GI_Forum 2016, Vol.1
Page: 217-230
Full Paper
Corresponding Author:
anita.graser@ait.ac.at
DOI: 10.1553/giscience2016_01_s217

Integrating Open Spaces into OpenStreetMap Routing Graphs for Realistic Crossing Behaviour in Pedestrian Navigation

Anita Graser
AIT Austrian Institute of Technology, Vienna, Austria

Abstract

Map data for pedestrian routing and navigation provided by OpenStreetMap is getting more and more detailed, but current approaches often fail to take advantage of available information. This paper addresses the issue of integrating open spaces, such as squares and plazas, into pedestrian routing graphs to support realistic crossing behaviour. We evaluate different approaches to solving this issue, including skeletonization algorithms as well as approaches for wayfinding in digital worlds, and recommend that – for pedestrian navigation applications – the visibility graph approach should be preferred over the commonly-used medial axis or straight skeleton approaches, since it provides direct routes, which are more realistic and better suited for pedestrian routing applications.

Keywords:

pedestrian, routing, navigation, open spaces, OpenStreetMap

1 Introduction

In contrast to car drivers, pedestrians do not follow clearly-defined linear routes; they can take short cuts, and they are allowed to use dedicated areas such as pedestrian zones. Even though the characteristics and needs of pedestrian routing and navigation have been the topic of numerous publications (Bauer et al., 2014; Elias, 2007; Millonig & Schechtner, 2007; Retscher, 2004), current routing applications still fail to take into account pedestrian-specific issues, such as creating suitable routing networks.

In the real world, streets, squares and plazas are areas with different widths and lengths, and pedestrians can move – more or less freely – in these spaces. In GIS, this pedestrian infrastructure is often modelled as lines (roads, sidewalks ...) and polygons (public open spaces, such as squares and plazas), but purely polygon-based datasets (often derived from CAD plans) exist as well. For routing and navigation applications on the other hand, all infrastructure is usually modelled as networks, where network nodes represent intersections and network edges represent roads and other paths. This network-based street model can be used directly to compute shortest paths using well-known standard algorithms. This

modelling approach has proved to be suffcient for car routing and navigation since cars move in a linear fashion along predefined lanes, but pedestrians move with a much higher degree of freedom, particularly when crossing open spaces.

The integration of open spaces into routing networks – which can be found in many current implementations – is generally effected by treating the outline of the polygon representing the open space as additional edges for the navigation graph. This leads to unrealistic routing results, where the pedestrian route follows the outline of the open space, often on a zig-zag path, as illustrated in Figure 1 by the examples on Rockefeller Plaza, New York City, USA and Josefsplatz, Vienna, Austria. While these examples show GraphHopper's pedestrian routing, other routing services, such as OpenRouteService, Mapzen and osm2po, return similar results. These sub-optimal routes, which wind their way along the outline of the open space rather than cross it on a more direct path, affect both the visual representation of pedestrian routes as well as their textual description in the form of directions or navigation instructions.

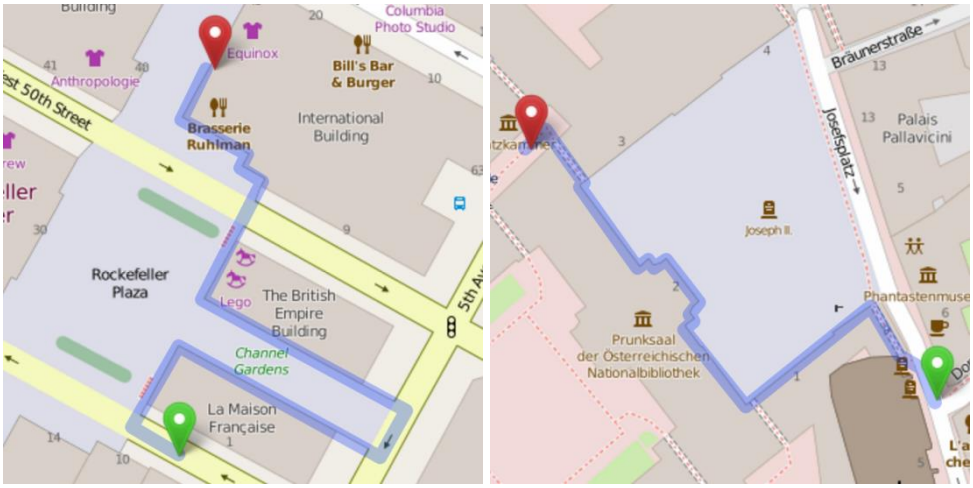

Figure 1: Example routes illustrating the issue of the typical integration of open spaces into pedestrian routing graphs: Rockefeller Plaza, New York City, USA (left), Josefsplatz, Vienna, Austria (right); screenshots from openstreetmap.org on 2016-01-03

