TOOLS FOR VISUAL AND SPATIAL ANALYSIS OF CAD MODELS

implementing computer tools as a means to thinking about architecture

ELLEN YI–LUEN DO
College of Architecture, Georgia Institute of Technology,
Atlanta, GA 30332-0155, U.S.A. ellendo@cc.gatech.edu
& Sundance Lab for Computing in Design and Planning,
University of Colorado, Boulder, CO 80309-0314,U . S. A.

AND

MARK D. GROSS
Sundance Laboratory for Computing in Design and Planning,
College of Architecture and Planning, University of Colorado,
Boulder, CO 80309-0314, U. S. A. mdg@cs.colorado.edu

Abstract. The paper describes a suite of spatial analysis programs to support architectural design. Building these computational tools not only supports the task of spatial analysis for designers but it also helps us think about the spatial perception. We argue that building design software is an important vehicle for understanding architecture, using our efforts to build various visual and spatial analysis tools as examples.

1. Tool building as a way to understand design concepts

The past twenty-five years have seen the development and refinement of tools for simulating, analyzing, and predicting the performance of building designs with respect to lighting, energy, acoustic, and structural behavior. Surprisingly, little work has been done on modeling the spatial characteristics of designs, although they strongly influence users' experience of buildings, and determining the spatial structure of a building is the central focus of architectural design. Tools for analyzing spatial characteristics have for the most part been neglected in CAAD research.

Plenty of work has been done outside the arena of CAAD on spatial and visual analysis of built environments. Architectural researchers have looked at the relation between physical features of the built environment and experiential qualities that users perceive. For example, Benedikt (Benedikt 1979), Thiel (Thiel 1961; Thiel 1981), Appleyard, Lynch, & Myer (Appleyard and others 1964) among others have proposed frameworks, theories, and specific analyses of perceptual space that can—and we argue ought—be made computable. Computations of these analyses would be immediately usable both to CAAD users directly as well as to intelligent CAAD programs of the type described by Koile (Koile 1997). We suggest however that implementing such analyses in a

computer program is often nontrivial and raises issues may have been glossed over in the original, non-computational, formulation.

In this paper we aim to do two things. First, we describe a class of tools for spatial analysis of building designs based on Benedikt's isovist research and other models of the perception of architectural space. Second, we point out that the detailed implementation strategy chosen to build a tool for analysis can have a tremendous effect on the outcome, not only making the tool more or less efficient, but actually affecting the framing of architectural concepts. Thus we see the building of CAAD tools not only as a means to perform useful calculations, but also as a way to think about underlying and central concepts of space, place, and architecture.

In the following sections we discuss computational tools for spatial analysis in architecture. In particular, we consider how different implementation approaches influence the results of the computation, and more fundamentally, how the tools lead us to think differently about underlying architectural concepts. We focus here on models of the perception of spaces and places as exemplified by Benedikt's work on isovist (Benedikt 1979). Accordingly, section two reviews briefly the underlying work of Benedikt and others on spatial and visual analysis. Section three discusses our experience implementing visual and spatial analysis tools for CAD models. We briefly describe the algorithms, and presentations of these tools. Finally, in section four, we compare the strengths and weaknesses of these different methods and propose an agenda for future work.

2. Concepts of visual and spatial analysis

Good designers manipulate the placement and spatial relations of walls, screens, and partially defined boundaries to provide feelings of containment and privacy, opportunities for survey and outlook, and the sense of flow and direction in a building. A well designed building is an appropriate and varied arrangement of spaces whose spatial characteristics match intended uses and offer a range of physical definitions of habitable space. Analyses of the spatial and visual character are often done for the design of museums, office work group layouts, prisons, and religious and ceremonial buildings. The level of analysis ranges from informal inspection to a more careful checking of sight lines and boundaries. Often, though, architects do not perform spatial analysis, but simply work on the basis of their own experience when designing the spatial characteristics of a building.

Rooted in the principles of human perception (e.g., Ittelson's *Visual Space Perception* (Ittelson 1960)) this relationship between form and perception, environment and behavior is widely discussed throughout the discipline of architecture. Porter's *How Architects Visualize* (Porter 1979)) offers a good overview of the perceptual issues. Ashihara's *Aesthetic Townscape* used 'degree of enclosure' to describe and discuss the composition of townscapes, observing for example, that people pause and linger in the 'inside corners' of spaces (Asihara 1983). Lynch discussed 'imageability' (Lynch 1960), and changes of views in sequence when moving in an environment (Lynch 1972), and the use of 'viewshed' -- terrain maps for analyzing visual effects from a major viewpoint

(Lynch 1976). Appleyard, Lynch, and Myer discussed spatial and visual perceptions associated with movement along a highway (Appleyard and others 1964).

