#159

GRASSHOPPER REACH ANALYSIS TOOLKIT:

Interactive parametric syntactic analysis

CHEN FENG
School of Architecture, Georgia Institute of Technology, USA
c.feng@gatech.edu

WENWEN ZHANG
School of City and Regional Planning, Georgia Institute of Technology, USA
wzhang300@gatech.edu

ABSTRACT

Peponis, Bafna, and Zhang (2008) proposed three measures of street connectivity—metric reach, directional reach, and directional distance—to evaluate the potential of access to individual road segment in a street network. A Java program has already been developed at Georgia Tech as an ArcGIS plug-in for calculating the above measures. Although the Java program is suited for analysing large maps coming in GIS file formats, the editing of the input data is often done with other software packages which impedes a smooth dialog between the design exploration and the spatial analysis. We recently developed a toolkit for conducting reach analysis based on the platform of Rhino Grasshopper, allowing users not only to control the various parameters for the analysis in an interactive manner but also to integrate the reach analysis as an essential part of the design formulation process. With the wide recognition and popularity of Rhino Grasshopper for doing parametric design in the design community, this toolkit can also be taken as an analysis component which produces and feeds the result back into the generative component to help find the satisfying or even optimal design solution. This paper explains the data structures and algorithms that we implemented to carry out the various kinds of reach analysis, including details about how we tackled problems encountered in analysing some special street configurations. At the end, it introduces the basic functionalities of the toolkit with a simple example.

KEYWORDS

directional reach, parametric analysis, space syntax, Rhino Grasshopper

1. INTRODUCTION

A range of computational tools has been developed to calculate the measures that are closely associated with or inspired by space syntax theory over the past thirty years. Some of the tools are standalone programs exclusively designed for performing space syntax analysis, such as Depthmap developed by Alasdair Turner at UCL, while some others are extensions to existing CAD or GIS software, such as Spatialist developed by John Peponis's research group at Georgia Tech and sDNA developed by Alain Chiaradia, Crispin Cooper and Chris Webster (Cooper & Chiaradia, 2015; Turner, 2007b). As specialized analytical tools, they share the same workflow: read the input dataset, compute relevant measures, and visualize or output the results in a meaningful way.

Recently, with the generative design approach becoming popular among architects and designers, several attempts have been made to incorporate space-syntax-related measures into the parametric modelling process (Nourian, Rezvani, & Sariyildiz, 2013; Schaffranek &

Vasku, 2013). Integrated with the computational design tools, the space syntax analysis—unlike in the traditional tools—is automated at the back end and the output measures are constantly evaluated to determine how the design should evolve according to a predefined algorithm. Most of the existing tools that fall into this category have been developed as plugins for Grasshopper, a graphical algorithm editor for Rhino.

Acknowledging the growing trend of employing generative design and parametric modelling during the design process and given the popularity of Grasshopper as an effective modelling platform serving that purpose, we developed Grasshopper Reach Analysis Toolkit. The toolkit consists of a series of Grasshopper definitions which not only enables an interactive parametric syntactic analysis but also facilitates communication with other user-defined Grasshopper components to do computational design. We are aware of the several existing Grasshopper tools which are capable of performing space syntax analysis, such as SpiderWeb and Decoding Spaces Components (Bielik, Schneider, & König, 2012; Schaffranek & Vasku, 2013). However, to our best knowledge, the tool presented here is the first Grasshopper plugin developed to compute the reach measures that were proposed by Peponis, Bafna, and Zhang (2008).

In this paper, we explain the data structure and the algorithms that we implemented to carry out the reach analysis. We also illustrate a few scenarios that require special attention through hypothetical examples. In the end, we introduce basic functionalities of the toolkit with a simple real-world example.

2. CONCEPTUAL BASIS FOR REACH ANALYSIS

Specifically, we incorporated the following reach measures in our tool: (1) metric reach, (2) directional reach, and (3) directional distance per length (DDL). The conceptual basis and formal definitions for the above measures can be found in Peponis et al. (2008).

To simply put, the metric reach of a particular point in a given street network can be defined as the street length that is accessible from that point within a specified network distance range. The directional reach of a particular point can be defined as the street length that is accessible from that point within a specified number of direction changes. The DDL of a particular point in the street network can be simply calculated as follows. First, split the total length of the street network into a set of street lengths, with each length representing the amount of street length that is accessible from that point within a certain number of direction changes. Second, factor in the cost of direction change needed to access each portion of street length by multiplying the two together. Finally, add up the weighted lengths and divide it by the total length of the street network.

While we require the input data to consist only of straight lines—as implied by the definition of the directional reach—we used the midpoint of each 'line segment' instead of 'road segment' to compute the average reach measures of a street network. Here, the term 'line segment' refers to one of the smallest straight line segments that the drawing could be deconstructed into, depending on how the drawing is constructed and represented originally. The term 'road segment' refers to the straight line or the chain of straight lines that lie between two adjacent street intersections or that extend from one street intersection but left disconnected at the other end, constituting a dead-end street. The adoption of the line segment as the unit of reach analysis also presents a major deviation from the early version of *Spatialist-Lines*, which is a GIS plugin developed by Zongyu Zhang at Georgia Tech for conducing reach analysis. There are certain advantages and disadvantages associated with this choice. We will explain that later in this paper.

