TOPOLOGY OF URBAN ENVIRONMENTS

Graph Construction from Multi-Building Floor Plan Data

EMILY WHITING, JONATHAN BATTAT and SETH TELLER
Massachusetts Institute of Technology, USA

Abstract: This paper introduces a practical approach to constructing a hybrid 3D metrical–topological model of a university campus or other extended urban region from labeled 2D floor plan geometry. An exhaustive classification of adjacency types is provided for a typical infrastructure, including roads, walkways, green-space, and detailed indoor spaces. We extend traditional lineal techniques to 2D open spaces, incorporating changes in elevation. We demonstrate our technique on a dataset of approximately 160 buildings, 800 floors, and 44,000 spaces spanning indoor and outdoor areas. Finally, we describe MITquest, a web application that generates efficient walking routes.

1. Introduction

We describe a representation and navigation system to support the walking traveler. While many online mapping tools offer powerful route searching functions, they are targeted at automobile travel and model only locally one-dimensional street networks. Current applications are unable to represent continuous indoor and outdoor spaces fundamental to a complex of buildings. Building interiors present the additional challenges of complex connectivity, changes in elevation, and movement through large open spaces.

Metrical-topological models describe the shape and connectivity of space, and are useful underlying tools for wayfinding. We introduce a new approach for constructing metrical-topological models of outdoor and indoor environments from labeled floor plan geometry. We provide an exhaustive classification of adjacency among rooms, corridors, and outdoor spaces, and describe an out-of-core algorithm for construction of a corresponding graph. We demonstrate our method on data from a university campus comprising over 160 buildings, 800 floors, 37,000 indoor and 7,000 outdoor spaces.

"walking route from building W1 (Ashdown House) to room 1-190"

Figure 1: The user made the query shown at left; our method produced the result on the right. Our contribution is to compute efficient routes that seamlessly combine indoor and outdoor paths, allow different levels of granularity for source and destination, and cross 2D spaces not limited to the linear travel of automobiles.

1.1 RELATED WORK

Substantial work has been done on extracting graph representations of street networks using digitized road maps (e.g. Haunert and Sester 2004; Gold 1997). However, these techniques are limited to 1D paths with a single direction of motion, constant width between opposing boundaries, and point intersections. The MIT Stata Walking Guide (Look et. al. 2005) used a manually constructed graph to support route-finding and written walking directions between different locations in MIT's Stata Center. Lee (2004) addressed the problem of automatic graph construction from floor plans using a model derived chiefly from hallway elements which share similar properties to 1D street networks.

We build on work by Funkhouser et al. (1996), Nichols (2004) and Kulikov (2004) on automatic interpretation of floor plan data in a campus environment. Their system analyzes floor plans to derive a hybrid metrical-topological environment representation. Our paper addresses the additional issue of vertical connections (stairs, ramps and elevators), and generates a more complete topological representation of an entire campus. Our contribution is to extend existing methods to 2D spaces such as lobbies and open lounges, and to vertical movement via stairs, elevators and ramps. Our methods involve automated construction of an underlying graph data structure, the topology of which is generated robustly from arbitrarily shaped spaces and varied connectivity types present in raw floor plan data. Further, in contrast to prior work that applied to networks within enclosed buildings, our system extends the graph structure to multiple buildings and adjacent outdoor terrain.

Summarizing, this paper makes the following contributions:

- Methods for inferring and storing a metrical-topological data model representing an extended, indoor/outdoor multi-building environment.
- A framework for constructing graphs of environments too large for processing in one pass.
- A prototype application for generating efficient fine-grained routes for a walking traveler.

1.2 DATA CORPUS

The input to our system is a set of floor plans in DXF format. We base our computations on floor plans because they are a generally available source of geometry data and follow strict conventions, such as unique space names, consistent room use codes, and implied adjacency information. Each construction floor plan is segmented into functional layers such as room contours, floor extents and exterior walls, enabling streamlined parsing.

To derive the physical layout of the terrain, we extract additional input from a *basemap* DXF. The basemap specifies position and orientation for building footprints, along with other physical infrastructure such as locations of sidewalks, streets and grass. For the purpose of route generation (Section 4), we segment basemap regions into units of similar area (Kulikov 2004).