Various theoretical and empirical research efforts have been made to account for the relation between physical form and human perception of spaces, which we review below briefly. These include Benedikt's work on isovists (Benedikt 1984; Davis and Benedikt 1979) and the work of Hillier and Hanson (Hillier and Hanson 1984) on space syntax and Peponis on various space partitioning schemes (Peponis and others in press). However these efforts to understand the relation between built form and perceived space have not been widely applied, in part because they involve tedious calculations from floorplans. These calculations can be easily made using computers, adding spatial analysis tools to the growing collection of tools for analyzing building performance.

Benedikt proposed that a space, or an environment, is perceived as a collection of visible surfaces not occluded by physical boundaries such as walls and partitions (Benedikt 1979). He defined an 'isovist' at a given point in the floorplan as the space visible from that point, looking around in 360 degrees. Quantifying the isovist areas, perimeters, and solid boundaries can be used to compare the quality of different spaces. He further described making physical (say, clear plexiglass) models of isovists along a defined path and stacking these models to visualize a moving user's changing perception of a built environment. In his papers (Benedikt 1979; Davis and Benedikt 1979) he described computational implementations that were done at the University of Texas at Austin in the 1970's.

In *The Social Logic of Space*, Hillier and Hanson presented 'space syntax' as a way to describe and analyze the character of spaces and predict human behavior patterns and cultural activities in these spaces (Hillier and Hanson 1984). In space syntax the connectivity between spaces—the integration value—captures the depths of topological values. Hillier and Hanson analyzed two dimensional planar views to draw and quantify connectivity and integration values. First, a convex space is indicated by drawing maximal 'space bubbles' between wall partitions. Then an 'axial line' is drawn to connect convex spaces. The number of intersections for each axial line in relation to overall area produces an 'integration value'. Empirical observations found that the frequency of activities coincides with integration values of an axial map, that is, in real life more highly integrated places tend to have more activities.

Researchers at University College, London have developed the Axman program to support axial map calculation. Users import a picture of the floor plan or bubble diagrams as an underlay and then draw axial lines on top of the plan. The Axman program finds intersections among all axial lines, calculates the integration values for each line, and displays axial lines in different (rainbow) colors: Red lines indicate the most integrated paths and blue or violet lines indicate the least integrated ones.

Peponis, WIneman, and others. (Peponis and others in press) recently proposed various spatial analyses to study the shape and spatial configuration of building plan. These analyses are based on the identification of the boundaries of spaces in a floorplan with certain properties, in particular with respect to a viewpoint moving through the floorplan. They recognized that a wall end or a corner can be an important index of spatial qualities because the 'endpoints' appear and disappear or remain invisible as the viewer moves. Their endpoint partition', for example, identifies convex areas that are

"informationally stable" with respect to shape, i.e. all their points are visually connected to the same edges of the plan. They are supervising a computer implementation of spatial partition analysis in the Microstation environment (Peponis and Wineman 1997).

3. Implementation methods of spatial analysis tools

We describe our work on automated spatial analysis of CAD models, which includes two main styles of analysis. The first, a discrete approach, involves calculations of perceived space, by computing the projections and extensions of physical boundaries, and the computation of viewsheds and visual fields. The second, a continuous approach, involves the calculation of spatial gradients or fields. We describe several programs, built on a variety of platforms, including PL/I, Topdown Pascal on Macintosh, Tk/Tcl on Sun, AutoLISP with AutoCAD, and Macintosh Common Lisp, that perform these computations on floor plans and sections entered by a designer.

3.1. ENCLOSURE CALCULATION (ENCLOSURE)

As an undergraduate project in 1977 Gross built an enclosure measuring program in PL/I, (subsequently reproduced in 1982 by Gross and Fred Wu in Apple II Logo). The Enclosure program calculated numeric values for subdivided spaces in a floor plan, intended to model a user's feeling of enclosure or protectedness at each point in the plan.

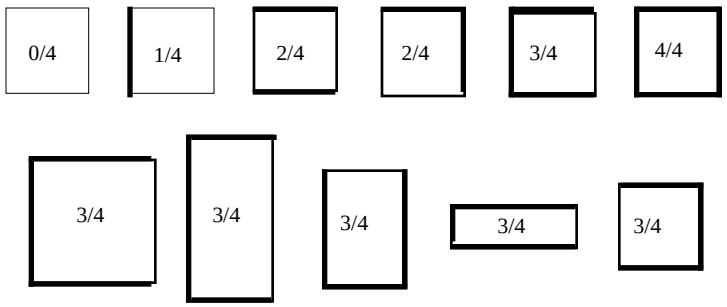


Figure 1. Top: Enclosure value counts number of adjacent walls. Bottom: Spaces bounded by three walls have same enclosure value 3/4 regardless of size.

As shown in Figure 1, enclosure values ranged from 0 (fully open) to 1 (fully enclosed). First, the program divided the floor plan into subspaces by extending the lines of walls and constructing perpendicular lines at wall endpoints (figure 2). This creates many subspaces with different sizes. Then it counted the number of walls in the boundaries of each subspace and divided by the number of boundaries to normalize the values from 0 to 1. For example, a four sided subspace has the value 1/4 if it has one wall boundary, 2/4 if there are two walls, and 4/4 if the space is surrounded, totally enclosed by four walls. The numeric enclosure value is displayed as gray scale intensities on the floor plan.