3. DATA STRUCTURE AND ALGORITHMS

We used Dijkstra's algorithm to find the shortest path between the midpoints of two line segments in a given street network (Cormen, Leiserson, Rivest, & Stein, 2009, pp. 658–662). Following the convention for analysing axial-line maps introduced by Hillier and Hanson (1984, pp. 93–94), we abstracted the original street network with a weighted undirected graph in which

each node represented a line segment (or more precisely, the midpoint of a line segment) while the edges represented the connections between those lines. The edge weights were assigned with either the metric distance or the directional distance between two nodes, depending on the type of reach analysis to be performed (Figure 1). We further represented and stored the above information with an adjacency list.

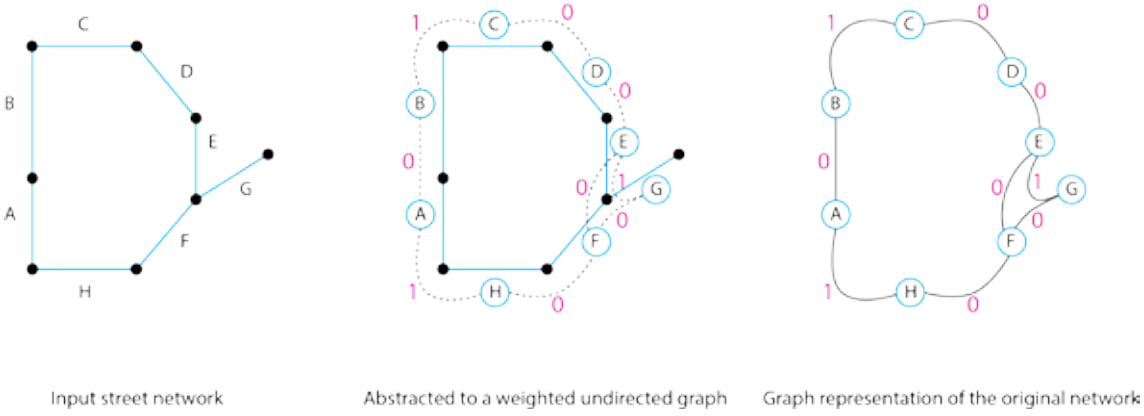

Figure 1 - Initial graph representation of the original street network. Here the edge weight indicates the directional distance between two nodes.

Although the undirected graph has proven to be an effective representation of the original street network in computing the metric reach, it could lead to error when computing the directional reach. Take for example the Y-junction formed by line segments E, F, and G in Figure 1. If we only count a deviation angle that is equal or greater than 90° as a direction change, then given the sharp turn to travel from E to G, one would naturally interpret the directional distance between E and G as one. However, in this case, if without any restriction on directionality, the Dijkstra algorithm would count the directional distance between E and G as zero instead of one because it "thinks" that by taking the alternative route E-F-G (instead of E-G), the sharp turn between E and G at the Y-junction could be avoided. One would immediately see the problem here: there is an implicit assumption that backtracking or making U-turns along F is cost-free.

The assumption is, of course, unjustified and questionable. Turner (2007a) proposed a solution for forbidding U-turns by incorporating 'back' links and 'forward' links for connections at each end of a segment. The way he handled this problem suited the purpose well for the angular segment analysis. However, instead of totally banning U-turns, we preferred to let the cost of taking a U-turn be a variable that could be explicitly controlled. As shown in Figure 2, we converted the original undirected graph into a directed graph as a more general solution.

We generated the directed graph by differentiating the directions of movement on each line segment. Take the line segment A in Figure 2 for example. Assuming the movement is bidirectional on A, we replaced it with two vector-like objects—A1 and A2—with the same endpoints but opposite directions. A1 and A2 are in principle connected, as represented by the two arcs (i.e., directed edges) pointing to each other in the bottom diagrams in Figure 2. However, one can explicitly adjust the cost of those connections (i.e., the cost of making U-turns) by assigning different edge weights onto those arcs. The two situations discussed above become two special cases now: if the arcs are weighted zero, then essentially it means U-turns are cost-free; if the arcs are weighted infinity, then essentially it means U-turns are banned.

