Depthmap

A program to perform visibility graph analysis

31

Alasdair Turner
University College London, UK

Abstract

Here we present Depthmap, a program designed to perform visibility graph analysis of spatial environments. The program allows a user to import a 2D layout in drawing exchange format (DXF), and to fill the open spaces of this layout with a grid of points. The user may then use the program to make a visibility graph representing the visible connections between those point locations. Once the graph has been constructed the user may perform various analyses of the graph, concentrating on those which have previously been found to be useful for spatial description and movement forecasting within the Space Syntax literature. Some measures which have not received so much mainstream attention have also been implemented. In addition to graph analysis, a few sundry tools are also enabled so that the user may make minor adjustments to the graph. Depthmap has been implemented for the Windows 95/98/NT and 2000 platforms.

Keywords:
visibility graph, spatial analysis, software development

31.1

Alasdair Turner
Center for Advanced Spatial Analysis, Torrington Place Site, University College London, Gower Street, London WC1E 6BT, UK
alasdair.turner@ucl.ac.uk

1 Introduction

The application of visibility graph analysis (VGA) to building environments was first introduced by Braaksma and Cook (1980). Braaksma and Cook calculate the co-visibility of various units within an airport layout, and produce an adjaceny matrix to represent these relationships, placing a '1' in the matrix where two locations are mutually visible, and a '0' where they are not. From this matrix they propose a metric to compare the number of existing visibility relationships with the number which could possibly exist, in order to quantify how usefully a plan of an airport satisfies a goal of total mutual visibility of locations. This type of analysis was recently rediscovered by Turner et al. (Turner and Penn, 1999; Turner et al., 2001), through considering recent developments in Space Syntax of various isovist approaches (see Hanson, 1998, for further background). They recast the adjaceny matrix as a visibility graph of locations, where a graph edge connects vertices representing mutually visible locations. Turner et al. then use several graph metrics as a few representative measures which could be performed on such graphs. The metrics they apply are taken from a combination of techniques used in Space Syntax and those employed in the analysis of small-worlds networks by Watts and Strogatz (1998).

Here we present a tool which enables a user to perform VGA on both building and urban environments, allowing the user to perform the kind of analysis proposed by Turner et al.. We call this tool 'Depthmap'. Depthmap first allows the user to import layouts in 2D DXF format and to fill the layout with a rectilinear grid of points. These points may then be used to construct a visibility graph, and we describe this procedure in section 2. After the graph has

been constructed, Depthmap gives the user several options to perform analysis on the graph, which we describe in detail in section 3. The methods include those employed by Turner et al., but also some, such as control (Hillier and Hanson, 1984) or point depth entropy (following a formula proposed by Hiller et al., 1987), which have not been previously applied to visibility graph analysis. We supply algorithmic and mathematical details of each measure, as well as samples of results, and short explanations of why we believe these may be of interest to other researchers. In section 4 we include a brief description of extra facilities included in Depthmap to adjust the edge connections in the visibility graph, and give guidance for their use in spatial analysis. Finally, we present a summary in section 5.

2 Making a visibility graph from a DXF file

In this section we describe how the user may get from a layout to a visibility graph using Depthmap. When the user first enters Depthmap, she or he may create a new graph file from the 'File' menu. This graph file initially appears blank, and the first thing the user must do is to import a layout to be analysed. The import format used is AutoDesk's drawing exchange format (DXF), and only two dimensional layouts may be imported. Once loaded, the user is shown the layout on the screen, and she or he may move it and zoom in and out as desired. Figure 1 shows the program at this stage. After the DXF file has been imported, the user is ready to add a set of locations to be used as a basis for the visibility graph.

