Proceedings of the Ninth International Space Syntax Symposium
Edited by Y O Kim, H T Park and K W Seo, Seoul: Sejong University, 2013

SPACE SYNTAX FOR GENERATIVE DESIGN:
On the application of a new tool

050

Richard Schaffranek
Vienna University of Technologie
e-mail : richard.schaffranek@tuwien.ac.at

Michael Vasku
Vienna University of Technologie
e-mail : michael.vasku@tuwien.ac.at

Abstract

Space Syntax can be used to predict, qualify and quantify aspects of human behaviour of different design proposals and the impact on their neighbourhoods. Therefore some design projects already incorporate Space Syntax during the design process stage (Dursun 2007) by proposing different design alternatives and continually testing and adopting them.

In recent years, it has become increasingly common to use physical characteristics like the wind, solar exposure or sound, as driving forces behind a generative design approach. Similar to the various physical models, Space Syntax analysis can be also, some simplifications assumed, processed automatically. Such an approach can be referred to as "inverse design", design by intents, used as constrains to a form and not as form by itself (Faucher and Nivet 2000).

This paper introduces "SpiderWeb" a new tool, under development since 2011, which enables various generative approaches based on Space Syntax. It provides the basics to copy some of the analysis available in "depthmapX". ("depthmapX" is a multi-platform software to perform a set of spatial network analyses designed to understand social processes within the built environment. "depthmapX" is based on the original "UCL Depthmap" and its developed by Tasos Varoudis at UCL's Space group. (Space Syntax Limited 2013)) Furthermore "SpiderWeb" extends it to other forms of modelling, applicable for urban and architectural design in three-dimensional space.

Keywords: *Architecture, Parametric - Generative Design, Grasshopper3D, Space Syntax*

Theme: *Modelling and Methodology Development*

Introduction

“SpiderWeb” - the tool introduced in this paper, - enables the use of Space Syntax in parametric modelling and its pursuit in computational design. Parametric design is a technique that enables a design depending on constraints. Constraints are defined associations and restrictions expressed in geometry and which help to fit specified requirements. Generally spoken, two types of constraints, i.e. geometric and dimensional, can be differentiated. Geometric constraints are used to control the relationships of objects in respect to each other. Dimensional constraints are used to control the distance, angle, radius and length values of objects. A parametric model forms the basis for computational design. Computational Design is a procedural, repeatable, mathematically definable process. It is based on preliminarily specified quantitative sets of rules that can be explored and developed in a framework. The framework has to allow variations to be generated and evaluated based on defined and undefined parameters, operations, or representations. This paper presents different approaches for using Space Syntax as a constraint in a computation design process.

Motivation

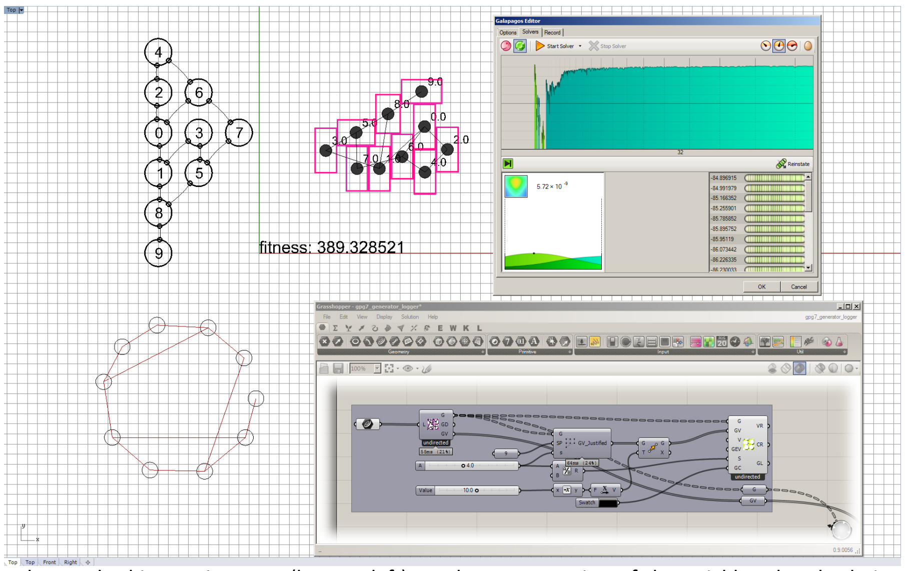

Figure 1: Background: Rhino 5 viewport. (bottom-left) graph representation of the neighbourhood relations between different rooms (walls, rooms, entrance). The graph is used to generate floorplan layouts (top-right). A justified graph (top-left) can be visualised to analyse the layout. Foreground: The ‘Grasshopper3d” definition, with some “SpiderWeb” components, to calculate the information in the viewport. In this screenshot a meta-heursitic solver (Simulated Annealing) is running to find a valid floorplan according to the input graph while minimizing the perimiter of the bounding box.

To be able to use Space Syntax in computation design processes it is necessary to introduce basic graph[1] operations inside a framework that allows modifying and generating geometry parametrically. For this the CAD software “Rhinoceros” was chosen - more precisely a plugin named “Grasshopper 3D”, which expands Rhino in the segment of parametric and generative modelling.[2] “Grasshopper 3D” is a visual programming language and therefore quickly to learn and easily to use. In such an environment “components” are wired together using various inputs to perform e.g. simple/ advanced calculations like the addition of values or the generation of a Voronoi diagrams.