This issue of unrealistic routes through open spaces is discussed, for example, in Bauer et al. (2014), who then present a post-routing integration approach: each node along the route is checked for whether it is located on the open space outline, and then the first and the last node on the same outline are connected in order to allow the pedestrian to cross the open space in the most direct fashion while taking potential obstacles into account. This approach thus requires post-processing to be performed for every route before the results are presented to the user. If the post-processing module is implemented as part of the router, users have to be able to deal with routing results which contain links, which are not part of the routing graph. This post-processing adds complexity to the routing process, which would not be necessary if squares and plazas were integrated into the pedestrian graph in advance. An example of this approach is presented by Elias (2007), who converts polygon

representations of streets from a cadastral map into a routing graph by transforming the areal features into line structures by extracting the polygon's medial axis. While this approach works well to convert elongated street polygons into linear features, results for squares and plazas are not optimal, as discussed in more detail in the following section.

Our contribution in this paper is a novel approach to generating pedestrian routing graphs using pedestrian infrastructure data in the form of both lines and polygons from OSM. Our approach solves the problem of how to integrate open spaces into a routable network-based street model in a way that makes it possible to compute pedestrian routes with realistic crossing behaviour. We define realistic crossing behaviour as: the pedestrian route crosses the open space on the most direct path possible. For empty squares or plazas, this means that the route goes from the entrance point to the exit point on or near the direct line of sight, without unnecessary detours. If there are obstacles such as buildings or fountains, these should be accounted for and the pedestrian route should find a direct path past the obstacles.

Section 2 presents different approaches to creating linear representations of areal features, including skeletonization algorithms and approaches for wayfinding in digital worlds. Section 3 discusses the advantages and disadvantages of these linear representations for the use case of pedestrians crossing open spaces. Section 4 presents our approach to integrating linear and areal features into a consistent graph for pedestrian routing. Section 5 sums up the results and presents topics for further research.

2 Linear representations of areal features

The current state of the art for generating linear representations of areal features includes approaches that have been suggested in the context of real-world routing and navigation applications (Elias, 2007; Roberts et al., 2005), as well as approaches better-known in the domains of (pedestrian) simulations and wayfinding in digital worlds (Lozano-Pérez & Wesley, 1979; Snook, 2000).

One approach that has been suggested in the context of real-world routing and navigation applications is to extract the medial axis – for example, to derive street centrelines (Elias, 2007; Haunert & Sester, 2008; Roberts et al., 2005). The medial axis is defined as the set of points that have more than one closest neighbour on the polygon boundary. It consists of straight lines and second-order lines that result from reflex angles of the polygon, and it can be computed in linear time using, for example, the algorithm by Chin et al. (1995). In practice, the medial axis is often approximated by the Voronoi diagram of the polygon boundary points, since the medial axis is equal to the line Voronoi diagram of the polygon edges (Haunert & Sester, 2008). To generate a good approximation, the polygon outline needs first to be densified. Figure 2 (left) presents the results of this approach (implemented using the QGIS Processing Modeler (Graser & Olaya, 2015)), with the outline densified to one point per meter. Another approach for constructing medial axes takes advantage of the fact that the Voronoi diagram is the dual graph of the Delaunay triangulation. Therefore, the approximated medial axis can also be constructed by connecting the centres of adjacent triangles of the triangulation, as demonstrated, for example, in Elias (2007).