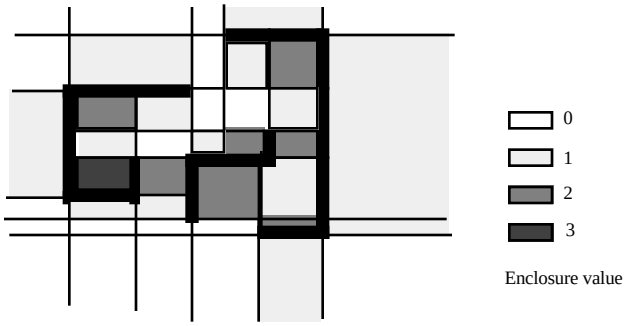


Figure 2. Values of subdivided spaces computed by Enclosure.

3.2. GRADIENT ANALYSIS - TOPDOWN ISOVIEW

A second approach to visual and spatial analysis is a discrete simulation program called Isoview that computes and displays "degree of openness" of cells in a space (Do 1993). Isoview was implemented in Topdown Pascal (Mitchell and others 1990), a parametric programming environment that provides interface widgets such as slider bars and buttons and a library of graphics routines. Users of the Isoview program used Topdown's parametric interface tools to position walls and openings and adjust the sizes of the rooms.

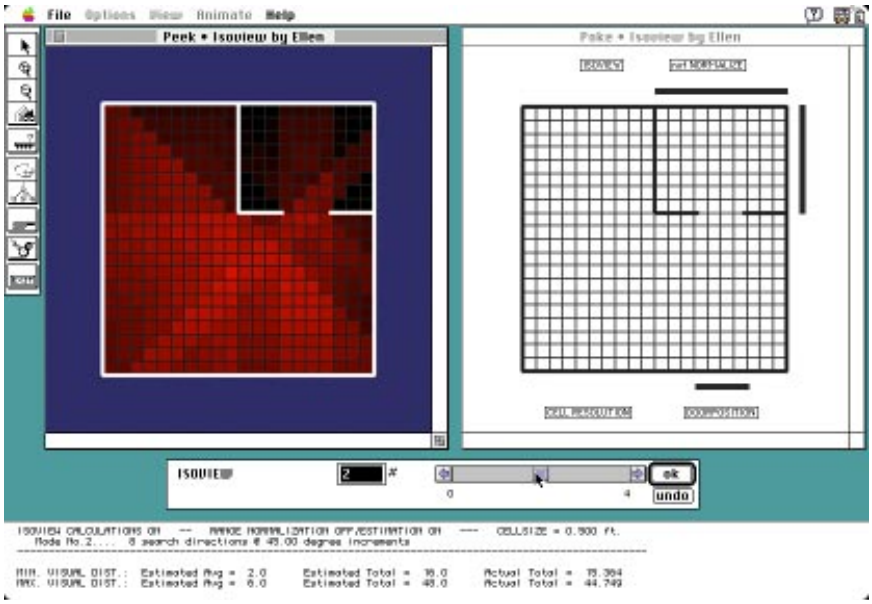


Figure 3. Gradient implementation of Topdown Isoview shows openness intensity in a floorplan. Left: graphic display. Right: user interface. Bottom: message to user.

A floorplan in Isoview is overlaid on a square grid whose resolution the user can set. Isoview allowed designers four degrees of grid resolution from a 10x10 to a 40x40 grid. Isoview's definition of "openness" is the average distance to the nearest surrounding

walls. Isoview computes the openness value for each grid cell and displays them simultaneously as color values in the floor plan (see figure 3).

The designer can change the granularity of the angular calculations. The crudest approximation uses a four directional $\pi/2$ radian (90 degree) calculation, in which the cell value is the average distance horizontally and vertically in both directions to the nearest wall. In addition to the simple 4 cardinal directions calculation, Isoview can use 8, 16, or 32 directions to perform finer grained calculations, corresponding to $\pi/4$, $\pi/8$. and $\pi/16$ radian angle intervals for distance-to-wall cell calculations. That is, for each cell x,y, Isoview calculates

$$V_{x,y} = \left(\sum_{0}^{k} d_i\right) / k \qquad k = 4,\ 8,\ 16,\ 32$$

where $V_{x,y}$ represents the value in cell (x,y); d_i represents the distances in the k directions to the nearest walls; and k represents the resolution of the calculation (4, 8, 16, or 32 directions).

Color intensity is used to display the cell values. Higher openness values are represented by light red, lower values are displayed as dark red. A normalized version shows the relative value of each cell with the most open cell displayed as bright red, and the least open cells as black.