Since we assume the movement on A is bidirectional, to capture the possibility of heading from A in both directions, we run Dijkstra's algorithm twice, with A1 and A2 as the source node respectively. For example, to find the shortest directional distance from A to G (with the assumption that the movement on G is also bidirectional), the algorithmic steps are as follows:

1. run Dijkstra's algorithm with A1 as the source node, find the shortest directional distance from G1 and G2 respectively, compare the two values and temporarily store the smaller one—a shorthand for this step could be *min(dist(A1, G1), dist(A1, G2))*;
2. run Dijkstra's algorithm with A2 as the source node, find the shortest directional distance from G1 and G2 respectively, compare the two values and temporarily store the smaller one—a shorthand for this step could be *min(dist(A2, G1), dist(A2, G2))*;
3. compare the temporarily stored results yielded from the two runs, and save the smaller one as the true shortest directional distance from A to G—a shorthand for this step could be *min(min(dist(A1, G1), dist(A1, G2)), min(dist(A2, G1), dist(A2, G2)))*.

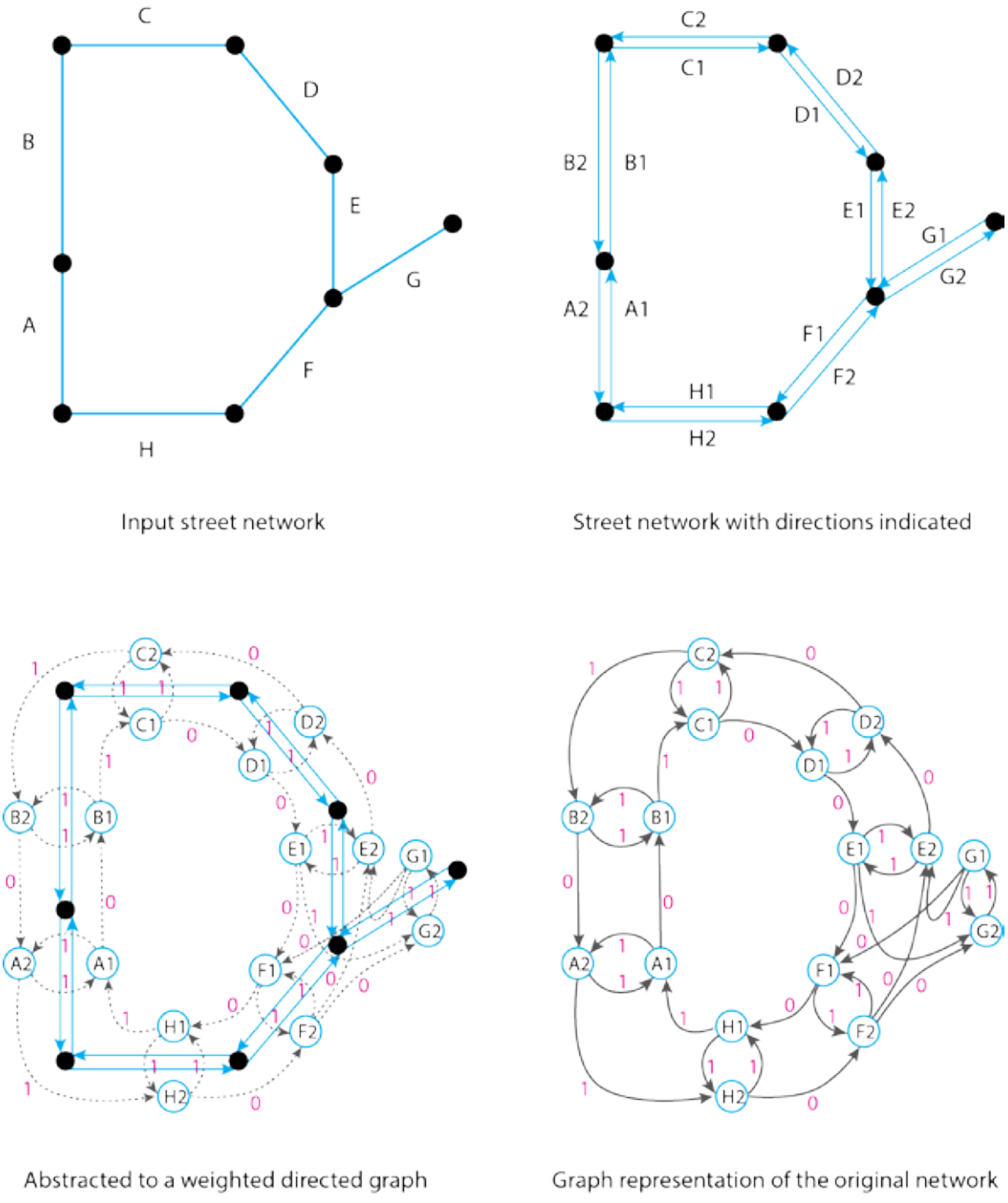

Figure 2 - Modified graph representation of the original street network for the directional reach analysis. Here the edge weight indicates the directional distance between two nodes.

Being capable of explicitly control the cost of taking U-turns, our algorithm can also be readily applied to a complex street network model where certain directions of traffic are restricted along the streets. In fact, it would only be a matter of editing the graph representation so that it reflects the restrictions on certain movement directions along the streets, without the need to change the core of the algorithm. Despite the potential benefits we mentioned above, we are aware of an obvious downside of the algorithm which is associated with the significant increase in the number of edges as compared to the undirected graph. It could be an issue if one is particularly concerned about the computational speed for running reach analysis on large street networks.