We pre-process the basemap and floor plan geometry to remove degeneracies such as self-intersecting polygons and repeated edges.

1.3 SYSTEM OVERVIEW

Our system comprises a central database for storing geospatial elements, and a set of modules designed for specific read/write operations (Figure 2).

Input modules. Input modules populate the data model by extracting information from common data sources. A DXF parser reads geometry data from raw DXF files, and populates the data model with pertinent information such as building footprints, Space contours, and Portal locations. Section 2 reviews the design of the data model for representing geometric and topological features, and Section 3 describes floor plan conventions used by the DXF Parser to recognize Portal locations.

Derivative modules. Derivative modules operate on the data corpus to derive richer geometric and topological information. For example, graph construction is performed by the Portal Linker (Section 3), the Location Server operates on the graph for path finding functions (Section 4), and an additional module generates Space triangulations.

Output modules. Output modules act as intermediaries between the geospatial database and external applications. For example, the visualization

interface is combined with the Location Server to produce MITquest, a web-based service providing walking routes (Section 4).

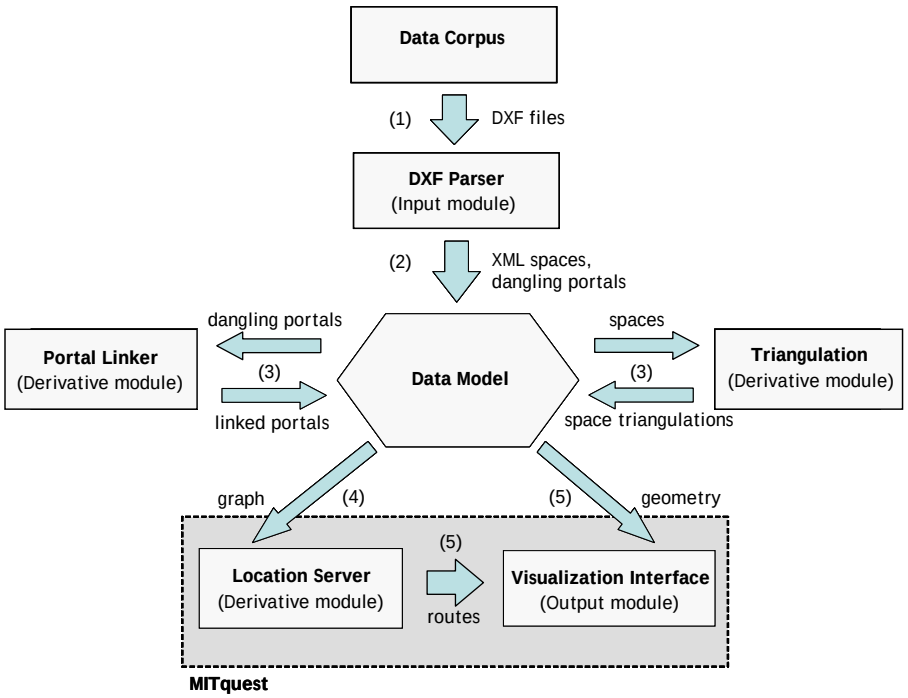


Figure 2: Project overview with ordering of operations. (1) Input DXF files to the DXF Parser, which (2) converts data to our Space and Portal representation. (3) Construct Space triangulations and Space-Portal linkages. (4) Generate routes from the Space-Portal graph and (5) visualize the route on a map.

2. Data Model

Our data model represents containment relationships with a tree data structure, and connectivity relationships with a graph structure. The basic elements of the data model are *Spaces* and *Portals*. Each Space represents a contiguous physical region such as a room, corridor, sidewalk, or patch of outdoor area. Each Portal represents an adjacency, such as a door, elevator, or stairwell connecting two Spaces.

2.1 SPATIAL HIERARCHY

We organize Spaces into a tree hierarchy to represent containment relationships. The root element is the terrain which has individual buildings as children nodes; buildings have floors as children nodes; and so on. Each node has properties such as its name and polygonal contour. We use the

XML file format to represent this hierarchy because of its ability to store information in a tree-based structure, and for ease of transfer over the web.