3.3. POINT LIGHT SIMULATION - TCL-LIGHT

Tk/Tcl is a well known and widely used human computer interaction tool kit in the C language that contains a suite of graphic library calls and widgets for rapid prototyping.

Because of the embedded graphic abilities, the spatial analysis programs implemented in Tk/Tcl have a different flavor than our earlier efforts. They focus more on graphic display than numeric calculation. Below we describe two programs that used graphic display to support spatial analysis, both based on the analogy of view point as light source. The first is a point source light simulation called Tcl-Light; the second is a shadow casting simulation, Tcl-Shadow (Do 1994a).

Do's Tcl-Light used a light intensity analogy to model the perception of space. From a distance we perceive only an outline profile of a facade; as we move closer we start noticing the positions of windows and doors, then detailing and building materials. Thus, the intensity or level of our perception of a place depends on its distance from our view point. The closest objects to the view point (light source) will be brightest, and farther objects receive less light and therefore appear dimmer.

Tcl-Light simulates the perception of space through a gradient lighting display (figure 4). A gradient fading circle with decreasing intensity toward the perimeter is drawn around each view point. The designer can adjust how far the circle reaches and the intensity gradient to represent different individuals' spatial awareness. Similar to illuminating a room by adding more light sources, a user's understanding of a room can be strengthened by perceiving it from more view points.

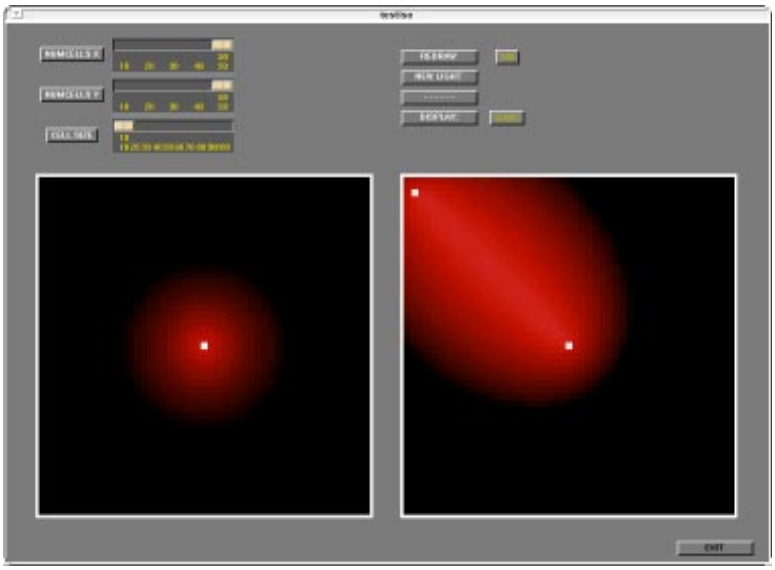


Figure 4. Gradient implementation of Tcl-Light models space perception as light sources.

3.4. SHADOW CASTING - TCL-SHADOW

Do implemented a second lighting analogy approach called Tcl-Shadow (Do 1994a) using a shadow casting technique: anything behind a wall receives a dark shading. Instead of painting the lighting effects from a light source like Tcl-Light, we simply plot shadows behind the walls.

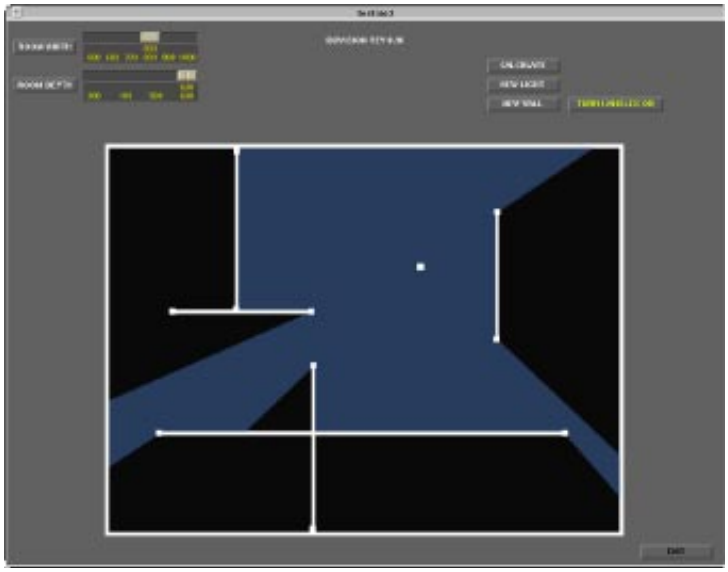
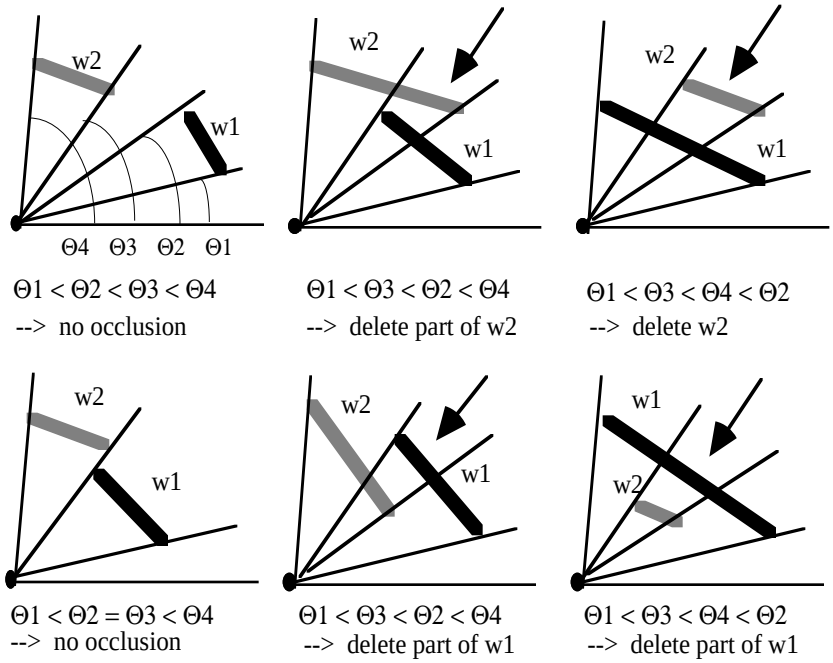