4. CAVEATS

In this section, we illustrate a few scenarios that require special attention when using the Grasshopper Reach Analysis Toolkit. Before we go into those issues, we first discuss the motivation behind choosing the line segment instead of the road segment as the basic unit of reach analysis.

4.1 LINE SEGMENT AS THE BASIC UNIT OF REACH ANALYSIS

Using the line segment is sometimes preferable to using the road segment as the basic unit of reach analysis—especially when the road segment is configurationally differentiated along its entire length.

Take the road segment AB in Figure 3 for example. Suppose that we are running the linear-reach analysis (i.e., the zero-direction-change-reach analysis) for AB and suppose that the angle threshold that we use to count as a direction change is relatively small so that traversing through the zigzag involves a series of sharp turns. Then we choose the midpoint of the road segment to be the starting point (as marked by the pink circle on the line segment cd) and see how much street length is accessible from that point without involving any direction changes. In this case, because one cannot go further beyond the endpoints of the line segment cd without taking turns, the linear reach of the road segment AB from its midpoint is essentially the length of cd. The linear reach value for AB that is calculated as such, however, would misrepresent the nature of the left portion of AB, which is a relatively long and straight line segment. Hence using the line segment as the basic unit of reach analysis here would be more appropriate.

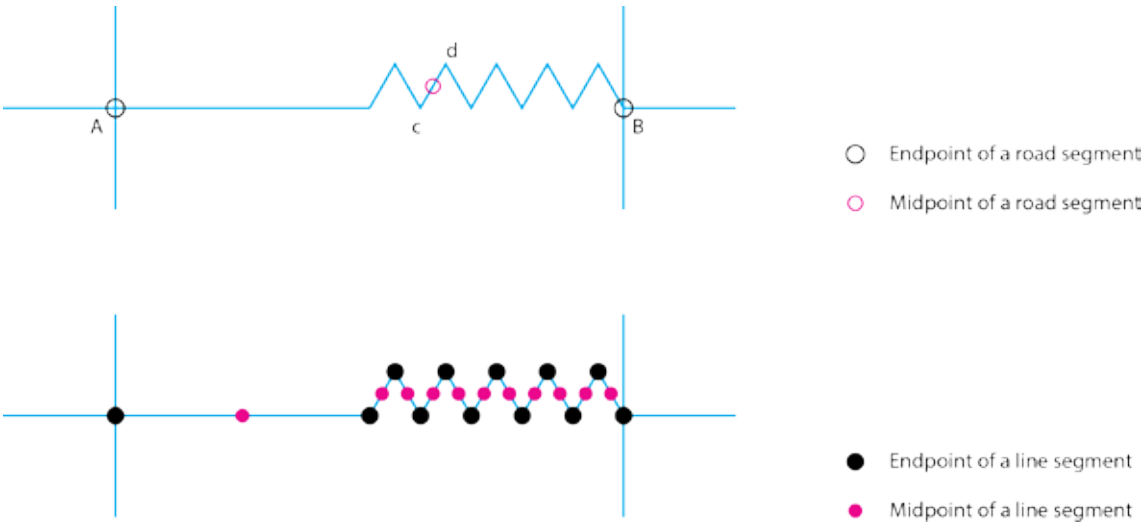

Figure 3 - Illustration of the difference between the road segment and the line segment.

We also notice that if the input street map was originally drawn with a CAD or GIS program, processing it into line segments usually involves less effort than processing it into road segments where special care is needed to ensure that the lines are joined properly between the street intersections.

4.2 TWO REALITIES

The issues that we bring up here have nothing to do with the soundness of the algorithms that we have designed for our toolkit. Instead, they are related to the quality of the input data of the street network and related to the theoretical question of how we should model and represent the spatial structure of our cities. Those issues essentially represent the conflict of two different realities—one reality regards the geometric shape of the road, while the other regards how people perceive and understand the structure of the street network. Here, we are more interested in bringing these issues to the table for a broader discussion than providing definite solutions.

One common issue is about how to model a road curve with line segments. In Figure 4, we illustrate different ways of modelling a sharp turn in the street, with increasingly higher fidelity to the geometry of the curve from left to right. If we consider a deviation angle that is greater than 30° as a direction change, then in Figure 4(a), it would take one turn to get from A to B; in Figure 4(b), it would take two turns; and in Figure 4(c), because none of the angles is greater than 30°, there would appear to be no turns! Although Figure 4(c), among the three, is the best approximation to the reality of the shape of the road, a directional reach analysis based on this representation may not reflect the reality of how a human being normally perceives and understands such a space.