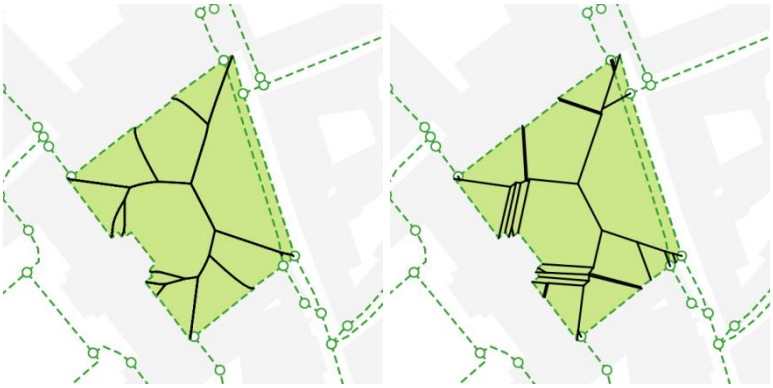

Figure 2: Two examples of algorithms used in GIS applications to generate polygons' centrelines: medial axis (left), and straight skeleton (right); centrelines are drawn in black, dashed green lines show existing pedestrian network

Another approach found in GIS literature is known as the straight skeleton method, introduced by Aichholzer et al. (1996). For example, Haunert & Sester (2008) use straight skeletons to extract road centrelines from road polygons. A popular illustration of the concept of straight skeletons is to imagine a roof which covers the polygon and consists of planes which rise with a constant slope from each polygon edge. Figure 2 (right) shows the straight skeleton computed using the 2D Straight Skeleton and Polygon Offsetting package from the CGAL (Computational Geometry Algorithms Library) (Cacciola, 2015). The construction of the straight skeleton is based on a stepwise shrinking of the polygon by translating each of its edges at a fixed rate, keeping sharp corners at the reflex vertices. During the shrinking, each edge collision is resolved, and skeleton edges to the collision points are created. The straight skeleton can be computed in sub-quadratic time using the algorithm by Eppstein & Erickson (1999). As illustrated by Figure 2 (right), the straight skeleton is quite similar to the medial axis. The main difference is that straight skeletons ensure continuous straight lines while the medial axis is more curved (Haunert & Sester, 2008).

Another GIS approach for finding routes through areal features is to compute least-cost paths over a cost surface (Smith et al., 2009). To adopt this approach to pedestrian routing in an urban environment, the cost surface can be configured to contain pedestrian-friendly areas, such as sidewalks and open spaces (as surface cells with low costs), areas that can be crossed but are less attractive, such as roads (medium costs), and areas that cannot be traversed, such as buildings (extremely high costs), as illustrated in Figure 3. This approach differs from previously presented approaches in that it does not transform the areal features into a network model, but instead uses a cost surface to compute the least-cost path. Since we are interested in generating a network-based routing graph, this approach was not pursued any further.

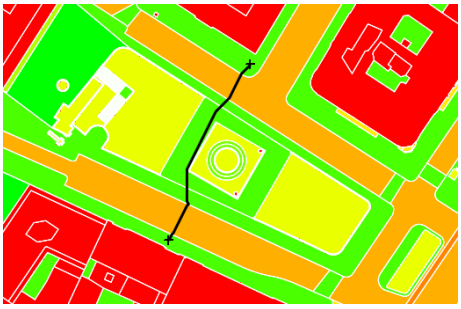

Figure 3: Least-cost path over a cost surface (colours from green = low cost, to red = extremely high cost); example using GRASS GIS r.walk at Dr.-Karl-Lueger-Platz, Vienna

In pedestrian simulation and wayfinding in digital worlds, such as in computer games, other approaches have been developed. For example, in pedestrian simulation, it is common to create a graph from a regular grid, where graph edges connect centres of empty (not occupied/blocked) grid cells. This approach is straightforward to reproduce in a GIS environment by generating a line grid and removing all lines which are not completely within the open-space polygon. Figure 4 (left) shows a square grid example with a cell size of five meters. Other commonly used grids include triangle and hexagonal grids.

Navigation meshes (Snook, 2000) are another approach used for simulations. A navigation mesh is a collection of convex polygons where, for example, simulated pedestrians can walk around without obstructions on straight lines between the polygon centres, edges and/or corners (depending on the chosen algorithm). Van Toll et al. (2011), for example, describe how to generate navigation meshes from medial axes by connecting the medial axis with line segments to the nearest obstacle. An open source implementation of navigation meshes is provided by Mononen (nd).