2.2 SPACE-PORTAL GRAPH

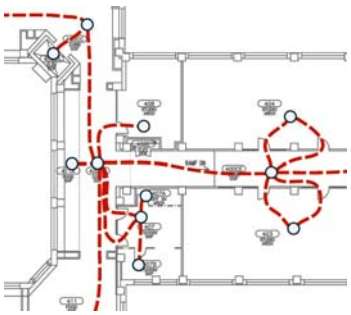


Figure 3: Graph superimposed on floor plan. Spaces are nodes and Portals are edges.

Locations and adjacencies are represented with a graph data structure containing a node for each Space and an edge for each Portal (Figure 3). The graph is directed: each edge $e(v_1, v_2)$ has a source and destination Space v_1 and v_2 respectively. This captures the fact that certain Portals such as security doors allow exit but not unconditional reentry. Each Portal has a weight: the distance between the centroids of the Spaces it connects. From the floor plan input, our goal is to produce a connected graph of all rooms, hallways, sidewalks, streets, and other Spaces within the modeled region.

2.3 DATA ELEMENTS

We now describe the representation of each element in our data model.

Floor. In the tree hierarchy, Floors embody the containing region for a set of Spaces, and are the parent element of all Spaces on that Floor. Each Floor has a set of contour points defining its polygonal boundary.

Space. Each Space has a unique room identifier, a type (e.g. classroom or lobby) and a polygonal boundary derived from the floor plans. From the contour we compute a constrained Delaunay triangulation (Lischinski 1994) and an axis-aligned bounding box (Figure 4 on the right). We analyze each Space to determine which Portals originate from or terminate within that Space, and to compute a height differential for each entry-exit Portal pair in the Space.

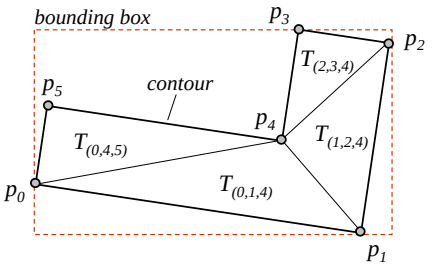


Figure 4: Geometric properties of Spaces are a set of contour points, a triangulation and a bounding box.

Portal. Each Portal has a source Space, a destination Space, and a classification as either Horizontal or Vertical (Figure 5). Vertical Portals

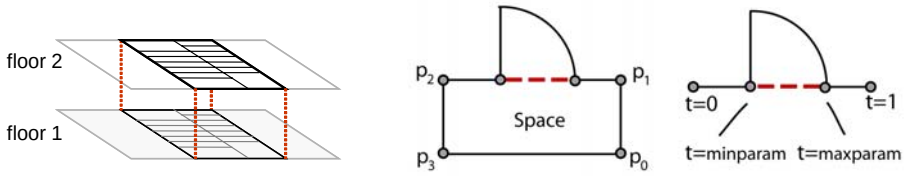


Figure 5: (left) Vertical Portal: two overlapping staircases; (right) Horizontal Portal with extent defined by a Space edge (p_1, p_2) and an interval (t_1, t_2) on this edge.

connect adjacent floors and have type Elevator, Stair or Ramp. They inherit a contour from their source Space (e.g. an elevator Portal has the same shape as the elevator itself).

Horizontal Portals include connections between Spaces on the same floor, connections between buildings, and connections between adjacent basemap Spaces. Each Horizontal Portal is represented as a line segment incident on its parent's contour. (We say that two polygonal contours are "incident" if they share part of at least one boundary segment.).

3. Topology Construction

This section describes the procedure for populating the data model with adjacency information. This involves constructing Portals whenever a pair of Spaces is found to be adjacent and no physical barrier prevents direct traversal from one Space to the other. The main challenge faced is the massive size of the dataset, in our case comprising millions of geometric primitives. Rather than process the entire dataset at once we process one floor at a time. Computations that require data outside the active floor are deferred to a later stage using *dangling Portals* – Portals that have a source Space but a temporarily unspecified destination Space.