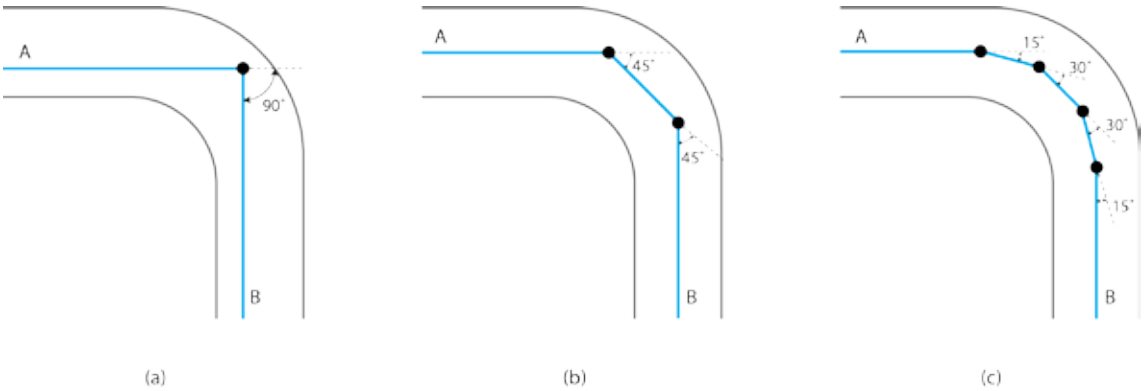

Figure 4 - Different ways of modelling a road curve.

In fact, the above observation also speaks of the "very-short-line-segments" problem mentioned by Peponis et al. (2008, p. 893)—that is, if very short lines are used, even a relatively sharp turn can resolve itself into a great number of smaller angles of deviation that might pass the test of a reasonably small threshold angle used to count as a direction change. As an expedient solution, they suggested to start accumulating the angle of direction change when the analysis encounters two or more consecutive line segments whose length is below a given metric threshold (Peponis et al., 2008, p. 893). We, however, did not implement this strategy in the current version of Grasshopper Reach Analysis Toolkit. We preferred to relay this issue to the users so that they can deliberately model the input street network in the way that suits best their purpose of analysis.