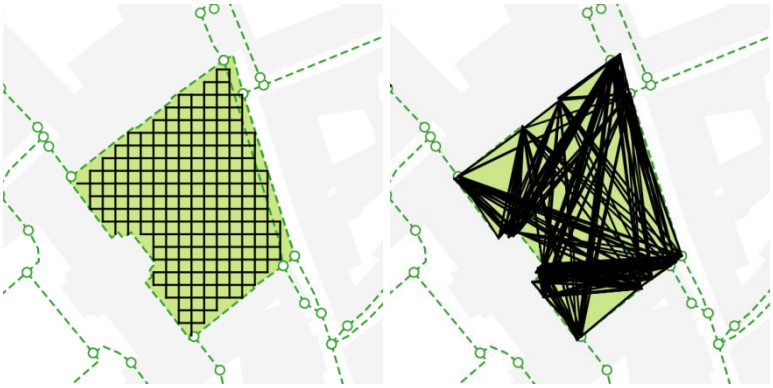

Figure 4: Two examples of algorithms used in pedestrian simulation and wayfinding in computer games: regular grid with 5m grid size (left), and visibility graph (right)

To describe complex open spaces, for example in indoor navigation scenarios (Liu & Zlatanova, 2015) and pedestrian simulation (Matyus et al., 2015), the construction of a

visibility graph is often suggested, instead of using regular grids or navigation meshes. In a visibility graph, each edge represents a visible connection between given point locations (Lozano-Pérez & Wesley, 1979). As depicted in Figure 4 (right), for our use case, these point locations are the nodes that make up the polygon outline. This approach is straightforward to reproduce using GIS APIs, by first generating a complete graph connecting all nodes and then checking which of the resulting lines are completely within the corresponding polygon. All lines which are partially or completely outside the polygon are removed, leaving us with the polygon's visibility graph.

The approaches presented in this section provide different solutions to the problem of generating linear representations of areal features. These approaches differ with respect to the characteristics of the resulting linear representation of the open space, as well as with respect to the complexity of the method used to generate this representation. The following section describes how selected approaches (including medial axis, straight skeleton, regular grid and visibility graph) can be used to integrate open spaces into OSM pedestrian routing graphs, and discusses the advantages and disadvantages of the different approaches with respect to the use case of realistic crossing behaviour.

3 Evaluation of linear representations for crossing open spaces

To create a continuous pedestrian routing graph, which includes edges for crossing open spaces, it is necessary to integrate the linear representation of the open spaces into the routing graph in a way that ensures that all entrances and exits to the open space are connected to the edges crossing the space. The connections can be constructed by either 1) connecting each entrance/exit of the open space polygon with the nearest point on a crossing edge, or 2) by intersecting crossing edges and extension edges, which are constructed by extending the incoming pedestrian graph edges, as described in Elias (2007). This step can be skipped for the straight skeleton and the visibility graph, since these approaches automatically ensure that all polygon border nodes – and thus all entrances and exits to the square – are connected.

Figure 5 presents the routing results for crossing the open space at Josefsplatz, Vienna on the medial axis, straight skeleton, regular square grid with a grid size of 5m, and the visibility graph. The resulting routes differ in shape and length, with the longest route on the regular grid and the shortest route on the visibility graph.

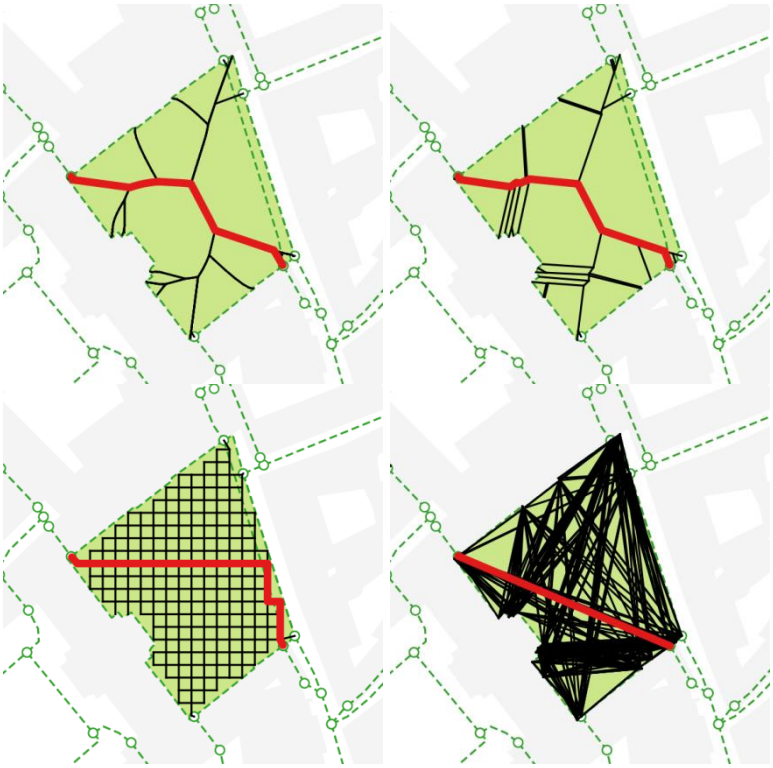

Figure 5: Routing results (red lines) for different linear representations of areal features (black lines): medial axis with route length 100m (top left); straight skeleton with route length 102m (top right); regular 5m grid with route length 116 (bottom left); visibility graph with route length 91m (bottom right)