Figure 5. Tcl-Shadow displays an isovist by casting shadows.

Tcl-Shadow cast shadows on invisible spaces without the more complicated calculation of occlusions and visible field. By painting black shadows behind the walls, we are left with a field of visible surface, in effect displaying Benedikt's isovist (figure 5).

3.5. REMOVE OCCLUDED WALLS - AUTO-ISOVIST

Do implemented an isovist module (Do 1994b) in the AutoCAD drafting environment (using AutoLisp) to automate viewshed calculations.



$\Theta 1 = ang_{1,start}$ $\Theta 2 = ang_{1,end}$ $\Theta 3 = ang_{2,start}$ $\Theta 4 = ang_{2,end}$

Figure 6. Possible occlusions between two walls.

The Auto-Isovist program maintains a database of the walls designers draw. A chosen view point becomes the origin for sorting the walls. First the program determines and tags 'start' and 'end' points for each wall according to their angles to the x-axis. Then it sorts the walls by their startpoint angles in a sequence: $w_1, w_2, \ldots w_n$, such that the angles from the view point to their startpoints ($ang_{1,start}, ang_{2,start}, \ldots ang_{n,start}$) increase monotonically. Each wall w_i and the next wall in sequence w_{i+1} are then compared to find occlusions (figure 6). If the angle of w_i's endpoint is smaller than angle of w_{i+1}'s startpoint, then there is no occlusion, and the program moves on to compare the next walls w_{i+1}, w_{i+2}. If the angle ($ang_{i,end}$) of the end point of w_i is greater than the angle ($ang_{i+1,start}$) of the starting point of w_{i+1} then the wall segments between these angles are in occlusion. The next step checks the distance from the view

point to the two occluding wall segments to determine which wall is occluded, and deletes the occluded portion from its original wall. The program continues, comparing w_i and w_{i+2}, and so on. If the angle of the end point of a wall w_j is larger than the angle of the end point of a second wall w_{j+k}, and the distance from view point to w_{j+k} is larger than w_j, then the farther wall w_{j+k} is completely blocked by the nearer and hence entirely deleted from the database. If w_j is further, then the occluded part will be deleted from w_j, deriving two visible wall segments.

After this round of checking, we are left with only wall segments that are visible from the view point. The isovist field is then the polygon constructed by connecting the wall segments in order. The sum of the wall segments is calculated and labeled as the real-surface perimeter (figure 7).

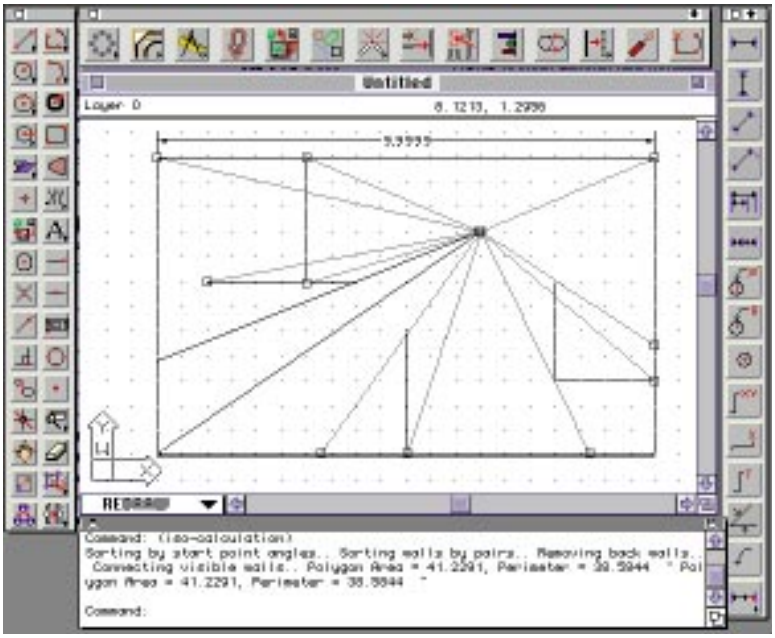


Figure 7. Isovist implementation in AutoCAD using startpoint angle sorting and occlusion checking .