[1] A graph is a mathematical representation of relations between a set of nodes. If two nodes have a relation they are connected by an edge. An edge has an edge weight. To calculate the shortest path (metric) the edge weight would need to be the Euclidian distance between the nodes connected by the edge.

[2] “Rhinoceros” and “Grasshopper3d” are developed by Robert McNeel & Associates. (www.rhino3d.com; www.grasshopper3d.com) Beyond “Grasshopper3d” other environments such as “Design Script”, currently under development by Autodesk are available. (designscript.ning.com)

Spiderweb

Components

"SpiderWeb" provides the following set of basic calculations/components to create, analyse and perform different operations on graphs (including visual graphs). In the most recent release the components are capable of handling directed graphs as well. Graphs (adjacency representation[3]) can be generated through different inputs e.g.:

Graph From Points – Creates a graph from a set of 3d points based on a distance. Points are understood as nodes of the graph. Two points are connected by an edge, if within a specified distance.

Graph From Lines – Creates a graph from a set of 3d lines. The lines represent edges, the endpoints the nodes of the graph. If set to "undirected" two edges are added to the graph for each line in the set. E_1(A, B), E_2(B, A) If set to "directed" one edge is added to the graph. E_1(A, B) This allows to model elements like escalators or a one-way street.

Graph From Cells – Creates a graph from a set of 3d polylines. Each shape represents a node, if two polylines share a boundary the nodes are connected by an edge.

Graph From Datatree – Creates a graph from a datatree[4]. The path indices of the datatree are interpreted as an edge list. E.g. this can be used to generate a graph of an axial map.

Dual Graph – Creates a new graph, where nodes represent the edges of the old graph. If two edges shared a node in the old graph the nodes representing these edges in the new graph are connected by an edge. The cost of the new edge is the angle between the edges or the original graph or if set to topologic 0 for no change in direction and 1 for a turn. In combination with the "Shortest Path" component this allows to calculate the simplest path. (Turner 2007), (Hillier und Iida 2005)

Visual Graph Grid – Creates a 2d point grid within the bounding rectangle of a polyline. Additional polylines can be added to represent obstacles. A list with integer values defines if a point is inside or outside of an area / an obstacle.

Further components allow the computation of different properties of a graph e.g.:

Breadth First Search - Preforms a breadth-first search[5] -algorithm on the given graph from a starting point. This can be used e.g. to calculate the integration of a given graph. (Hillier and Hanson 1984, 108)

Minimal Spanning Tree – Calculates the minimal spanning tree.[6]

Shortest Path – Calculates the shortest path (distance) from a root node to all other nodes. If there are multiple solutions (i.e. a regular grid) all possible paths are calculated.

[3] An adjacency representation describes a graph through its neighbourhoods. This can be done e.g. through a vertex list, where the neighbours of every vertex (node) are stored in a separate index list.

[4] A datatree is a special data structure within "Grasshopper3d". It is based on a sorted dictionary. A value in a datatree is accessed through a path. This path is the basis to create the graph e.g. Path: {0;2} will create an edge connecting node 0 with node 2;

[5] Breath-first search is a way to search through a graph from a given root node. In a first step it searches all neighbours of the root node and then continues with the first neighbour found. This is repeated until all accessible nodes, from the root node are visited.

[6] A tree is a connected graph without circles. A spanning tree of a connected graph is a sub graph, containing all nodes of a graph without any circles. In a minimal spanning tree, the sum of all edge weights is a minimum.

Recursive Shadow Casting - Preforms recursive shadow casting on a visual graph grid and outputs the visual graph. (Turner, Doxa, et al. 2001)

Shortest Path Between Points – Calculates how often a line segment is part of the shortest path between each pair of given starting points; e.g. this can be used to calculate the choice of a given graph; (Hillier and Iida 2005)

"Grasshopper 3D" also provides components where scripts can be written and executed. All classes[7] used within the described components are also accessible from within such script components.

Structure & Application

The described components allow copying some measurements, provided in "depthmap X" (Space Syntax Limited 2013), to "Grasshopper3d". To do so one needs in depth knowledge about those measurements, since the components perform only the most basic graph-based-algorithms. Less advanced users can apply clusters[8] provided to perform the different standard measurements like betweenness, closeness or connectivity. Furthermore, "SpiderWeb" works in the third dimension. There have been other attempts to provide Space Syntax measurements in parametric design environments. The Decoding Spaces Group (2012) provides some measurements in 2d, as predefined component for "Grasshopper3d". For generative design, however, this is too restrictive. The basic components, and even more the classes and functions that can be accessed from within the script components, provided by "SpiderWeb", on the contrary, allow adapting to various specific tasks:

The Euclidian position of the graph nodes and the adjacency representation of the graph are accessible as data streams within "Grasshopper3d", unlike the Decoding Spaces plugin. Therefore the graph can become more than the topological representation of a/the form but can be a/the genitor of form (see examples below: 3d Circulation and Digital Assumption, on circulation and identity).