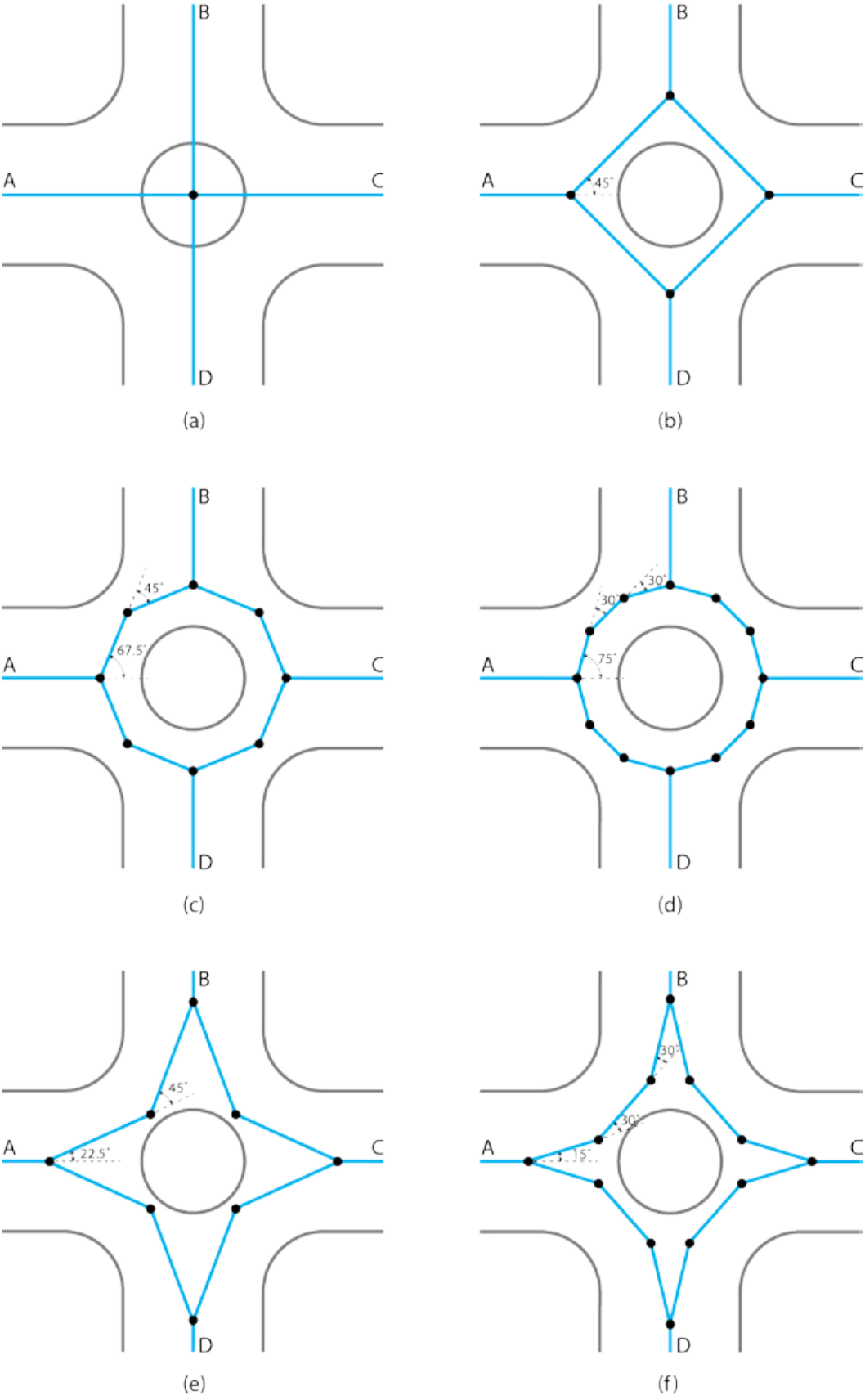

Figure 5 - Different ways of modelling the roundabout at a street intersection.

A related but more complex issue is about modelling the roundabout at a street intersection. Figure 5 illustrates a number of different ways of modelling the roundabout at a cross intersection. Suppose that we only count as a direction change if the angle of deviation is greater than 30° and here we are interested in the directional distance from A to B, C, and D, respectively. It is clear that different representations of the roundabout will yield dramatically different results: in Figure 5(b), it would take two turns to travel from A to either B or D, and an additional turn to get to C; in Figure 5(d), it would always take two turns to get to B, C, and D, and likewise it would take two turns to go through the roundabout and travel back to A itself—which is quite different from the representation in Figure 5(b) based on which it would take 5 turns to get back to A through the roundabout; in Figure 5(f), because the line segments closely follow the curves of the streets bounding the adjacent blocks instead of the roundabout, it would take no turns to travel from A to B and D, and only one turn to get from A to C. Figure 5(a) illustrates the simplest way to model such an intersection and defies the existent geometry of the roundabout. However, one might find that this representation indeed more accurately reflects the reality of people's perception or understanding of this space. Of course, a more complex modelling technique is needed if we would like to show further the real flow of traffic—a third reality.

The third issue occurs when modelling a cross intersection where the centrelines of the incoming streets are not perfectly aligned and do not converge at a single point. As shown in Figure 6(a), the centrelines of the two vertical streets hit the centreline of the horizontal street at two points that are only slightly apart. Nevertheless, based on the algorithm we presented, it would be registered as two turns as one travel from one of the vertical streets to the other because, in principle, one needs to take a turn to travel a distance along the horizontal street—no matter how small that distance is—then take another turn to get to the street on the other side. Since it is very unlikely for a pedestrian to perform the two turns to cross the horizontal street and even less likely to mentally register the two turns, we suggest editing the centrelines of the vertical streets so that they do converge at one point, as illustrated in Figure 6.