In a first pass, our algorithm processes one floor plan at a time. Portals whose source and destination are contained within that floor are fully linked, while any Portal that requires data outside the floor is created as dangling. In a second pass, a Portal Linker matches pairs of dangling Portals. Section 3.1 reviews the various Portal types. Section 3.2 describes the procedure for linking dangling Portals.

3.1 PORTAL TYPES

Portals may be *explicit* or *implicit*, and may be *horizontal* or *vertical*.

Explicit Portals. Explicit Portals, a type of horizontal Portal, represent connections between Spaces that are physically separated by a barrier (e.g. a doorway through a wall). In our source floor plans, each explicit Portal is indicated by a circular arc (Figure 6). We use geometric proximity to determine each Portal's source and destination Space. If the door segment is

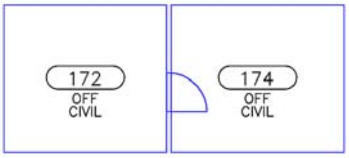


Figure 6: Explicit Portal. Source and destination Spaces are determined by proximity.

incident on the floor contour, this indicates a destination Space outside the building, and the Portal is created as dangling.

Implicit Portals. Implicit Portals, another type of horizontal Portal, represent connections between Spaces with no physical barrier. For example, the photo in Figure 7 shows a large open area outside a cafe. Although there is no wall between the corridor and the café tables, the two regions are defined separately in the floor plans to differentiate between circulation zones and food services. We identify implicit Portals by an incident edge between two Space contours. Similar to explicit Portals, if a Space is coincident with the floor contour, the Portal is left dangling for connection to an abutting building.

Basemap Portals are implicit Portals. Any two basemap Spaces whose boundaries share any portion of a boundary segment are considered adjacent.

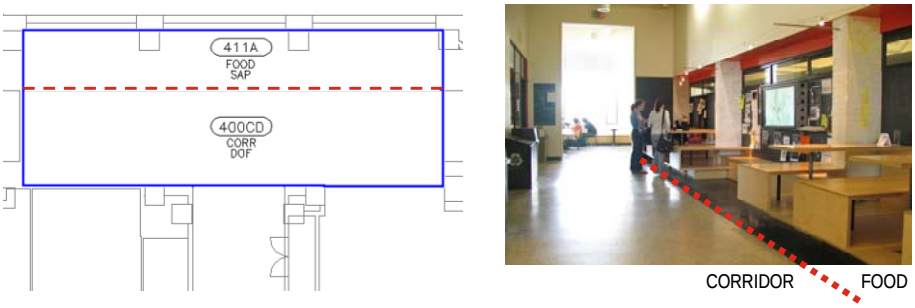


Figure 7: Implicit Portal identified by an incident edge between Space contours (dotted line). Although no physical barrier exists, the Portal places an implicit divide between the corridor and café tables (photo).

Vertical Portals. Vertical Portals connect adjacent floors in multi-story buildings. We identify vertical Portals by searching for stair, elevator and ramp Spaces. Since source and destination Spaces typically lie on separate floor plans, vertical Portals are initially dangling.

3.2 LINKING DANGLING PORTALS

The three types of dangling Portals are building-to-building, building-to-basemap, and vertical. This section discusses the linking procedure for each.

Building Connections. Building Portals model connections between abutting buildings, such as continuous corridors or breezeways. Dangling building Portals are resolved later by finding proximal Space pairs.

Basemap Connections. Connections from buildings to the basemap are handled analogously, with the additional constraint that only explicit Portals are assumed to link an interior Space to the campus terrain. For each dangling Portal leading out of a building, we search for the closest basemap edge. When a match is found, the Portal is linked, and an oppositely-directed Portal is constructed from the basemap Space to the building interior.

Vertical Connections. Two stages are involved in linking vertical Portals. First we construct an ordered list of floors for each building, assuming consecutive floors are vertically adjacent. Second, for each ordered pair of floors, we find stair and elevator Spaces whose axis-aligned bounding boxes overlap in plan view (Figure 8a). Two oppositely-directed Portals are constructed representing upward and downward movement respectively.

Mezzanines present a special case when determining floor adjacencies. A staircase may connect either to a mezzanine or to the full floor above (Figure 8b). To determine the proper connection we cast a ray upward from the bottom-most staircase. If the ray lies outside the mezzanine extents then the mezzanine can be ignored.