Unlike the well-defined measurements provided by "depthmapX", with "SpiderWeb", models can be adapted to incorporate different measurements in models for early stage planning decisions. In this stage the exact spatial layout is unknown, but the few constraints known can help to make informed decisions (see example below: Space Allocation Planning and Communication Design).

Different models to describe aspects of human behaviour (e.g. Choice, VGA or findings on the topic of way finding (Hochmair and Frank 2000)) can be incorporated and accessed within the same model to describe relationships between different levels of perception (local – global, visual perception – pedestrian flow, space allocation – pedestrian flow (see example below: Labyrinth Runner example).

If appropriate, simplified and automatically generated models of spatial arrangements can be used, resulting in a loss of precision but a gain in speed. Further only the values needed, can be calculated (see example below: Labyrinth Runner example). This improves the speed of the calculation and enables the application of meta-heuristic solvers[9] searching through a vast amount of different solutions for specific properties.

[7] A class is an abstract model of an Object (e.g. a "GraphEdge"). A class can have different properties (e.g. start, endpoint and costs) and methods that can be executed (e.g. inserting a vertex in a "GraphEdge", returning two new Edges, or changing the costs of a "GraphEdge")

[8] A cluster can be understood like a block or a group in CAD programs. It combines a set of GH-components into a single component including inputs and outputs.

[9] Meta-heuristic solvers can be used to solve optimisation problems without knowing specifics about the problem. Therefore they can be applied to a variety of problems. Depending on the type, e.g. a genetic algorithm, the use of different strategies e.g. evolutionary principles to search possible solution trying to find the best solution.

Examples of generative applications

Network Improvement

Network improvements aim to find small but weighty changes in a given axial, segment or road centre line representation of any urban network in order to fit requirements defined by Space Syntax values. (Vasku 2013) There are several cases in which such an improved scenario might be useful, like controlling and redirecting movement in informal settlements after identifying areas of structural segregation. A redesign may cause positive effects, aiming to create a condition to allow slums to self-correct themselves. (Karimi und Parham 2012) (Karimi, Amir, et al. 2007) The task is to find efficient interventions that oppose "spatial discrimination" with minimal interventions on build stock. In a first step new connections between existing axial lines are created. In praxis, the consequence of a new street means the removal of existing buildings within the road's new area. Theoretically there are infinite possibilities for new line generations, so one has to limit their amount to a manageable amount. In this case new connections are drawn up to the maximal line length of 350 m by connecting only 10% of the most integrated axial lines at local radius with the 10% of most integrated axial lines at a global radius.

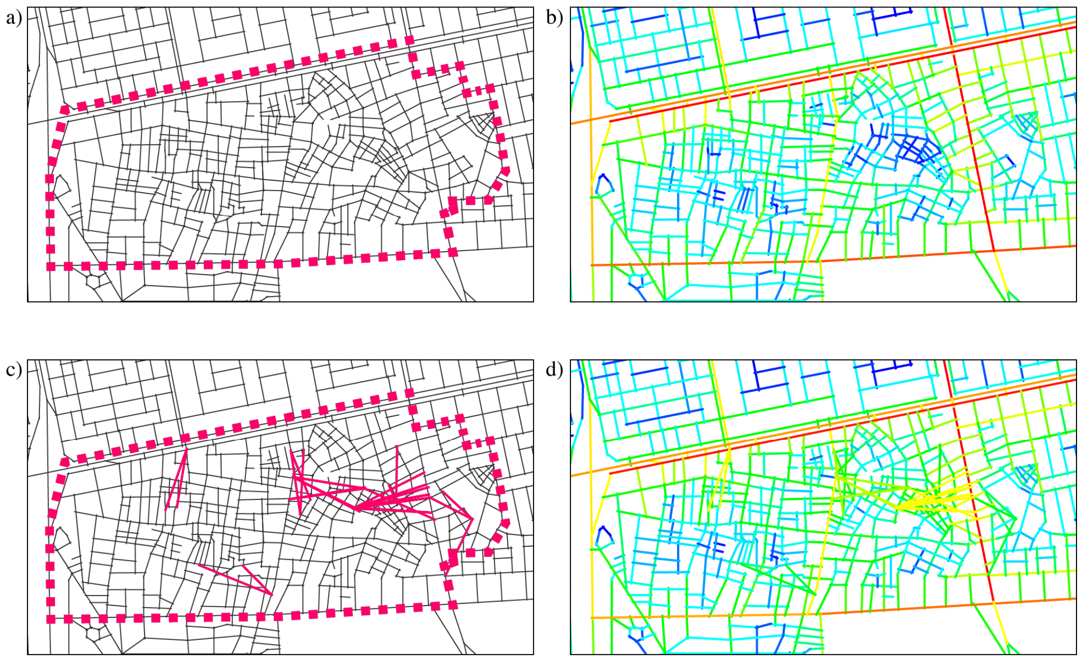

Figure 2: a) initial axial lines map (Karimi und Parham 2012) b) R5 integration analysis of initial axial map c) highlighted new axial lines for improvement d) R5 integration analysis of improved axial line map

For every single axial line that is generated, an individual Space Syntax integration analysis at a global radius is calculated. For every calculation different values are extracted and saved for later processing, such as: Generation ID (to identify the axial line), Intelligibility (a value defined by Space Syntax theory), 0.25 quantile of all integration values, and 0.75 quantile of all integration values. Here “SpiderWeb” in combination with Grasshopper allows the generation of new lines, their individual analysis and further an extraction of selected values in a batch process.

