthmap executable program and restart Depthmap.

Bibliography

Batty, M, Rana, S, 2002, "Reformulating space syntax: The automatic definition and generation of axial lines and axial maps", Working Paper 58, Centre for Advanced Spatial Analysis, UCL, London

Benedikt, M L, 1979, "To take hold of space: Isovists and isovist fields" *Environment and Planning B: Planning and Design* **6**(1) 47–65

Chapman, D, Kontou, F, Penn, A, Turner, A, 1999, "Automated viewshed analysis for configurational analysis of retail facilities", in *Proceedings 19th International Cartographic Conference*, Ottawa, Canada

Cutini, V, 2003, "Lines and squares: Towards a configurational approach to the morphology of open spaces", in *Proceedings of the 4th International Symposium on Space Syntax*, UCL, London, UK, pp 49.1–49.14

Dalton, N, 2001, "Fractional configurational analysis and a solution to the Manhattan problem", in *Proceedings of the 3rd International Symposium on Space Syntax*, Georgia Institute of Technology, Atlanta, Georgia, pp 26.1–26.13

de Arruda Campos, M B, Fong, P S P, 2003, "Bending the axial line: Smoothly continuous road centre-line segments as a basis for road network analysis", in *Proceedings of the 4th International Symposium on Space Syntax*, UCL, London, UK, pp 35.1–35.10

Desyllas, J, Duxbury, E, 2001, "Axial maps and visibility graph analysis", in *Proceedings of the 3rd International Symposium on Space Syntax*, Georgia Institute of Technology, Atlanta, Georgia, pp 27.1–27.13

Doxa, M, 2001, "Morphologies of co-presence and interaction in interior public space in places of performance: the Royal Festival Hall and the Royal National Theatre", in *Proceedings of the 3rd International Symposium on Space Syntax*, Georgia Institute of Technology, Atlanta, Georgia, pp 27.1–27.13

Fong, P S P, 2003, "What makes big dumb bells a mega shopping mall?", in *Proceedings of the 4th International Symposium on Space Syntax*, UCL, London, UK, pp 10.1–10.14

Gibbons, A, 1985 *Algorithmic Graph Theory* (Cambridge University Press, Cambridge, UK)

Hillier, B, 1999, "The hidden geometry of deformed grids: or why space syntax works, when it looks as though it shouldn't" *Environment and Planning B: Planning and Design* **26**(2) 169–191

Hillier, B, 2003, "The architectures of seeing and going — is there a syntax of urban spatial cognition?", in *Proceedings of the 4th International Symposium on Space Syntax*, UCL, London, UK, page forthcoming

Hillier, B, Hanson, J, 1984 *The Social Logic of Space* (Cambridge University Press, Cambridge, UK)

Hillier, B, Iida, S, 2004, "The human's shortest path is angular not metric" *(in preparation)*

Hillier, B, Penn, A, 2004, "Rejoinder to Carlo Ratti" *Environment and Planning B: Planning and Design* **31** forthcoming

Hillier, B, Penn, A, Hanson, J, Grajewski, T, Xu, J, 1993, "Natural movement: or configuration and attraction in urban pedestrian movement" *Environment and Planning B: Planning and Design* **20** 29–66

Jiang, B, Claramunt, C, 2004, "Topological analysis of urban street networks" *Environment and Planning B: Planning and Design* **31**(1) 151–162

Kuipers, B, Tecuci, D, Stankiewicz, B, 2003, "The skeleton in the cognitive map: A computational and empirical exploration" *Environment and Behavior* **35**(1) 80–106

Llobera, M, 2003, "Extending GIS-based visual analysis: The concept of visualscapes" *International Journal of Geographical Information Systems* **17**(1) 25–48

Montello, D R, 1991, "Spatial orientation and the angularity of urban routes" *Environment and Behavior* **23**(1) 47–69

O'Sullivan, D, Turner, A, 2001, "Visibility graphs and landscape visibility analysis" *International Journal of Geographical Information Systems* **15**(3) 221–237

Penn, A, 2001, "Space syntax and spatial cognition, or why the axial line?", in *Proceedings of the 3rd International Symposium on Space Syntax*, Georgia Institute of Technology, Atlanta, Georgia, pp 11.1–11.16

Penn, A, 2003, "The shape of inhabitable space", in *Proceedings of the 4th International Symposium on Space Syntax*, UCL, London, UK, pp 62.1–62.16

Penn, A, Conroy, R, Dalton, N, Dekker, L, Mottram, C, Turner, A, 1997, "Intelligent architecture: New tools for the three dimensional analysis of space and built form", in *Proceedings of the 1st International Symposium on Space Syntax*, University College London, London, UK, pp 30.1–30.19

Penn, A, Turner, A, 2002, "Space syntax based agent models", in *Pedestrian and Evacuation Dynamics* Eds M Schreckenberg, S Sharma (Springer-Verlag, Heidelberg, Germany) pp 99–114

Peponis, J, Wineman, J, Rashid, M, Kim, S H, 1998, "On the generation of linear representations of spatial configuration" *Environment and Planning B: Planning and Design* **25** 559–576

Ratti, C, 2004, "A critique of space syntax" *Environment and Planning B: Planning and Design* **31** forthcoming

Teklenburg, J A F, Timmermans, H J P, van Wagenberg, A F, 1993, "Space synax: Standardised integration measures and some simulations" *Environment and Planning B: Planning and Design* **20**(3) 347–357

Turner, A, 2000, "Angular analysis: a method for the quantification of space", Working Paper 23, Centre for Advanced Spatial Analysis, UCL, London

Turner, A, 2001*a*, "Angular analysis", in *Proceedings of the 3rd International Symposium on Space Syntax*, Georgia Institute of Technology, Atlanta, Georgia, pp 30.1–30.11

Turner, A, 2001*b*, "Depthmap: a program to perform visibility graph analysis", in *Proceedings of the 3rd International Symposium on Space Syntax*, Georgia Institute of Technology, Atlanta, Georgia

Turner, A, 2003, "Analysing the visual dynamics of spatial morphology" *Environment and Planning B: Planning and Design* **30**(5) 657–676

Turner, A, 2005, "Choosing a place to go or a direction to follow: The syntax the agent sees", in *Proceedings of the 5th International Symposium on Space Syntax*, TU Delft, Delft, Netherlands, page (in preparation)

Turner, A, Doxa, M, O'Sullivan, D, Penn, A, 2001, "From isovists to visibility graphs: a methodology for the analysis of architectural space" *Environment and Planning B: Planning and Design* **28**(1) 103–121

Turner, A, Penn, A, 1999, "Making isovists syntatic: Isovist integration analysis", in *Proceedings of the 2nd International Symposium on Space Syntax* Vol. 3, Universidad de Brasil, Brasilia, Brazil

Turner, A, Penn, A, 2002, "Encoding natural movement as an agent-based system: an investigation into human pedestrian behaviour in the built environment" *Environment and Planning B: Planning and Design* **29**(4) 473–490

Turner, A, Penn, A, 2005, "Following the tulips: Bringing the concept of turns into segmental analysis", in *Proceedings of the 5th International Symposium on Space Syntax*, TU Delft, Delft, Netherlands, page (in preparation)

Turner, A, Penn, A, Hillier, B, 2004, "An algorithmic definition of the axial map" *Environment and Planning B: Planning and Design* page in review

Watts, D J, Strogatz, S H, 1998, "Collective dynamics of 'small-world' networks" *Nature* **393** 440–442

Index