As Figure 5 shows, neither medial axis nor straight skeleton produces optimal results with respect to realistic crossing behaviour, which we defined as the pedestrian being able to cross the open space on the most direct path without unnecessary detours. Neither returns the most direct path possible but follows an S-shape, which makes the paths longer (100m and 102m respectively) than the shortest path on the visibility graph (91m).

In a regular grid, a shortest-path algorithm will usually find multiple routes of the same length. The most direct route will usually run on a zig-zag path from start to destination, but other routes with fewer turns and of the same length are possible as well. Figure 5 (bottom left) shows one example with a route length of 116m, which makes it the longest routing result in this comparison.

Furthermore, it is also worth noting that the direction changes in the routes analysed so far pose a challenge for potential consecutive navigation instruction algorithms, which evaluate changes in the route direction to provide turn instructions, as described, for example, by Dräger & Koller (2012).

The comparison shows that visibility graphs are preferable with respect both to the complexity of the implementation as well as to the properties of the resulting routing graph, since they provide direct connections without unnecessary detours. In the following section,

we discuss the integration of open spaces into the pedestrian graph using visibility graphs, with a focus on the OSM data structure.

4 Integrating open spaces into OSM pedestrian routing graphs

The OSM data model consists of nodes, ways and relations. Nodes can be used to represent point features, such as traffic lights or trees, but at the same time they are the building blocks for ways. Ways, in turn, can represent linear features, such as roads or rails, or – in the case of closed ways – polygonal features, such as squares and plazas. Finally, relations can describe complex features, such as polygons with holes, or logical relationships, such as all the features belonging to a bus route. Each node, way and relation has a unique OSM ID which is used to reference it within the database.

Since OSM raw data is not directly routable, it is necessary to pre-process the data to generate a routing graph. This process requires knowledge of the OSM data structure in order to preserve bridges and tunnels while inserting valid intersections, as well as to correctly resolve the various restrictions on different modes of transport, such as one-ways and turning restrictions for vehicles, or roads closed to pedestrians such as motorways and some tunnels. The main concept for making OSM routable is to split the ways at valid intersections. The convention is that two roads intersect if they share a common node. A way representing a bridge can cross an underlying road, but they don't share a node at the intersection point. A detailed description of the necessary pre-processing steps is outside the scope of this paper, but the OSM Wiki (Wiki, 2015a) provides a good introduction to the topic. Existing tools which implement the necessary data pre-processing steps include the open source tools osm2pgrouting (Wiki, 2011), Open Source Routing Machine (OSRM) (Luxen & Vetter, 2011), and the freeware tool osm2po (Möller, nd).

In OSM, open spaces for pedestrians are modelled using closed ways with the tag combination area=yes and highway=pedestrian (Wiki, 2015b). Open spaces and incoming road and pedestrian ways also share nodes where the road intersects the closed way that describes the open space. Our example at Josefsplatz in Vienna is a closed way consisting of 31 nodes (http://www.openstreetmap.org/way/29067366). Other open spaces can be much more complex, for example, nearby Stephansplatz, which is modelled as a relation (http://www.openstreetmap.org/relation/152813) with one closed way describing its outer border and another closed way describing the inner border, consists of 234 nodes due to the detailed mapping of the cathedral at its centre, as depicted in Figure 6.

Figure 6: OSM pedestrian area Stephansplatz, Vienna, Austria (screenshot from openstreetmap.org on 2016-01-01)

It is worth noting another common pattern found in OSM: in some places, such as Stephansplatz, OSM mappers have manually created additional pedestrian ways (marked by dotted lines in Figure 6) traversing the open space, in order to make it possible for default routing applications to find routes through complex open spaces. The same can be observed in other maps besides OSM, including Google and Bing Maps, but the coverage is far from complete and many open spaces have not been manually enhanced by adding such helping edges. However, these additional edges do not negatively affect the approach presented in this paper.