These values are stored in a table to enable their comparison and evaluation.

The IDs of the top one per cent of each measurement (descending order) are selected. The more often an ID is selected the larger the impact of its associated axial line.

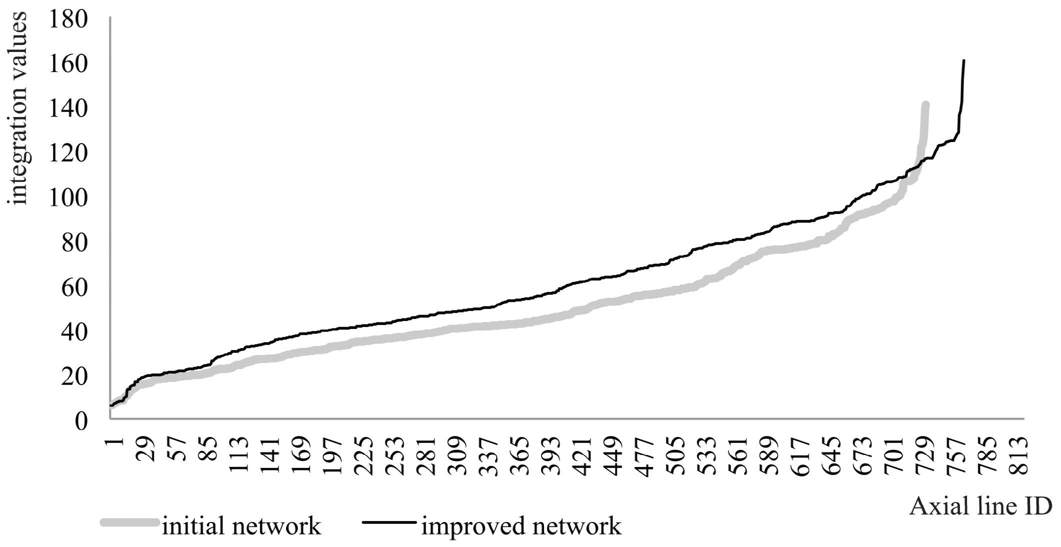

Figure 3: The graph compares integration values of the system before and after the intervention. As new lines were generated to improve the network, the total amount of axial lines is increased (x-axis).

Figure 3 compares values obtained by an integration analysis of the system with and without the calculated intervention. It clearly shows the significant improvement of integration, which is equally distributed throughout the entire spectrum. That implies, that the upgrading effect measured by integration effects most settlers without privileging certain locations. The values of each axial line within the informal settlement and in its direct neighbourhood are displayed in ascending order for their integration value.

Labyrinth Runner

Not only measurements proposed by the Space Syntax community can be included in a generative-parametric model with "SpiderWeb", but also other assumptions like way finding strategies in unknown networks, as described by Hochmair and Frank (2000) can be utilized to describe aspects of human behaviour through a model. Such a model can be used to generate labyrinths in which one can get lost. For this example three different measurements have been used to define the fitness of a solution, for use with a meta-heuristic solver.

Let $G(V,E \in G)$ be the graph representing the Labyrinth and $P = (P_0 = A, P_1, P_2, ..., P_n = B; P_{0..n} \in V)$ the shortest path from A to B. The costs *c(p)* (sum of changes in direction) for a trip, along the shortest path, from A to B must be as high as possible. The more the shortest path p converges to a space filling curve, the higher the costs *c(p)*. The average of Euclidian distances *d(p)* from all nodes part of the shortest path p from A to B must be as high as possible. This contradicts the way finding strategies of humans described by Hochmair and Frank (2000). In order to create a network-like structure, the more edges *n(p)*, not part of the shortest path from A to B, the better. The larger the fitness $f_2 = c(p)*d(p)*n(p)$ is, the harder to find the path through the labyrinth. The generated labyrinths were used as levels for a casual game.[10] The player paths were traced in order to analyse and verify the generated spatial layout. The traces showed that if in an unknown network a decision to change direction is only made if there are no alternatives. Some players played the same level more often in a row improving their understanding of the spatial layout constantly. But when they replayed the level, after a few minutes, they got lost again.

[10] The game can be played at http://wp10478398.server-he.de/LabyrinthRunner. To get access to the collected data please contact the author.