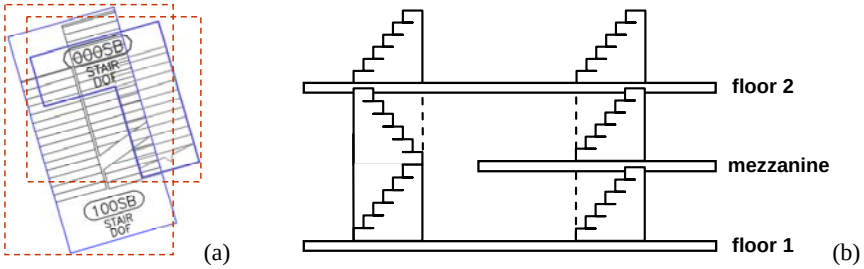


Figure 8: Vertical Portals. (a) Plan view of stairs on adjacent floors. Axis-aligned bounding boxes (red) overlap; (b) Mezzanines in the floor stacking sequence.

TABLE 1: Summary of Portal types and detection/linking methods.

Type	Connection	Test
Implicit	Intra-Floor	Incident edge between two Space contours
	Inter-Building	Incident edge between Space and Floor contour
	Intra-Basemap	Incident edge between two basemap Space contours
Explicit	Intra-Floor	Door arc proximal to two Space contours
	Inter-Building	Door arc proximal to Space and Floor contour
Vertical	Inter-Floor	Axis-aligned bounding box overlap

4. MIT Route Finder

To demonstrate the practical value of our fine-grained metrical-topological model, we developed MITquest, a prototype application that generates efficient walking routes between any two locations on the MIT campus. The route generation problem is divided into three sub-problems: finding a sequence of Spaces that efficiently connects the source and destination (Section 4.1); finding the actual path through each Space (Section 4.2); and combining these results into a route to be displayed to the user (Section 4.3).

4.1 SPACE PATH

The first step determines an efficient sequence of Spaces connecting the source and destination in the Space-Portal graph (Figure 9). With each Portal's edge weight set to the distance between adjacent Space centroids, the Space sequence is determined by applying Dijkstra's shortest-path graph search algorithm (Cormen et al. 2001).

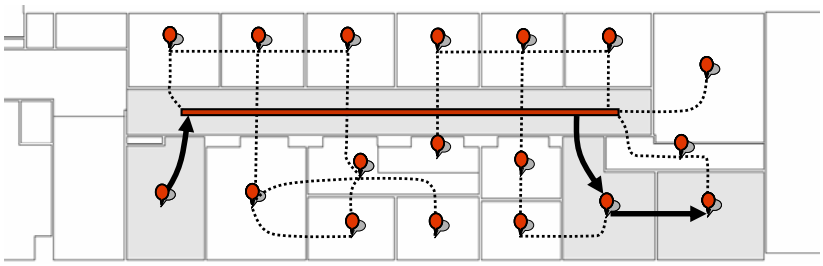
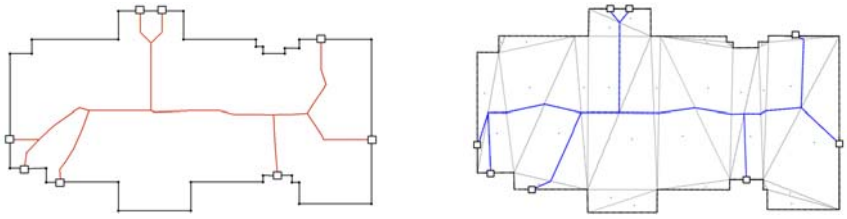


Figure 9: Example Space-Portal graph. Edges (dotted lines) are placed between Spaces (red nodes) connected by Portals. Edges are weighted by centroid-to-centroid distance. Solid arrows represent traversed edges between the start and end Spaces.

4.2 POLYLINE PATH

The second step determines a metrical path *through* each Space in the route. The challenge is to handle non-convex Spaces where a simple straight line between Portals may intersect the boundary.