Figure 7 presents our base algorithm for integrating open spaces into a pedestrian OSM routing graph by constructing corresponding visibility graphs. The visibility graph is generated by creating a complete graph and removing all edges which are not completely within the open-space polygon. The edge start- and end-node IDs, which are needed by the routing algorithm, are derived from the corresponding original OSM node IDs. Since each open-space polygon is processed independently, this algorithm can be parallelized for faster processing but, since this is a pre-processing procedure which only needs to be executed once to generate the graph, this is not necessarily a priority.

```
1  foreach open space polygon P do
2  |  extract all nodes N of P;
3  |  create edges between all N;
4  |  foreach edge between all N do
5  |  |  set edge from_id and to_id to the respective node IDs;
6  |  |  avoid duplication of edges by checking node_from_id < node_to_id;
7  |  end
8  |  remove edges which are not within P;
9  |  insert all remaining edges into the routing graph.;
10 end
```

Figure 7: Base algorithm for the integration of open spaces into a pedestrian OSM routing graph

A proof of concept for this algorithm has been implemented using the QGIS Python application programming interface (API) with PostGIS as a data store. In the data-preparation stage, a regular pedestrian routing graph is generated and loaded into PostGIS. Open spaces are extracted from OSM by first importing the OSM data into PostGIS using ogr2ogr and then selecting the open-space pedestrian area features from the polygon table using an appropriate SQL query.

Figure 8 shows example routes through Josefsplatz and Stephansplatz which were computed on the pedestrian network that was generated using our base algorithm. These examples show routes that start on the normal pedestrian routing network and then continue on to cross the open space using the most direct route possible. As illustrated by Figure 8 (bottom), this approach is also suitable for open spaces which contain obstacles such as buildings, which are modelled as holes within the polygon.

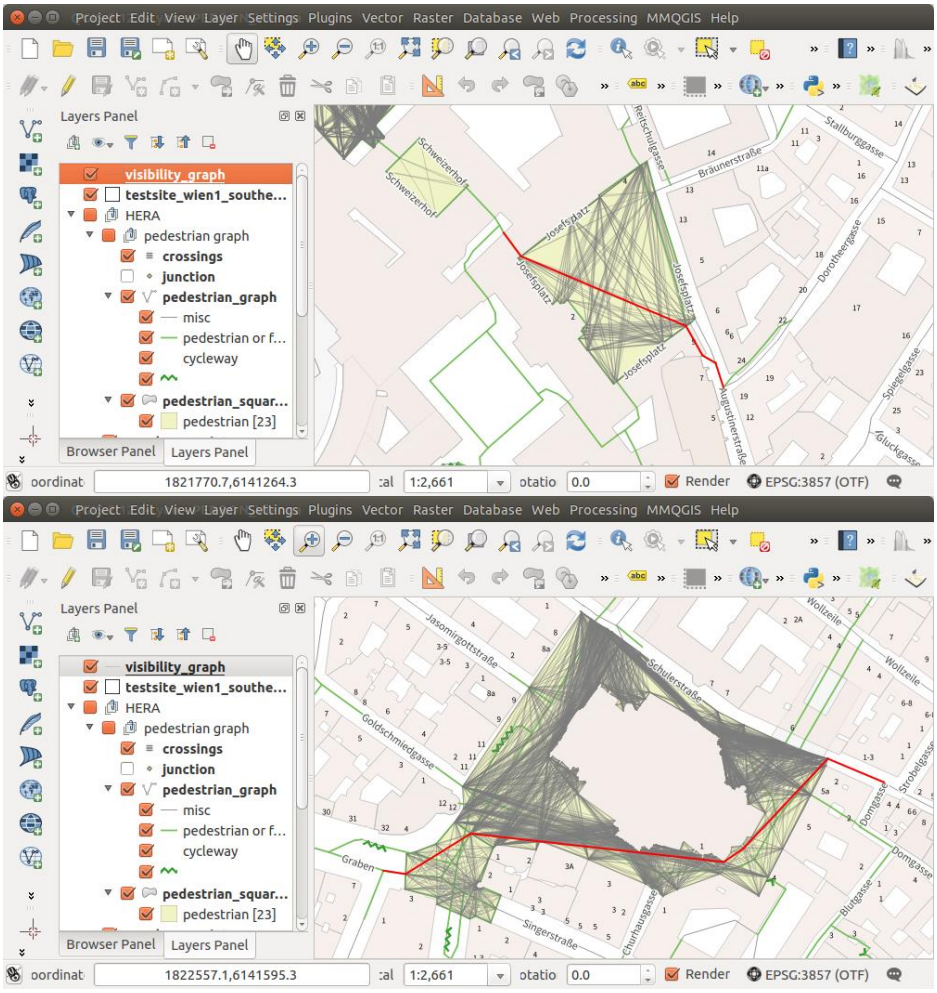

Figure 8: Screenshots of the routing results (red lines) through Josefsplatz (top) and Stephansplatz (bottom) generated using the proof of concept implementation in QGIS