3.6. FINDING INTERSECTIONS AND VISIBLE WALLS - MCL- ISOVIST

The last program we called IsoVist (Do 1995), named following Benedikt. It is implemented in Macintosh Common Lisp.

An internal database of walls is maintained and updated as the designer adds, deletes, and moves walls. The program constructs rays from the view point to the end points of all walls. The *find-intersections* routine computes all intersections of each ray with all other walls. It returns for each ray a data structure containing the intersection points and their distances to the view point, and with each intersection point, the intersected wall. Next it sorts the rays by their angles. Finally it constructs a representation for the

isovist by making triangular polygons bounded by the rays and the walls they intersect (figure 8).

It constructs triangles clockwise from the first ray. It starts a triangle from the view point, finding the first wall intersection point on the first ray. It checks if that wall intersects the second ray. If so, it finds the wall's intersection with the second ray, and follows the second ray back to the view point to complete a triangle. If not, it finds the next wall intersection of the first ray. As above, it checks this wall to see if it intersects the second ray, and so on.

Initially for debugging, we displayed the construction of light rays to all the intersections, and displayed the intersections as dots. Designers found this display of rays and intersections interesting. It reveals how the computation is done, so we left the debugging display on.

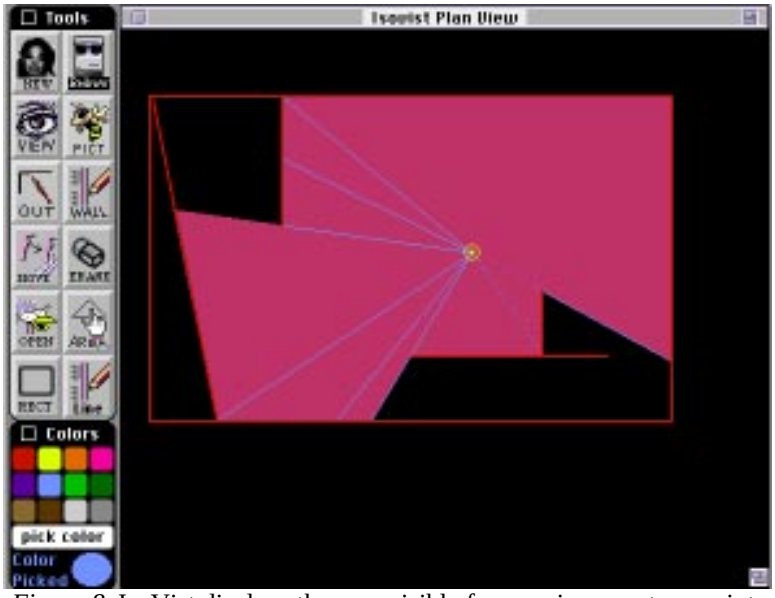


Figure 8. IsoVist displays the area visible from a given vantage point.

4. Discussion and Future Work

4.1. COMPARISON OF THE DIFFERENT APPROACHES

We have described six different computational implementations that support similar spatial analyses. Because we used different programming languages, platforms, data structures, algorithms and presentation methods, each tool addresses a different aspect of spatial analysis and encourages different ways of thinking about the problem.

Roughly, the six tools we described above can be classified into two categories: (1) accurate portrayal of the Isovist concept—this includes Tcl-Shadow, Auto-Isovist and MCL-IsoVist, and (2) a gradient and intensity approach—this includes Enclosure, Topdown Isoview, and Tcl-Light.

In the first category, the Tcl-Shadow, Auto-Isovist and MCL-IsoVist programs all portray the Isovist concept as defined by Benedikt, and they all consider and display a single view point depiction of the field of vision. Among them, the Tcl-Shadow version is a purely visual approach; it displays the isovist but does not maintain a computational representation of it. It simply displays the isovist field by shadowing out invisible places behind all walls. On the contrary, both Auto-Isovist and MCL-IsoVist all compute the visible field with sorting, intersection finding, and polygon constructions. Auto-Isovist takes a "behind" approach, finding occluded walls and eliminating them from the database to leave only visible walls. MCL-IsoVist takes a "front" approach, finding the nearest walls to construct visible triangles bounded by rays.