To extract the lines linking Portals, we use a method by Joan-Arinyo et al. (1996) that approximates the medial axis (Lee 1982; Latombe 1991). In practice, this implementation has lower computational cost and yields nearly equivalent results. We construct a graph inside each non-convex Space by placing straight line segments between midpoints of shared triangle edges (Figure 10). To find the shortest path between any pair of Portals, we then run Dijkstra's algorithm on the resulting graph. The effect is a natural-looking path that follows the shape of the Space. This method is fast because we pre-compute triangulation data when we parse the DXF file.



Medial axis construction Our triangle-based approximation

Figure 10: Internal graph connecting Portals for a typical room contour. We approximate the medial axis by connecting midpoints of shared triangles edges.

4.3 ROUTE GENERATION

Finally, we combine the Space sequence and polyline paths to generate efficient routes. To do so we must determined the polyline path through the entire sequence of Spaces. We first determine the union of intra-Space graphs (Figure 11). The graphs are combined by merging corresponding Portal nodes between each pair of adjacent Spaces. When multiple matches are available, nodes are paired by geometric proximity. Dijkstra's algorithm is run on the merged graph to determine the shortest polyline path from source to destination.

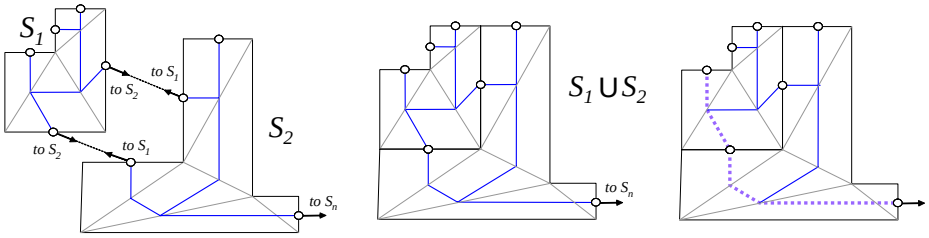


Figure 11: Combination of internal graphs for Spaces S_1 and S_2. The two graphs are connected at corresponding Portal nodes. Dijkstra's algorithm searches the combined graph to find a path through the Space sequence.

4.4 PATH RELAXATION

Once the polyline path has been determined, a final shape simplification process is applied. This stage reduces the number of segments in the path to yield a cleaner looking map, while leaving the overall shape of the route intact. For each Space, we step through the path removing interior points. When a removal causes the path to intersect the Space boundary, we re-insert the point, and repeat the process with an updated starting position. The process is demonstrated in Figure 12. Our approach is similar to that of Agrawala (2001) in shape simplification of road maps, with the difference

that we maintain overall shape by constraining the route within the Space boundaries.

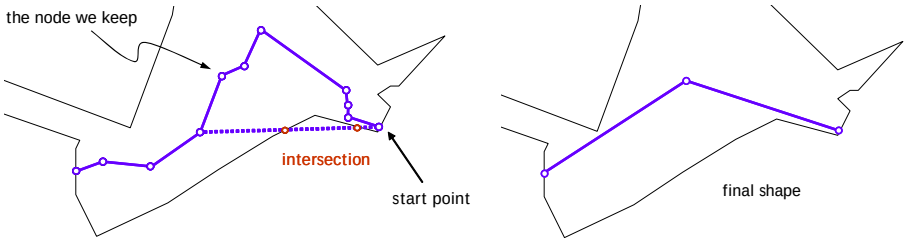


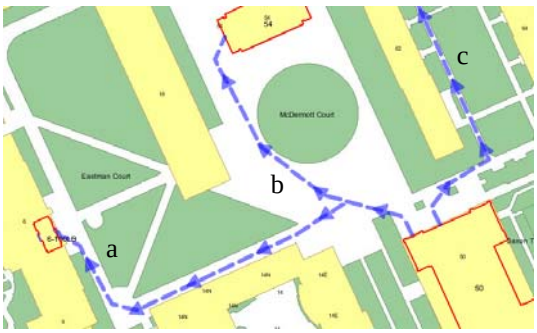
Figure 12: The path relaxation process removes interior points, while ensuring that removals do not cause the path to intersection the Space boundary.