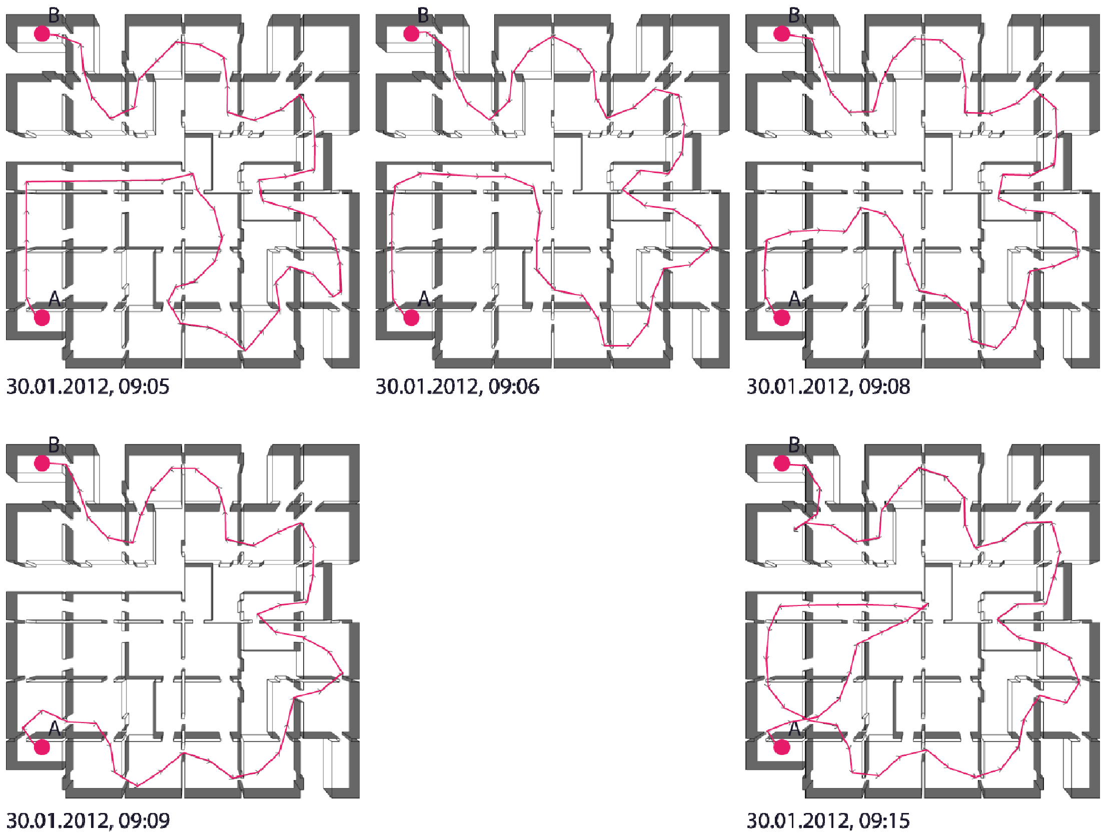

Figure 4: Showing multiple runs by a player nicknamed "LeDove." The six minutes inbetween the last two plays of the Level "RoomToRoom 6x6" "LeDove" tried to solve other levels. When he replayed this level after six minutes he partly lost orientation.

Further the data collected made it possible to propose a simple procedural algorithm to generate labyrinths. This simple algorithm could also be used to generate layouts with more than two endpoints or within different boundaries. (Schaffranek 2013)

Digital Assumption, on Circulation and Identity

This example continues to explore the possibilities outlined in the Labyrinth Runner example using different fitness functions. It was developed together with students within a studio project. The students where provided with a parameterised-line-based -model of a building. The task for the students was to use the model and define an algorithm that would generate geometry from the line system, which should help to orientate and define the identity of the different zones of circulation. To generate these differences, numerical outputs of different measurements were used.

In the parameterised-line-based model, each line would represent a certain area of the building (e.g. a square of about 4 by 6 meters). It could be set to one of four states: 0-line is deleted, 1-line is flat, 2-line is going up, 3-line is going down; Utilizing a meta-heuristic solver, different properties (fitness functions) could be searched for, within this simplified line system e.g. shortest path between all points, largest distance between local centres, highest difference between the choice value of the most used line and the average choice value.

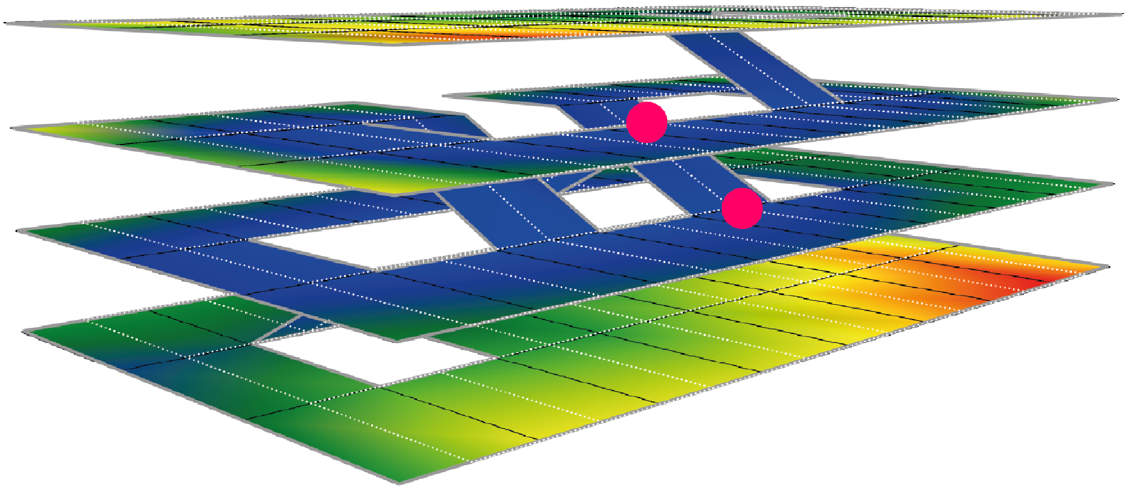

Figure 5: Example with the largest distance between local centers (pink circless); A double loop emerged as circulation, generating two areas which are closer to all other points than their neighbours; white dotted lines:underlying graph for calculating the different properties (e.g. avg distance between all points); blue: smallest avg. distance, red: heighest avg. distance to all points;