Second, the gradient and intensity approach programs all deal with more than one view point. Enclosure and Topdown Isoview both compute enclosure or openness values and they use a field display method to show the relative values of many different view points at the same time. This is quite different from the Isovist single view point approach. In other words, Isovist portraying programs plot a single isovist field at a time, while gradient programs like Enclosure and Topdown Isoview plot a multiple view point isovist field simultaneously (see figure 9).

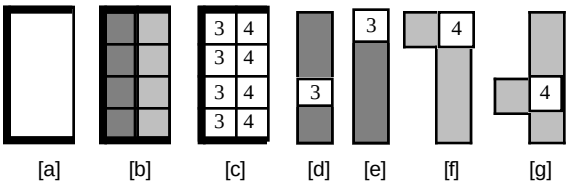


Figure 9. Topdown Isoview depiction of a space:
[a] space bounded on three sides,
[b] Isoview gradient display, $\pi/2$, 90 degree angular resolution
[c] cell distance to walls is 3 (2 up + 1 down) before normalization,
[d]-[g] eamples of cell value calculations: sum of distance to wall.

Enclosure and Isoview both measure the space quality in response to the bounding walls. Enclosure divides space into uneven subspaces, and computes a value based on the number of adjacent walls in each subspace (see figure 1). Subspaces of varied dimensions all have same enclosure value if they are bounded by same number of walls. Topdown Isoview takes a grid approach to divide space into even size subspace cells. A finer gradient of values occurs when grid or angular resolution is increased (figure 10). The crudest resolution of Isoview that only calculates cardinal directions in a four sided subspace, however, obtains similar results as the Enclosure method: both compute the same value for spaces with the same wall bounding conditions (figure 11).

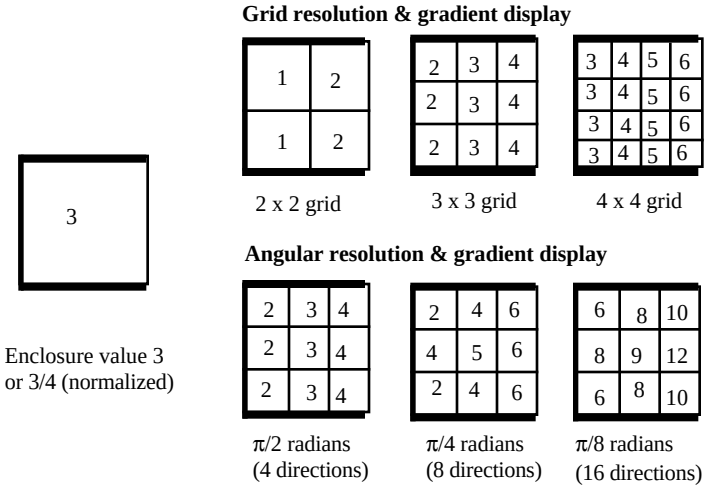


Figure 10. [a] Only one value derives from the number of walls of a bounded space in Enclosure, [b] The same space modeled in Topdown Isoview, gradient values with different granularity resulting from increased resolution: Top, grid resolutions from 2 x 2 to 4 x 4 with same angular resolution (4 directions); Bottom, increase of angular resolution (from 4 to 16 directions) with same grid resolution (3 x 3).

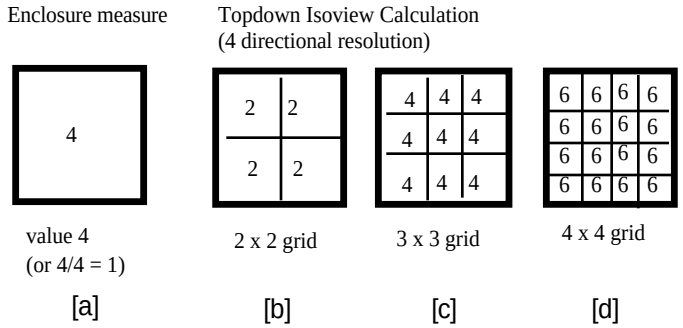


Figure 11. Enclosure and/or openness value for a space bounded by four walls. After normalization, all values in the space appear the same. [a] Enclosure computation derives value 4, [b - d] Topdown Isoview calculation with different grid resolutions.

The Tcl-Light approach is an outlying case; it supports multiple view points and also uses gradients like Enclosure and Topdown Isoview. However, it differs from these other implementations in that it uses gradient to display the fading intensity of perceptual space. Although Tcl-Light also supports single view point depiction like Isovist portraying programs it differs from these approaches in that it only plots a gradient perception circle instead of the shape of the visible field.

By using different programming environments and computation approaches, we have explored spatial analysis tools that address similar but varied concepts. The earlier

efforts by Gross aimed to understand spatial character through building computational tools (Enclosure). Later Do's series of programming experiments began with an attempt to portray multiple view point isovist fields on the same floor plan and resulted in a discrete gradient openness display program (Topdown Isoview). Seeing that gradient displays can help a designer to visualize relative openness values and that the isovist idea focused on single viewpoint, the point light analogy with intensity gradient was explored (Tcl-Light), which supported both single and multiple view points. A shadow casting approach to display isovist fields (Tcl-Shadow) then emerged from the lighting analogy. The discovery that the visible field from a single view point can be displayed by darkening spaces behind all the walls led to the eliminating walls "behind" implementation (Auto-Isovist). Finally, a "front" wall finding program (MCL-IsoVist) resulted from efforts to simplify computation.