4.5 ROUTE CONSTRAINTS

Route constraints are implemented by selectively restricting undesirable Space and Portal types in the campus graph. For example, “paved” routes prohibit Portals leading to grass or construction on the basemap, but allow streets, sidewalks, and any indoor Space type. “Rolling” routes are additionally constrained to traverse only elevators and ramps when moving vertically within buildings.

5. Results

We have implemented our route visualization tools as an interactive web-based application modeled after the Google Maps API. The interface allows users to specify source and destination points at varying levels of granularity, either by building, room or a combination. We illustrate a few representative scenarios in the following figures.



(a) “building 50 to room 6–100”

(b) “building 50 to building 54”

(c) “building 50 to building 66”

Figure 13: Outdoor routes crossing a large courtyard, resulting from queries (a)–(c). The multi-directional walking routes exploit the 2D nature of the terrain.

Google Maps route to address:
"50 Massachusetts Ave. Cambridge MA"

Our result from query:
"building W84 to building W13"

Figure 14: Increased level of detail of MITquest in comparison to Google Maps. Our outdoor route employs the nearest building entrance rather than a street address.

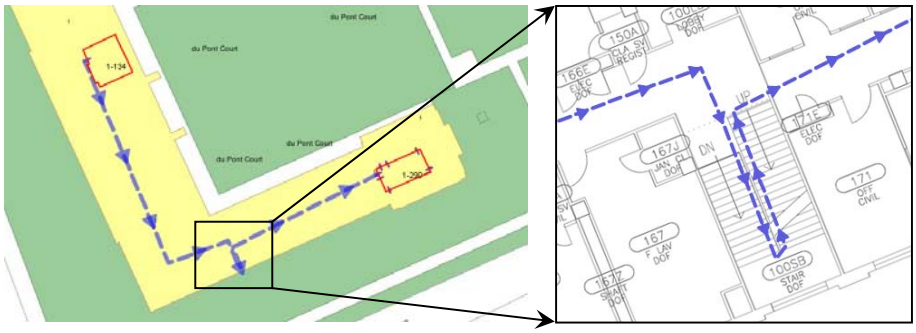


Figure 15: Indoor route with vertical segment: "room 1-134 to room 1-290". The route goes through 1st and 2nd floor corridors, ascending a staircase along the way (detail). Visualization of height differences is an area for future work.

5.1 POSSIBLE EXTENSIONS

Non-Convex Spaces. Shortest routes leaving and re-entering a Space form a cycle in the Space sequence, and are not handled by our graph search algorithm. This scenario arises only with non-convex Spaces. A possible extension is to break up all non-convex Spaces into convex components.

Misaligned Floors. When linking Portals between adjacent buildings, we assume that connections occur at identical floor numbers. This assumption does not always hold in an environment with buildings of varied ages and styles. An alternative is to determine elevations from grade and impose the constraint that connections occur at identical elevations.

Graphical Identifiers. The DXF Parser relies on a labeling scheme specific to MIT. A remaining challenge is to incorporate more universally applicable

graphical identifiers. For example, a staircase is more commonly represented by a sequence of closely spaced line segments than by a "STAIR" label.

IFS and OpenGIS. Our data model and graph construction routines were developed for efficient performance on the MIT campus data corpus. Future developments could improve interoperability with existing IFC (Liebich and Wix 2000) and OpenGIS (OGC 2007) tools.

6. Conclusion

This paper makes three contributions. First, we describe a data model for storing geometric and topological properties of the built environment. Second, we provide an exhaustive classification of adjacency types for an urban infrastructure, and methods for inferring topological relationships from floor plan geometry. Third, we present MITquest, a prototype application that demonstrates how our data model can be used to generate walking routes spanning interior and exterior environments. The MIT campus is used as a test-bed which is representative of many urban datasets and demonstrates that our algorithms are readily usable.

The work presented in this paper could help answer numerous practical questions in architectural design. What are the common navigation paths within an environment? Do they provide adequate circulation capacity? Are landmarks effectively placed in the environment to aid navigation? Does a building provide appropriate facilities for access and navigation by people with disabilities? Directions for future research include automated landmark identification, improved route visualization, incorporation of object-oriented building models using IFC, and integration of our local environment models with global urban features.