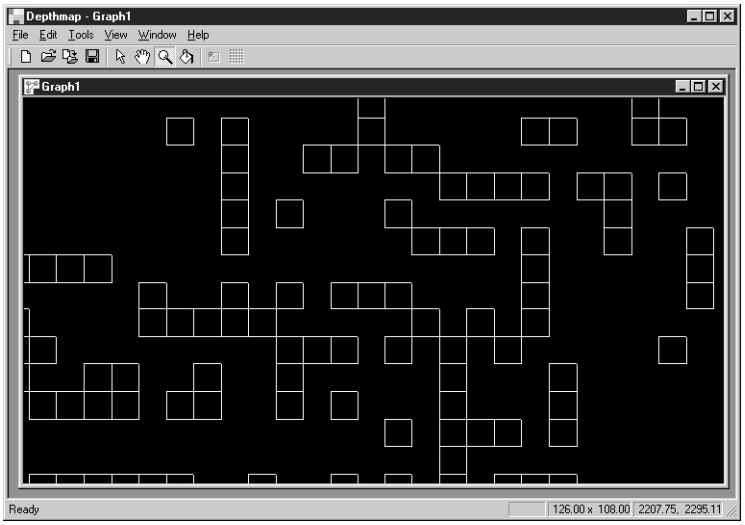


Figure 1: The application as it appears after the DXF file has been imported

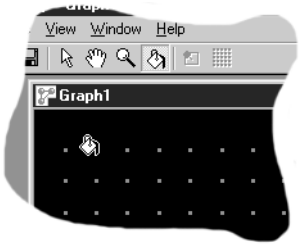


Figure 2: The user fills a section of open space using the fill tool

2.1 Making a grid of locations

To make a set of locations, the user selects the 'Fill' tool from the toolbar, and then clicks on a seed point in open space to make a grid of points, as shown in figure 2. When the user clicks for the first time, she or he is given the option of choosing a grid spacing, based on the units used in the original DXF, and may use for example, a 1m grid. The grid is filled from the seed point using a simple flood fill algorithm. Points may be deleted by selecting either individual points or ranges of points. Once the user is happy with the grid, she or he may then go on to make the visibility graph.

The user is restricted to a rectilinear grid of points at present. We realise that this is a problem for anyone attempting a methodological investigation of VGA, rather than simply using it to analyse environments. This is due to using a grid assumption when we calculate point intervisibility, and unfortunately a constraint. However, various themes on the basic grid are still available, for example by using different resolutions or by rotating the DXF prior to import, or by only partially filling grids.

2.2 Making the graph

The graph is made by Depthmap by selecting the 'Make Graph' option from the 'Tools' menu. The program attempts to find all the visible locations from each grid location in the layout one by one, and uses a simple point visibility test radiating from the current location to achieve this. As each location is considered, a vertex is added to the graph for this point, and the set of visible vertices is stored. Note that this is only one way that the visibility graph could be stored. It is actually more space efficient only to store the edges, but this would lead to slower algorithms later when analysing the graph. We can write a simplified form of the algorithm in pseudocode as follows. In the algorithm we use graph set notation: V(G) is the set of all locations or vertices that exist, and v_i an individual location or vertex in the graph we are making. Each vertex v_i will have a set of vertices connected to it, which will labelled the set $V(\Gamma_i)$, otherwise known as the vertex's neighbourhood.

```
for v_i in V(G)
begin
    for v_j in V(G)
    begin
      if v_i 'can see' v_j then add v_j to V(Γ_i)
    end
end
```


The number of vertices in the neighbourhood is obviously easily calculable, and Depthmap records these neighbourhood sizes as it makes the graph. In graph theory, the neighbourhood size for a vertex is commonly written ki, and may be expressed as in equation 1.

$$k_i = |V(\Gamma_i)| = |v_j : \{v_i, v_j\} \in E(G). \qquad (1)$$

where E(G) is the set of all edges (i.e., visibility connections) in the graph. Note that the set E(G) is not actually stored by Depthmap, and so k_i is actually calculated using the first form of this equation. Figure 3 shows a simple layout after the visibility graph has been made using Depthmap. As the actual number of connections is huge for each vertex, only k_i values are shown for each location rather than the mess of actual connections. Depthmap colours k_i values using a spectral range from indigo for low values through blue, cyan, green, yellow, orange, red to magenta for high values. The user may change the bounds for this range, or choose to use a greyscale instead, using a menu option from the 'View' menu. Since this paper is reproduced in greyscale, all the figures shown have used the greyscale option, where black represents low values and white represents high values. Once the graph has been constructed, the user has various options, including graph analysis, which we describe in the next section, and modification of the graph, which we describe in section 4.