The model of the building would be considered inadequate for an accurate analysis, but through simplification it was possible to calculate the fitness in an acceptable time (<<1s/solution). Compared to the project Labyrinth Runner the different fitness functions in this example were rather simple but still produced interesting result. The shortest path would connect each storey of a building with two flights of stairs leading in the opposite direction. The largest distance between local centres generated a loop-like circulation with two local centres, from which all other areas not part of the loop could be accessed.

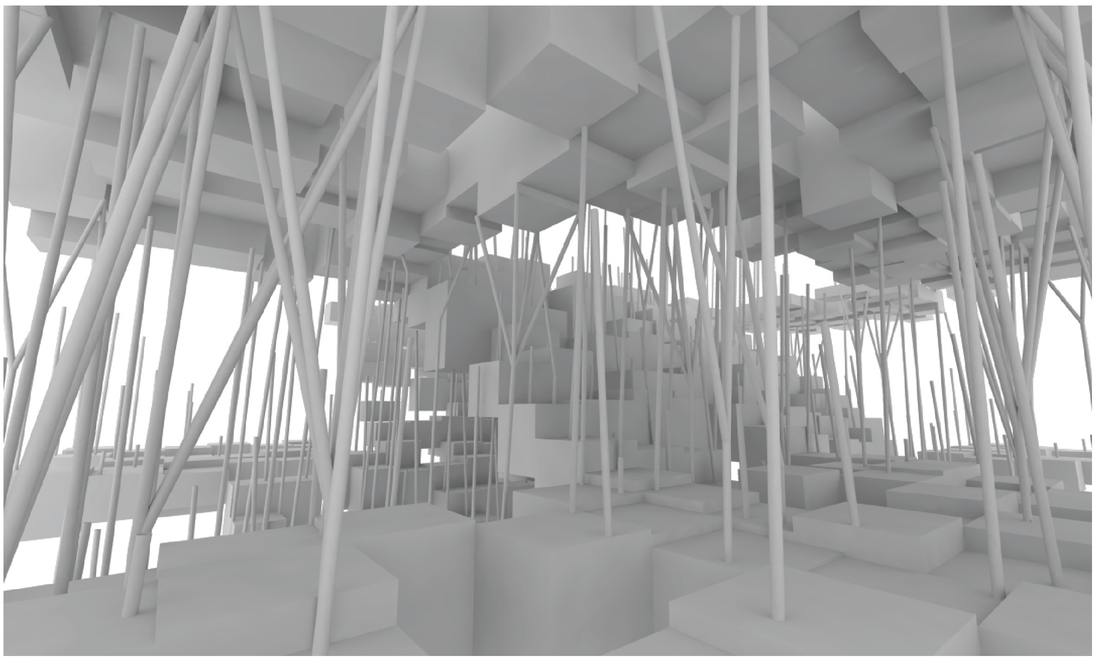

Figure 6: Geometry sample generated by students (Benedikt Aschauer, Werner Scheuringer); Differences in roughness of the floor and the density of the “construction” define the difference in usage (choice) and should enable orientation in the generated three dimensional geometry.

3d Implementation

Furthermore, the implementation in a 3D CAD environment enables the extension of the theory to three-dimensional space. First applications already revealed interesting observations like the "three storey suggestion": Applying angular segment analysis (Hillier and Iida 2005) in multi-storey buildings seems to respect that people tend to take an elevator for crossing more than three storeys. An angular segment analysis calculation for an elevator, entering the elevator horizontally, moving vertically and exiting horizontally, shows that the costs result in 180°. Moving from one level via a staircase to the next level amounts in 60°, assuming a slope of 30° for standard staircase. By this calculation the crossing of one or two floors cost 60° respectively 120°. This is less than the resulting costs when using an elevator (180°).

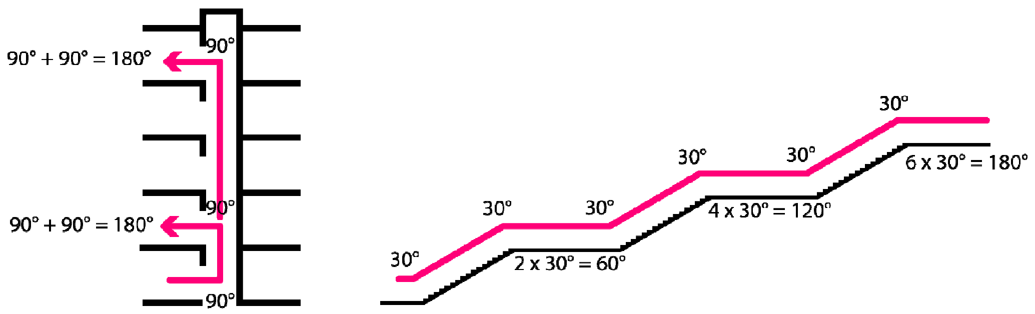

Figure 7: (left) section of a building showing paths using an elevator considering angular changes. (right) section of a building showing paths using a stair case considering angular changes.