References

Agrawala, M and Stolte, C: 2001, Rendering Effective Route Maps: Improving Usability Through Generalization, *in Proc. ACM SIGGRAPH 2001*, pp. 241-249.

Bell, J: 2003, *An API for Location Aware Computing*, Master's Thesis, Massachusetts Institute of Technology.

Bern, M and Eppstein, D: 1992, Mesh generation and optimal triangulation, *in* F Hwang and D Du (eds), *Computing in Euclidean Geometry*, World Scientific, pp. 23-90.

Cormen, T, Leiserson, C, Rivest, R and Stein, C: 2001, *Introduction to Algorithms – 2nd ed.*, The MIT Press and McGraw-Hill.

de Berg, M, van Kreveld, M, Overmars, M and Schwarzkopf, O: 1998, *Computational Geometry: Algorithms and Applications 2nd ed.*, Springer-Verlag, Berlin.

Funkhouser, T, Séquin, C and Teller, S: 1992, Management of Large Amounts of Data in Interactive Building Walkthroughs, *in Proc. 1992 Symposium on Interactive 3D Graphics*, pp. 11-20.

Funkhouser, T, Teller, S, Sequin, C and Khorramabadi, D: 1996, The UC Berkeley System for Interactive Visualization of Large Architectural Models, *Presence: Teleoperators and Virtual Environments* **5**(1): 13-44.

Gold, C: 1997, Simple Topology Generation from Scanned Maps, *in Proc. Auto-Carto 13, ACM/ASPRS*, pp. 337-346.

Google: 2007, *Google Maps API*, http://www.google.com/apis/maps/.

Guibas, L and Stolfi, J: 1985, Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams, *ACM Transactions on Graphics* **4**(2): 75–123.

Haunert, J-H and Sester. M: 2004, Using the Straight Skeleton for Generalisation in a Multiple Representation Environment, *in Proc. ICA Workshop on Generalisation and Multiple Representation*.

Joan-Arinyo, R, Perez-Vidal, L and Gargallo-Monllau, E: 1996, An Adaptive Algorithm to Compute the Medial Axis Transform of 2-D Polygonal Domains, *in* P Brunet and D Roller (eds), *CAD Tools for Products*, Springer-Verlag.

Kulikov, V: 2004, *Generating a Model of the MIT Campus Terrain*, Master's Thesis, Massachusetts Institute of Technology.

Latombe, J: 1991, *Robot Motion Planning*, Kluwer Academic Publishers.

Lee, DT: 1982, Medial Axis Transformation of a Planar Shape, *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-4, pp. 363–369.

Lee, J: 2004, A Spatial Access Oriented Implementation of a Topological Data Model for 3D Urban Entities, *GeoInformatica* **8**(3): 235-262.

Liebich, T and Wix, J (eds): 2000, *IFC Technical Guide – Release 2x*, International Alliance for Interoperability.

Lischinski, D: 1994, Incremental Delaunay Triangulation, *in* P Heckbert (ed), *Graphics Gems IV*, Academic Press, pp. 47-59.

Look, G, Kottahachchi, B, Laddaga, R and Shrobe, H: 2005, A Location Representation for Generating Descriptive Walking Directions, *in Proc. 10th International Conference on Intelligent User Interfaces*, pp. 122–129.

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

MassGIS, Office of Geographic and Environmental Information: 2006, *Datalayers/GIS Database Overview*, http://www.mass.gov/mgis/spc-pts.htm.

Nichols, P: 2004, *Location-Aware Active Signage*, Master's Thesis, Massachusetts Institute of Technology.

Open Geospatial Consortium, Inc.: 2007, *OpenGIS Specifications (Standards)*, http://www.opengeospatial.org/standards/.

Roush, W: 2005, *Killer Maps*, Technology Review **108**(10): 54–60.

Timpf, S and Frank, A: 1997, Using Hierarchical Spatial Data Structures for Hierarchical Spatial Reasoning, *in* S Hirtle and A Frank (eds), *Spatial Information Theory*, Springer, Berlin, pp. 69–83.

Varadhan, G and Manocha, D: 2002, Out-of-Core Rendering of Massive Geometric Environments, *in Proc. IEEE Visualization*, pp. 69–76.