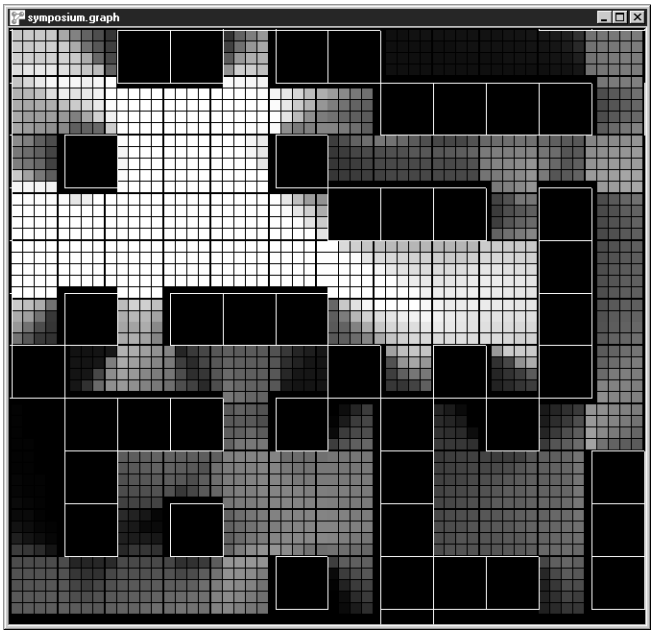


Figure 3: Neighbourhood size calculated for a sample layout

3 Measurement of the graph

In this section we describe the graph measures available to a user of Depthmap. Analysis of the graph is split into two types: global measures (which are constructed using information from all the vertices in the graph) and local measures (which are constructed using information from the immediate neighbourhood of each vertex in the graph). The user may elect to perform either or both of these types of measure by selecting from the 'Tools' menu. She or he is presented with a dialog box, as shown in figure 4. The 'radius' box allows the user to restrict global measures to only include vertices up to n-edge steps from each vertex. When the user clicks OK, the program calculates the measures for the whole system. The key global measures are mean depth and point depth entropy, while the key local measures are clustering coefficient and control. Once the measures have been calculated, these and derived measures are given as options on the 'View' menu, and may be exported as plain text for use in GIS and statistical packages from 'File' menu. We now turn to a discussion of the algorithmic implementation of each measure, and explore possibilities for their use within Space Syntax research.

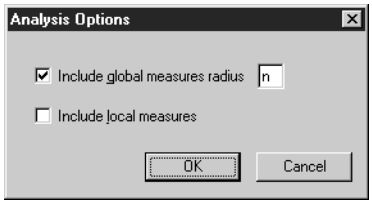


Figure 4: The options dialog box for graph analysis

3.1 Clustering Coefficient

Clustering coefficient, γ_i, is described in detail by Watts (1999) and has its origin in the analysis of small-world networks. Turner et al. found it useful for the detection of junction points in environments. Clustering coefficient is defined as the proportion of vertices which are actually connected within the neighbourhood of the current vertex, compared to the number that could possibly be connected, as shown in equation 2.

$$\gamma_i = \frac{|E(\Gamma_i)|}{k_i(k_i - 1)} \qquad (2)$$

where $E(\Gamma_i)$ is the set of edges in the neighbourhood of v_i and k_i is the as previously calculated. This is implemented in Depthmap by the following algorithm for each vertex in the graph*

$\gamma_i = 0$
for v_j in $V(\Gamma_i)$
begin
    for v_k in $V(\Gamma_j)$
    begin
        if v_k in $V(\Gamma_i)$ then $\gamma_i = \gamma_i + 1$
    end
end
$\gamma_i = \gamma_i / k_i(k_i - 1)$

* Again note that as the set of edges $E(\Gamma_i)$ is not recorded, the information must be recovered from the vertices in the neighbourhood, $V(\Gamma_i)$

Figure 5 shows the clustering coefficient calculated for a sample spatial configuration. As noted by Turner et al. junctions in the configuration are picked out. However, as they suggest, and looking at the figure, it actually seems to pick out better the changes in visual information as the system is navigated, so low clustering coefficients occur where a new 'area' of the system may be discovered. Examining γ_t, seems a promising line of investigation when looking at how visual information varies within an environment - for example, as suggested by Conroy (2000) for finding pause points on journeys.