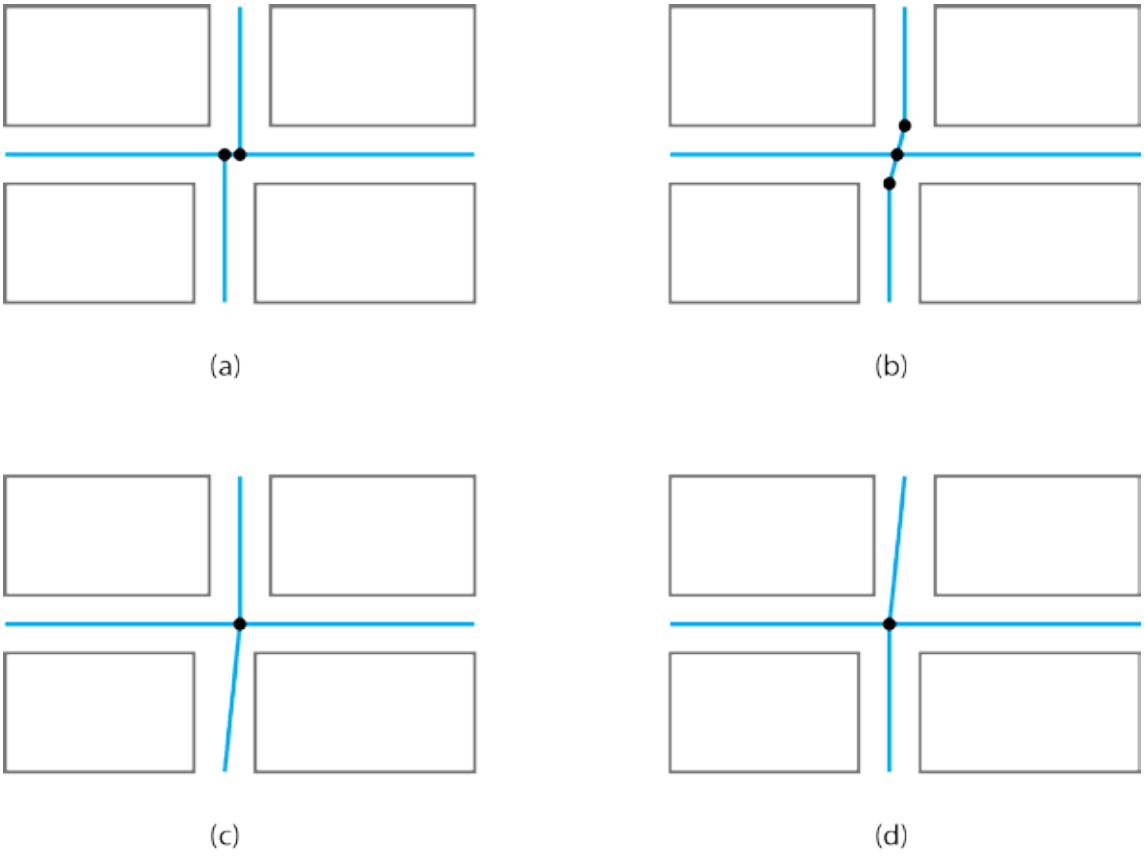

Figure 6 - Different ways of modelling a cross intersection where the incoming streets are not perfectly aligned.

5. BASIC FUNCTIONALITIES

The basic functionalities of the Grasshopper Reach Analysis Toolkit are summarized in Table 1.

Single-Source Reach Analysis			
Type of analysis	Input parameters	Output values	Visualization (optional)
metric reach	• line ID (i.e., the source) by typing or interactive selection • distance threshold	metric reach value for the selected line	• line ID tags • midpoint of the source • lines and fractions of lines within reach
directional reach & DDL	• line ID (i.e., the source) by typing or interactive selection • angle threshold • max number of direction changes	• directional reach value for the selected line • DDL value for the selected line	• line ID tags • midpoint of the source • lines within reach
Multiple-Source Reach Analysis			
directional reach	• line IDs (i.e., the sources) by interactive selection • angle threshold • max number of direction changes	• directional reach value for each of the selected lines • aggregated directional reach value	• line ID tags • midpoints of the sources • lines within reach
All-Lines Reach Analysis			
metric reach	distance threshold	metric reach value for each line in the map	lines coloured based on the reach value
directional reach	• angle threshold • max number of direction changes	directional reach value for each line in the map	lines coloured based on the reach value
DDL	angle threshold	DDL value for each line in the map	lines coloured based on the reach value

Table 1 - Basic Functionalities of the Grasshopper Reach Analysis Toolkit

We use the street centrelines of Gassin, a southeastern French commune, as a case study to show the range of reach analyses that could be performed with the Grasshopper Reach Analysis Toolkit in Figure 7.