It is worth noting that a disadvantage of this implementation is that OSM node IDs for the open-space polygons and pedestrian network edges are lost during the data-preparation stage. Therefore, it is necessary to resort to geometry-matching of nodes in order to connect the routing and visibility graphs. To avoid the need for this matching procedure and to allow the use of the original OSM node IDs, the visibility graph generation should in the future be integrated directly into OSM pre-processing tools such as osm2po or osm2pgrouting.

Another potential improvement to the algorithm presented is to reduce the number of inserted visibility graph edges by removing irrelevant edges. Figure 9 outlines this extended algorithm. Visibility graph edges are considered relevant if they belong to the shortest path between two entrance/exit points of the open space – that is, where the open space meets the existing pedestrian routing network.

```
1 foreach open space polygon P do
2 |   execute lines 2-8 of the base algorithm;
3 |   identify all nodes Nc which connect to the existing routing graph;
4 |   compute shortest paths between all Nc;
5 |   remove edges which are not on the shortest paths;
6 |   insert all remaining edges into the routing graph;
7 end
```

Figure 9: Extended algorithm for adding only relevant visibility graph edges to the routing graph

This approach can reduce the number of additional edges in the graph considerably. For example, on Josefsplatz the number of edges is reduced from 282 to 19, as illustrated by Figure 10. This extension provides an important step towards making the algorithm scal-able to bigger areas, since the number of graph edges affects both the speed of routing requests, as well as the memory requirements.

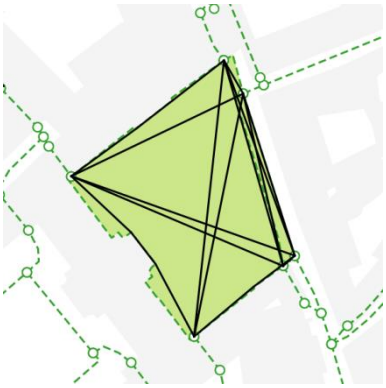

Figure 10: Visibility graph reduced to the relevant edges located on the shortest paths between entrances and exits of the open-space polygon

The resulting pedestrian routing graph enables us to compute more realistic routes and better navigation instructions. On Josefsplatz, for example, current Graphhopper walking directions for the route depicted in Figure 1 (right) instruct the user to first "Turn left onto Josefsplatz", then walk along the outline of the square (there are no explicit directions for this), followed by another "Turn left" after 140 meters to exit the square. In contrast, instructions based on our routing graph state "Bear left and cross Josefsplatz", followed by "Bear right (and use the building passage through Hofburg)" after 91 metres. These instructions represent pedestrian behaviour much more realistically, since turning instructions can be computed with respect to the direct path between entrance and exit of the square rather than the outline.

5 Conclusion

Pedestrians are still second-class citizens in current routing applications, even though data sources such as OSM contain extensive information relevant to pedestrian wayfinding. This paper addresses the shortcomings of current typical integrations of open spaces, such as squares and plazas, into pedestrian routing graphs.

We presented an algorithm for the effcient integration of open spaces into pedestrian routing graphs using visibility graphs – a concept commonly used for wayfinding in digital worlds. This approach enables realistic crossing behaviour following direct routes, without unnecessary detours, and respects potential obstacles such as buildings within the open space. We demonstrated the algorithm using a proof of concept implementation for QGIS to compute routes through open spaces in the city of Vienna, Austria. For an effcient, scalable implementation, the algorithm should be integrated directly into OSM graph-generation tools, such as osm2pgrouting or osm2po.

A better integration of open spaces into routing graphs does not just improve routing results. It should also enable the generation of more appropriate navigation instructions to describe routes that cross open spaces, which are needed to improve current pedestrian navigation systems.

Acknowledgements

This work was supported by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT) within the programme "Mobilität der Zukunft" under Grant 844434 (project PERRON).