3.2 Control

Control for a location, which we will label ci, is defined by Hillier and Hanson (1984), and is calculated by summing the reciprocals of the neighbourhood sizes adjoining the vertex, as shown in equation 3.

$$c_i = \sum_{v_j \in V(\Gamma_i)} \frac{1}{k_j} \qquad (3)$$

A simple algorithm can be used to calculate this value as follows:

$c_i = 0$

for v_j in $V(\Gamma_i)$

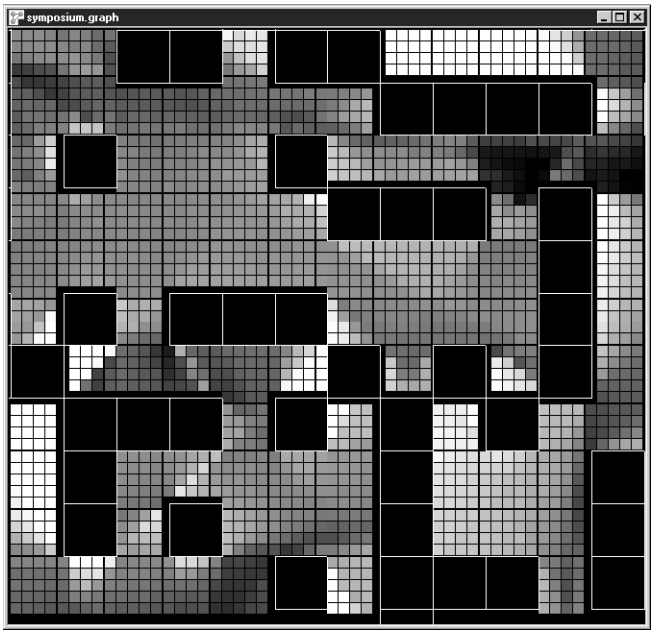
begin

$c_i = c_i + 1/k_j$

end

It should be noted that in VGA many of the immediately adjoining neighbourhoods will overlap, so that perhaps a better definition of VGA control would be the area of the current neighbourhood with respect to the total area of the immediately adjoining neighbourhood - that is, rather than use the sum the size of all the adjoining neighbourhoods, use the size of the union of those adjoining neighbourhoods as shown in equation 4. The results of applying this method are shown in figure 6, although Depthmap is also capable of calculating control as defined by Hillier and Hanson.