4.2. FUTURE WORK

The suite of spatial and visual analysis tools described here were developed over a span of years, hardly a short term research project. It is interesting to see how the issues of spatial analysis were implemented differently. The approaches differ not only in the platform and programming language used, but also in their data structure, algorithm and presentations. We have reexamined the approaches of these programs to revisit the issues of computational support for spatial analysis.

We are interested in implementing these concepts in a three dimensional space, instead of the plan and sectional views supported by the various prototypes we have described. We are currently investigating the use of a head mounted display to provide a simulated walk through of a space in connection with traditional plan and section isovist analyses as outlined here. We would like to provide an analysis tool for designers to see and explore a simulated virtual environment. The project will combine not only visual presentation of a space walk through but also analysis and feed back of spatial perception.

Acknowledgements

Chenning Hsi, Ali Malkawi provided valuable programming help on random occasions, and Jean Wineman, John Peponis, Craig Zimring, and Aaron Fleisher provided insightful discussions.

We sadly dedicate this work to the memory of Wade Hokoda (1957-1997), artist and master craftsman of digital media, teacher, and friend, who died unexpectedly shortly before this paper was submitted for publication. Wade worked with Ellen Do on the projects described in this paper; many of the best insights herein are due to his sharp intellect and programming talent. His untimely death is a personal loss as well as an impoverishment of our field.

References

Appleyard, D., K. Lynch, and J. Myer. 1964. *View from the Road.* Cambridge, MA: MIT Press.

Asihara, Y. 1983. *The Aesthetic Townscape.* Cambridge, MA: MIT Press.

Benedikt, Michael L. 1979. To take hold of space: isovist and isovist fields. *Environment and Planning B* 6:47-65.

Benedikt, Michael L. 1984. Perceiving Architectural Space: From Optic Arrays to Isovists. In *Persistence and Change*, edited by W. H. Warren and R. E. Shaw. Hillsdale, N.J.: Lawrence Erlbaum.

Davis, Larry S., and Michael L. Benedikt. 1979. Computational Models of Space: Isovists and Isovist Fields. *Computer Graphics and Image Processing* 11:49-72.

Do, Ellen Yi-Luen. 1993. *Imaging the concepts of isovist field in design process -- Topdown Isoview -- a tool for spatial analysis.* Georgia Institute of Technology. Special Topics, independent study ARCH8183E.

Do, Ellen Yi-Luen. 1994a. *Design and description of form -- using tool command language Tk/Tcl to visualize isovist by lighting and shadow casting analogy*. Georgia Institute of Technology. Computer program ARCH8193A4.

Do, Ellen Yi-Luen. 1994b. *Isovist calculation in AutoCAD*. Georgia Institute of Technology. Computer program and independent study.

Do, Ellen Yi-Luen. 1995. *Visual Analysis through Isovist -- building a computation tool*. Georgia Institute of Technology & Sundance Lab for Computing in Design and Planning, University of Colorado, Boulder. Working paper and computer program

Hillier, W., and J. Hanson. 1984. *The Social Logic of Space*. Cambridge, UK: Cambridge University Press.

Ittelson, W.H. 1960. *Visual Space Perception.* New York: Springer.

Koile, K. 1997. Design Conversations With Your Computer. In *CAAD Futures '97*, edited by R. Junge.

Lynch, Kevin. 1960. *The Image of the City.* Cambridge: MIT Press.

Lynch, Kevin. 1972. *What Time is This Place?* Cambridge, MA: MIT Press.

Lynch, Kevin. 1976. *Managing a Sense of Region.* Cambridge, MA: MIT Press.

Mitchell, William J., R. Liggett, and M. Tan. 1990. Top-Down Architectural Design. In *The Electronic Design Studio*, edited by M. McCullough, W.J. Mitchell, and P. Purcell. Cambridge, MA: MIT Press.

Peponis, John, J. Wineman, M. Rashid, S-H. Kim, and S. Bafna. in press (Environment and Planning B). On the description of shape and spatial configuration inside buildings: three convex partitions and their local properties.

Peponis, John and J. Wineman, 1997. SPATIALIST-PARTITIONS, computer program for Microstation software, Copyright Georgia Tech Research Corporation.

Porter, Tim. 1979. *How Architects Visualize.* New York: Van Nostrand Reinhold.

Thiel, Philip. 1961. A Squence Experience Notation for Architectural and Urban Space. *Town Planning Review* 32 (April):33-52.

Thiel, Philip. 1981. *Visual Awareness and Visual Perception.* Seattle: University of Washington Press.