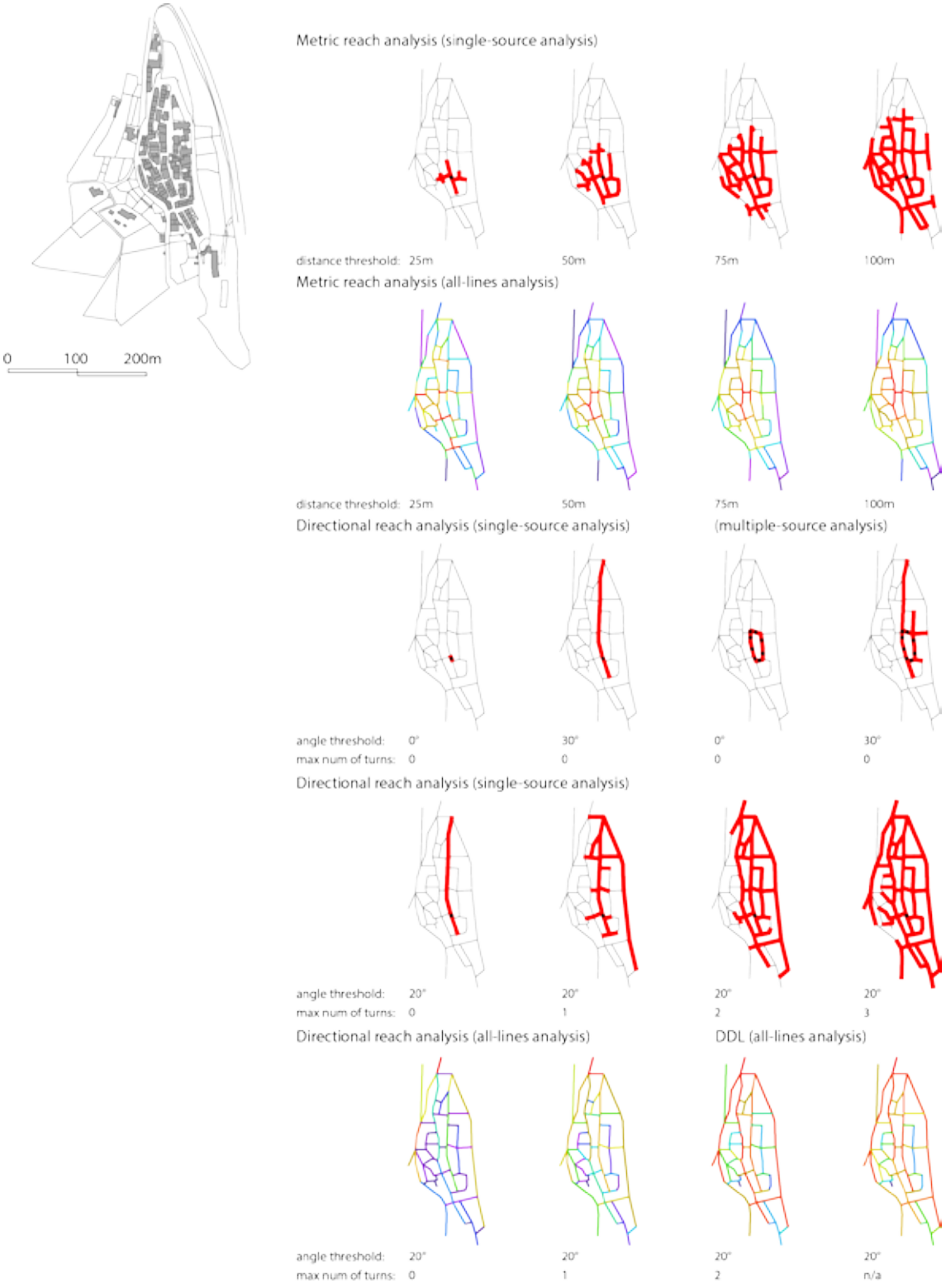

Figure 7 - Reach analysis of the street network of Gassin.

The users can choose different types of reach analysis and, more importantly, can parametrically control the analysis in a straightforward way. Developed as normal Grasshopper definitions, the toolkit is interactive and can be easily integrated with other Grasshopper definitions not only for pure analysis but also for design purposes. One possibility is to treat this toolkit as an analysis component and use it as part of a generative design scheme to arrive at a satisfactory or even the optimal design solution based on certain reach measures.

6. CONCLUSION

We have introduced the basic functionalities of the Grasshopper Reach Analysis Toolkit in this paper. We have also explained the data structure and algorithms that we designed and implemented in developing the toolkit. The motivation behind our work is two-fold: (1) to complement the original version of Spatialist-Lines to enable a smoother dialogue between design exploration and spatial analysis; (2) to open up the potential for integrating the reach analysis as an essential part of the design formulation process by developing the toolkit on a platform that is commonly used for developing computational design tools.

ACKNOWLEDGEMENTS

This work has been funded by Perkins+Will.

We would like to thank Monica Gentili and Matthew Swarts for giving us valuable suggestions about developing the algorithm for the directional reach analysis.

REFERENCES

Bielik, M., Schneider, S., & König, R. (2012). Parametric urban patterns: Exploring and integrating graph-based spatial properties in parametric urban modelling. Paper presented at the eCAADe 2012, Prague, Czech Republic.

Cooper, C., & Chiaradia, A. (2015). sDNA: How and why we reinvented Spatial Network Analysis for health, economics and active modes of transport. In N. Malleson, N. Addis, H. Durham, A. Heppenstall, R. Lovelace, P. Norman, & R. Oldroyd (Eds.), GIS Research UK (GISRUK) 2015 Proceedings (pp. 122–127). Leeds, UK: The University of Leeds.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). Cambridge, Massachusetts: The MIT Press.

Hillier, B., & Hanson, J. (1984). The social logic of space. Cambridge, UK: Cambridge University Press.

Nourian, P., Rezvani, S., & Sariyildiz, S. (2013). A syntactic architectural design methodology: Integrating real-time space syntax analysis in a configurative architectural design process. In Y. O. Kim, H. T. Park, & K. W. Seo (Eds.), Proceedings of the 9th International Space Syntax Symposium (pp. 048:001–048:015). Seoul, Korea: Sejong University Press.

Peponis, J., Bafna, S., & Zhang, Z. (2008). The connectivity of streets: Reach and directional distance. Environment and Planning B: Planning & Design, 35(5), 881–901.

Schaffranek, R., & Vasku, M. (2013). Space syntax for generative design: On the application of a new tool. In Y. O. Kim, H. T. Park, & K. W. Seo (Eds.), Proceedings of 9th International Space Syntax Symposium (pp. 050:001–050:012). Seoul, Korea: Sejong University Press.

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

Turner, A. (2007b). UCL Depthmap 7: From isovist analysis to generic spatial network analysis. Paper presented at the 6th International Space Syntax Symposium, Istanbul Technical University, Istanbul, Turkey.