Three floors amount to the same costs and for more than three floors the costs of using the elevator is always smaller than the stairs.

This seems to confirm the general assumption that people might move between floors, up to three floors, on foot, if the staircase is accessible and use elevators to cross more than three storeys.

Space Allocation Planning and Communication Design

In information driven environments e.g. universities or research centres, it is desirable to facilitate informal communication between users. As K. Sailer (2007) suggested, attractors and walking distances have a strong impact on the movement patterns within workplace environments. As already shown (Figure 5) this can be used as a trigger to design a new circulation pattern. In this sample, ideas from Space Syntax are not used to analyse an already well developed design but to incorporate the theory in an early stage of the design process.

The scenario is the re-allocation of different departments of a university in an existing building already used by the university. The building was constructed in the early 19th century, consisting of several connected wings forming a series of different sized courtyards. Access to the different rooms of the building is provided by staircases (mostly at the connection points of two or more wings) and corridors. The total area that has to be re-allocated is a little higher than the area actually provided by the building (This is mainly because of a suboptimal circulation).

The idea was to showcase the possibilities of space allocation planning in an early design stage, not only balancing the areas but to generate circulation patterns that would assist users of a building to meet incidental. Furthermore this could be used to place attractors for informal communication (such as coffee machines, public seating …) at the right places, using minimal floor area and maximizing effects.

Therefore an abstraction of the building had to be generated. The possible circulation within the building was simplified through a model of axis (corridors, staircases and elevators). If an axis would represent a corridor of a wing the available cross floor area was assigned as a property (represented in the model by simple rectangles).

The re-allocation of spaces/usage to a specific part of the building could be done by hand (moving a rectangle representing a room e.g. lecture hall, office space, toilet… into a rectangle representing the available space) or automatically by different simple allocation heuristics. The allocation of spaces to an axis has two consequences:

- Case1: There is enough space left to form a corridor connecting the two endpoints of the wing.
- Case2: There is not enough space left to form a corridor. The spaces allocated to this wing can be accessed from either endpoint but no through movement is allowed.

When placed by hand the model updates automatically and the impact on the circulation is displayed right away.

To understand the impact of the spatial allocation on the building’s usage the distances to the entrances and an adapted version of choice (metric) are computed. Instead of calculating the values, for choice, based on all shortest paths between all edges of the graph, the shortest paths between all allocated spaces are used. (The entrances to the building are also assumed as starting/endpoint of possible paths) Paths are weighted due to the size of the room they represent (simply assuming that a larger room would provide for more people).

Running the different allocation heuristics a series of randomized solutions are generated and a 3d preview is stored for further investigation. The different allocation heuristics would guarantee enough space, in certain areas of the building, to form a corridor (Case 1). One heuristic would ensure that all areas on a randomly chosen floor could form a corridor, in another heuristic these areas are randomly chosen or the areas of a certain wing would be only filled with rooms to such extend that a corridor would form on all levels. For each allocation heuristic 100 different solutions are generated and visually examined. For most of the results the usage was centred in the courtyards between the different wings of the building.

The most promising results were achieved with an allocation heuristic that would ensure corridors across only half of the building in one level and the other half three stories higher. In this case it was possible to lift the area most used (highest “choice” value) to an upper floor in several of the 100 different solutions (e.g.: Compare Background Figure 8). The average distance between the rooms and to the entrances of the building, was similar for all allocation heuristics.