$$c_i' = \frac{k_i}{|\bigcup V(\Gamma_j) : v_j \in V(\Gamma_i)|} \qquad (4)$$



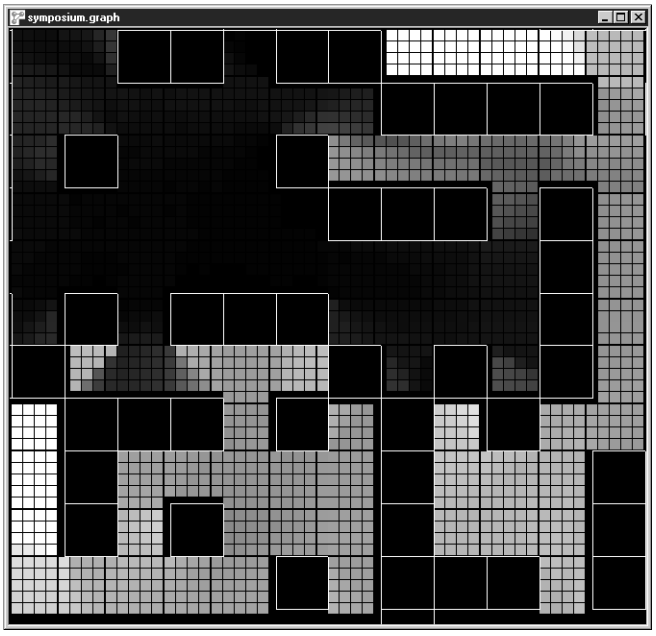


Figure 5: Clustering coefficient calculated for a sample layout

Figure 6: Control calculated for a sample layout

3.3 Mean Depth

The mean path length L_i from a vertex is the average number of edge steps to reach any other vertex in the graph using the shortest number of steps possible in each case. This sort of graph measure has a long history stretching back as far as Wiener (1947), and is pertinent to visibility graph analysis due to the parallels with the use of integration in space syntax theory (Hillier et al., 1993), showing how visually connected a vertex is to all other vertices in the system. We calculate L_i by constructing the set of point depths, as follows. The algorithm we use is not the most time efficient, as shortest paths are recalculated for each vertex, rather than being stored in a cache. However, the memory constraints on current personal computers mean that storing all the shortest paths in the system would rapidly use up the available memory. Hence, the algorithm that follows works in $O(n^2)$ time. It obtains point depths for all the vertices in the system from the current vertex, by adding ordered pairs of vertices and depths to the set P.

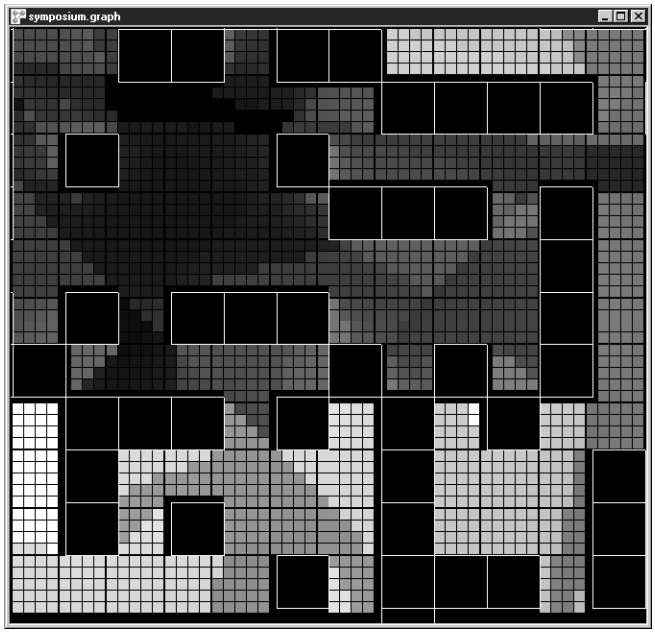


Figure 7: Mean depth calculated for a sample layout

$P_i = \{v_i, 0\}$

$n = 0$

for $\{v_j, n\}$ in P_i

begin

  for v_k in $V(\Gamma_j)$

  begin

    if $\{v_k, *\}$ not in P then add $\{v_k, n+1\}$ to P_i

  end

  if finished $\{*, n\}$ then $n = n + 1$

end

An asterisk, such as in the set pair $\{v_1, *\}$, represents a wild card matching operation. For example, $\{v_1, *\}$ matches any of $\{v_1, 1\}$, $\{v_1, 2\}$ or $\{v_1, 4\}$.

Once the point depth set has been constructed, it is facile to calculate measures such as mean depth and integration. Figure 7 shows mean depth for our sample system. As has been shown, this measure would seem to be useful understanding movement of people within building environments, where it is difficult to apply traditional Space Syntax methods such as axial analyses at high resolutions. However, in urban environments, since we are measuring numbers of turns from location to location, VGA integration quickly approximates to axial integration (albeit with each line weighted by the street area), and due to speed considerations, it may not be as beneficial to use VGA integration in these situations.

3.4 Point Depth Entropy

In addition to calculating measures such as mean depth, the point depth set P_i allows us to explore measures based on the frequency distribution of the depths. One such measure is the point depth 'entropy' of a location, s_i, which we can express using Shannon's formula of uncertainty, as shown in equation 5. Entropy occurs in many fields, including informatics, and is proposed for use in Space Syntax by Hiller et al. (1987).

$$s_i = \sum_{d=1}^{d_{\max}} -p_d \log p_d \qquad (5)$$

where d_{max} is the maximum depth from vertex v_i and p_d is the frequency of point depth *d* from the vertex. This is implemented algorithmically in Depthmap as follows:

$s_i = 0$
$d_{\max} = 0$
for $\{*, n\}$ in P_i
begin
    if $n > d_{\max}$ then $d_{\max} = n$
end
for n in 1 to $d_{\max}$
begin
    $p = \text{count}(\{*, n\}) \ / |P_i|$
    $s_i = s_i - p \log p$
end