References

Aichholzer, O., Aurenhammer, F., Alberts, D., and Gärtner, B. (1996). A novel type of skeleton for polygons. Springer.

Bauer, C., Almer, A., Ladstätter, S., and Luley, P. M. (2014). Optimierte Wegefindung für Fußgänger basierend auf vorhandenen OpenStreetMap-Daten. In Angewandte Geoinformatik 2014, pages 408-413, Salzburg, Austria. Herbert Wichmann Verlag.

Cacciola, F. (2015). 2D Straight Skeleton and Polygon Offsetting. In CGAL User and Reference Manual. CGAL Editorial Board, 4.7 edition.

Chin, F., Snoeyink, J., and Wang, C. A. (1995). Finding the medial axis of a simple polygon in linear time. In Algorithms and Computations, pages 382-391. Springer.

Dräger, M. and Koller, A. (2012). Generation of Landmark-based Navigation Instructions from Open-source Data. In Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, EACL '12, pages 757-766, Stroudsburg, PA, USA. Association for Computational Linguistics.

Elias, B. (2007). Pedestrian Navigation – Creating a tailored geodatabase for routing. In WPNC, pages 41-47. IEEE.

Eppstein, D. and Erickson, J. (1999). Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. Discrete & Computational Geometry, 22(4):569-592.

Graser, A. and Olaya, V. (2015). Processing: A Python Framework for the Seamless Integration of Geoprocessing Tools in QGIS. ISPRS International Journal of Geo-Information, 4(4):2219.

Haunert, J.-H. and Sester, M. (2008). Area collapse and road centerlines based on straight skeletons. GeoInformatica, 12(2):169-191.

Liu, L. and Zlatanova, S. (2015). An Approach for Indoor Path Computation among Obstacles that Considers User Dimension. ISPRS International Journal of Geo-Information, 4(4):2821.

Lozano-Pérez, T. and Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. Communications of the ACM, 22(10):560-570.

Luxen, D. and Vetter, C. (2011). Real-time routing with OpenStreetMap data. In Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '11, pages 513-516, New York, NY, USA. ACM.

Matyus, T., Seer, S., and Schrom-Feiertag, H. (2015). Simulation-based Forecasts of Crowd Flows at Major Events using Real-time Measurements. In Proceedings of the 11th Conference on Traffic and Granular Flow (TGF2015).

Millonig, A. and Schechtner, K. (2007). Developing landmark-based pedestrian-navigation systems. Intelligent Transportation Systems, IEEE Transactions on, 8(1):43--49.

Möller, C. (n.d.). osm2po. http://osm2po.de/; accessed 2016-01-02.

Mononen, M. (n.d.). recastnavigation – Navigation-mesh Toolset for Games. https://github.com/recastnavigation/recastnavigation; accessed 2016-01-03.

Retscher, G. (2004). Pedestrian navigation systems and location-based services. In 3G Mobile Communication Technologies, 2004. 3G 2004. Fifth IEE International Conference on, pages 359-363. IET.

Roberts, S. A., Hall, G. B., and Boots, B. (2005). Street Centreline Generation with an Approximated Area Voronoi Diagram. In Developments in Spatial Data Handling, pages 435-446. Springer.

Smith, M. J. d., Goodchild, M. F., and Longley, P. A. (2009). Geospatial Analysis - A Comprehensive Guide to Principles, Techniques and Software Tools (Third Edition). Matador (an imprint of Troubador Publishing Ltd) on behalf of The Winchelsea Press.

Snook, G. (2000). Simplified 3d Movement and Pathfinding using Navigation Meshes. In Game Programming Gems, pages 288-304. Charles River Media, Newton, MA.

Van Toll, W., Cook IV, A. F., and Geraerts, R. (2011). Navigation meshes for realistic multi-layered environments. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pages 3526-3532. IEEE.

Wiki, O. (2011). Osm2pgrouting OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Osm2pgrouting&oldid=667728; accessed 2016-01-01.

Wiki, O. (2015a). Routing OpenStreetMap Wiki. http://wiki.openstreetmap.org/w/index.php?title=Routing&oldid=1256183; accessed 2016-01-01.

Wiki, O. (2015b). Tag:highway=pedestrian OpenStreetMap Wiki,. http://wiki.openstreetmap.org/w/index.php?title=Tag:highway%3Dpedestrian&oldid=1242393; accessed 2016-01-01.