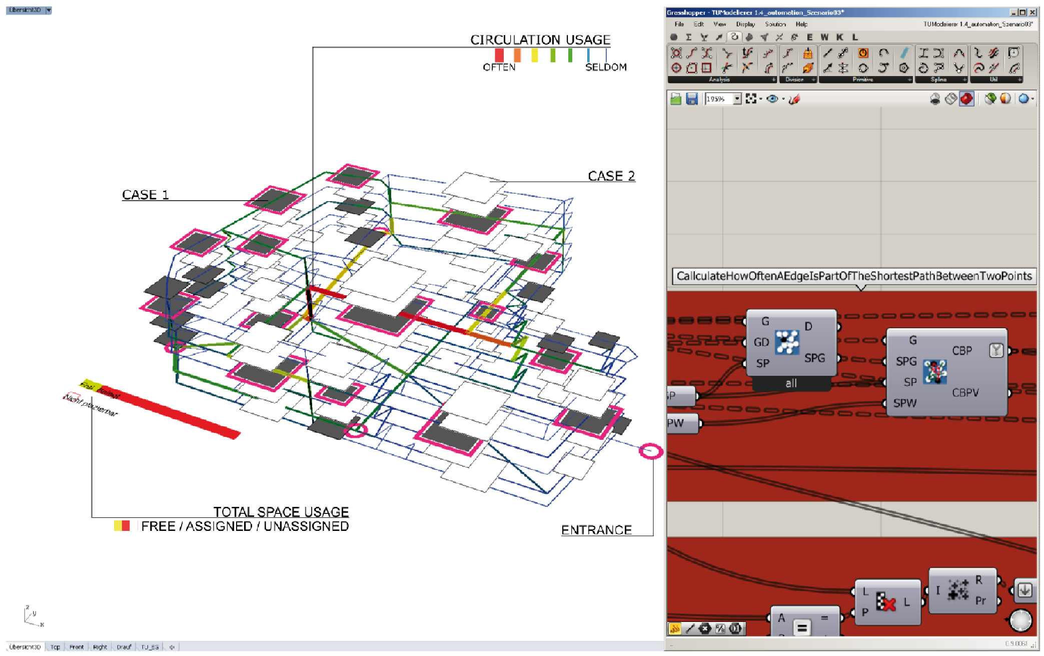

Figure 8: Background: Rhino 5 viewport, showing a simplified version of the TU-Wien (lines: wing-axis; rectangles: associated cross floor area). Rectangles with a pink boarder are guranteed, by the allocation heuristic, to have enough space left to form a corridor. Grey rectangles represent wings where there is enough space left and white rectangles have to little space left to form a corridor connecting the two endpoints of the wing. The axis display the usage, from blue (seldom) to red (often). Foreground: The 'Grasshopper3d" definition, with some "SpiderWeb" components, to calculate the information in the viewport.

As can be seen in Figure 8 not all of the spaces that would need to be allocated into the building could be placed by the simple heuristic. This only shows how pressing the issue of available floor area is in this example, but it is something that can be solved with some more effort. But the heuristics do suggest basic principles, constraints, dependencies where to but the extra effort and allocate infrastructure aiding informal communication.

Conclusion

This paper covers just a fraction of the possibilities available by combining Space Syntax theory and other mathematical descriptions of human behaviour with different computational design methods in a three dimensional environment, but it clearly shows the variety of possibilities of such a combination. Beyond the different application scenarios that have been outlined this already suggested a few ideas such as the "three storey suggestion", which for the moment remains unproven.

For those planners who lack knowledge of the basic graph algorithms, clusters provide a simple access to Space Syntax analysis in a parametric design environment. However, for a sensible and innovative use of mathematical descriptions of human behaviour for generative design, an in depth understanding of the different theories, knowledge of the underlying math and of computational design methods in general is needed. As has been shown in this paper, the structure of the implementation is of special importance. This structure is the major difference to existing analytical software like "depthmapX", which in their analytical capacities exceeds the possibilities of "SpiderWeb". But generative design needs different components working together, not a single component preforming an analysis. This makes it possible to build models useable within a parametric environment.

References

Decoding Spaces Group. 2012. "DECODINGSPACES." http://www.decodingspaces.de/ (accessed 4 9, 2013).

Dursun, Pelin. 2007. "Space Syntax in Architectural Design." In *6th International Space Syntax Symposium*, edited by S. Kubat, Ö. Ertekin, Y. I. Güney, and E. Eyüboglu. Istanbul: ITU Faculty of Architecture. 056-1 - 056-12.

Faucher, D., and M. L. Nivet. 2000. "Playing with design intents: integrating physical and urban constraints in CAD." *Automation in Construction*: 93–105.

Hillier, B., and S. Iida. 2000. "Network and psychological effects in urban movment." In *5th International Symposium on Space Syntax*. 475-490.

Hillier, Bill, and Julienne Hanson. 1984. *The Social Logic of Space*. Cambridge, New York: Cambridge University Press.

Hochmair, Hartwig, and Andrew U. Frank. 2000. "Influence of estimation errors on wayfinding-decisions in unknown street networks – analyzing the least-angle strategy." *Spatial Cognition and Computation 2* (Springer Netherlands) (4): 283-313.

Karimi, K., and E Parham. 2012. "An evidence informed approach to developing an adaptable regeneration programme for declining informal settlements." In *8th International Space Syntax Symposium*. Santiago de Chile.

Karimi, K., et al. 2007. "Evidence-based spatial intervention for regeneration of informal settlements: the case of jeddah central unplanned areas." In *6th International Space Syntax Symposium*. Istanbul.

Sailer, Kerstin. 2007. "Movement in workplace environments – configurational or programmed?" In *6th International Space Syntax Symposium*. Istanbul.

Schaffranek, Richard. 2013. "Inventing Circulation Patterns using Available Metaheuristic Solvers." In *Computation and Performance – Proceedings of the 31st eCAADe Conference – Volume* 1, edited by Rudi Stouffs and Sevil Sariyildiz. Delft, the Netherlands. 347-355.

Space Syntax Limited. 2013. "depthmapX | Space Syntax Network." http://www.spacesyntax.net/software/ucl-depthmapx/ (accessed 04 09, 2013).

Turner, Alasdair. 2007. "From axial to road-centre lines: a new representation for space syntax and a new model of route choice for transport network analysis." *Environment and Planning B* 34(3): 539 – 555.

Turner, Alasdair, Maria Doxa, David O'Sullivan, and Alan Penn. 2001. "From isovists to visibility graphs: a methodology for the analysis of architectural space." *ENVIRON PLANN B*: 103-121.

Vasku, Michael. 2013. "Generative Improvement of Street Networks Based on Space Syntax." In *Computation and Performance – Proceedings of the 31st eCAADe Conference – Volume* 1, edited by Rudi Stouffs and Sevil Sariyildiz. Delft, the Netherlands, 367-374.