Calculating point depth entropy can give an insight into how ordered the system is from a location. For example, if a doorway is connected to a main street then there is a marked disorder in the point depths from the point of view of the doorway: at depth 1 there are only a few locations visible from the doorway, then at depth 2 there are many locations from the street, and then order contained within further depths will depend on how the street is integrated within its environment. Figure 8 shows the point depth entropy as calculated by Depthmap. Other entropy-like measures are also calculated. The information from a point is calculated using the frequency with respect to the expected frequency of locations at each depth, as shown in equation 6. Obviously, what is meant by 'expected' is debatable, but as the frequency is based on the probability of events occurring (that is, of the j graph splitting), it seems appropriate to use a Poisson distribution (see Turner, 2001, for a more detailed discussion), which has the advantage of depending only on a single variable, the mean depth of the j graph. The resulting formula is shown in equation 6 and is similar to that used by Hiller et al. for relativised entropy. So, why calculate the entropy or information from a point? The answer is threefold: firstly, it was found to be useful by Hiller et al.; secondly, it appeals intuitively to a tentative model of people occupation of a system, in that the entropy corresponds to how easy it is to traverse to a certain depth within the system (low disorder is easy,

high disorder is hard); and thirdly, it remedies the problem that VGA integration is heavily biased towards large open areas. In axial integration, because the system is dimensionless, large open areas do not unduly weight the values of the lines; that is, the large areas only weight the values by their increased connections, not through their area. By contrast, in VGA integration the measure approximates a mean of distance times area, as discussed in the previous section. Hence, by using a topological measure such as point depth entropy we eliminate the area dependence of the measure, and instead concentrate on the visual accessibility of a point from all other points.

$$r_i = \sum_{d=1}^{d_{\max}} p_d \log \frac{p_d}{q_d} \quad \text{where} \quad q_d = \frac{{L_i}^d}{d!} e^{-L_i} \tag{6}$$

4 Further tools available in Depthmap

As well as allowing the user to make and analyse graphs, Depthmap includes the facility to modify the graph connections. We believe this will be useful when the user wishes to model something other than a two dimensional plan - for example, when modelling a building with more than one storey, or trying to include one way entrances or exit, escalators and so on. The method to modify connections in Depthmap is as follows. Once the graph has been made, the user first selects an area by dragging a select box across the desired portion of the graph, as she or he would in many computer applications. She or he may also select other areas by holding down the 'Shift' key and then selecting again. The user then 'pins' this selection, using a toolbar button. Now the user may select another area in the same way. Once these two areas have been selected, the user can alter the connections between them by selecting the 'Edit Connections' dialog box from the 'Edit' menu. An example is shown in figure 9. The dialog box gives several options but essentially these reduce to set operations on the neighbourhoods of the selected points. The 'Add' option simply adds the selected points in the other set to the neighbourhood, and is useful for turning points, for example a stairwell landing. The 'Merge' option allows the user to union the neighbourhoods of the two selected sets, and is useful for adding seamless merges, for example, descending an incline. Finally the 'Remove' option can be used to take away any connections from the selected set, and for example, might be useful to convert a two way entrance to a one way entrance.

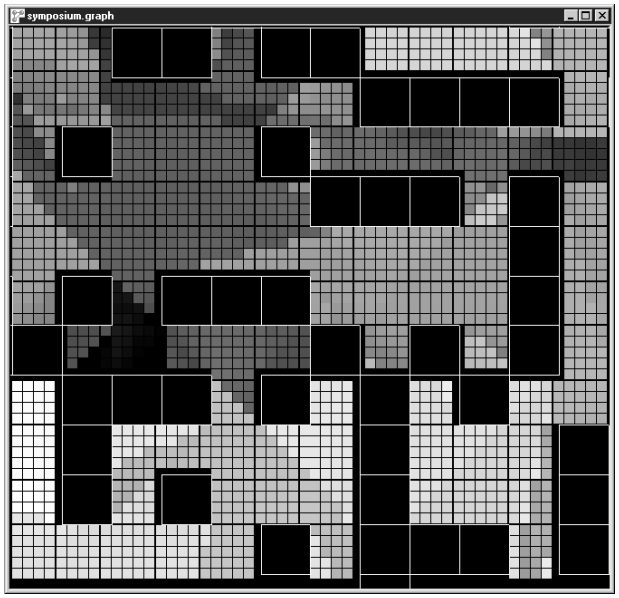


Figure 8: Point depth calculated for a sample layout

5 Conclusion

In this paper, we have presented a description of the Depthmap program, designed to perform visibility graph analysis (VGA). Depthmap can be used to analyse a layout to obtain various graph measures, including integration, which has been shown to correlate well with observed movement patterns when used with axial maps (see Hillier et al., 1993, for example), and also shown to correlate with movement patterns when used with VGA (see Turner and Penn, 1999). Although we have talked only about the overall application of Depthmap to a system to make and analyse graphs, Depthmap also has many other features

which a user would expect from a program, such as printing, summarising graph data and so on, which we have restricted to a user manual. What we do hope to have given here is a flavour of what is achievable with the program, an insight into how the graph is analysed, and our reasons for choosing the graph measures we have included.

Finally, it is worth noting that Depthmap is designed to be a tool for Space Syntax researchers. This means that we have chosen DXF as the input medium, and that the program runs interactively, allowing the graph to be modified and the analysis to be reapplied. It also means that sometimes we have not used the fastest algorithm to achieve a task, but have instead designed the program to work with the memory available on most personal computers today. On a 333 MHz machine running Windows 98, Depthmap takes around an hour to process a graph with 10 000 point locations, from start to finish, and the program has been tested on graphs with up to 30 000 point locations. We hope that the Space Syntax community will enjoy using our tool and we look forward to improving it with the input and insight of future researchers.

References

Braaksma, J P and Cook, W J, 1980, "Human orientation in transportation terminals" *Transportation Engineering Journal* **106**(TE2) 189-203

Conroy, R, 2000 *Spatial Navigation in Immersive Virtual Environments* PhD thesis, Bartlett School of Graduate Studies, UCL

Hanson, J, 1998 *Decoding Houses and Homes* (Cambridge University Press, Cambridge, UK)

Hiller, B, Hanson, J and Peponis, J, 1987, "The syntactic analysis of settlements" *Architecture and Behaviour* **3**(3) 217-231

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

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

Turner, A, 2001, "Angular analysis" *Proceedings of the 3rd International Symposium on Space Syntax* Georgia Institute of Technology, Atlanta, Georgia.

Turner, A, Doxa, M, O'Sullivan, D and 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) Forthcoming

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

Watts, D J, 1999 *Small Worlds* (Princeton University Press, Princeton, NJ)

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

Wiener, H, 1947, "Structural determination of paraffin boiling points" *Journal of the American Chemistry Society* **69** 17-20

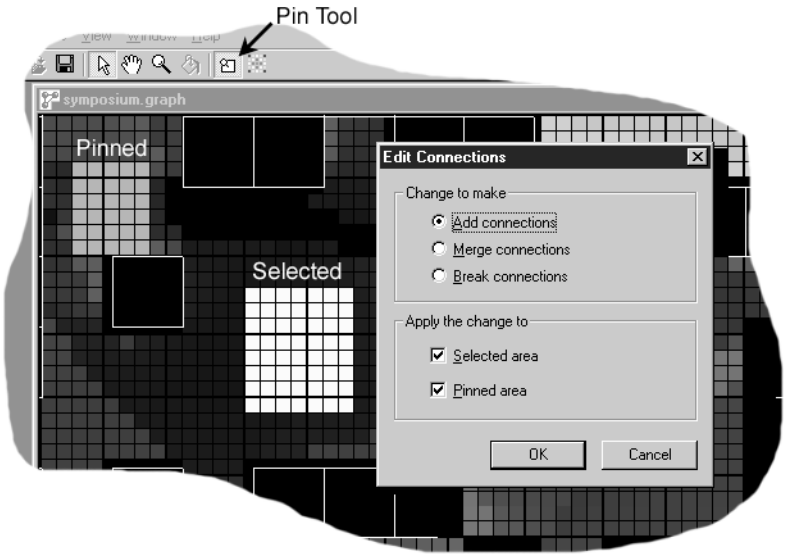


Figure 9: Editing connections