# A Building Database for Simulations Requiring Schemata

**Gabriel Wurzer[1], Jelena Djordjic[2], Wolfgang E. Lorenz[1] and Vahid Poursaeed[3]**

[1]TU Wien
Vienna, Austria
{*firstname.surname*}@tuwien.ac.at

[2]Moser Architects
Vienna, Austria
lela.simanic@gmail.com

[3]Iran Univ. of Science and Techn.
Tehran, Iran
poursaeed_v@arch.iust.ac.ir

**ABSTRACT**

Obtaining spatial representations of existing buildings for use in simulation is challenging: To begin with, getting permission to access submitted construction plans can take a long time. Then these might only be available in analog form, making it necessary to scan and vectorize them at the regulations office. The resulting representation might still not be adequate for simulation, requiring further extraction of relevant features and enrichment by additional information in order to fit the simulation domain. In our work, we have specifically targeted simulation types that work with schemata (e.g. occupancy, work and egress simulations). Our contribution lies in restructuring the aforementioned workflow so as to (1.) minimize time and effort spent on digitizing and to (2.) automatically derive schemata – sets of boundary polygons which (3.) can be further enriched by attributes. These steps are embedded into a web-based building database which allows uploads and queries per web interface as well as web services. The query interface furthermore includes (4.) the ability to download the schemata both in vector as well as raster form so that they can be used for both discrete and continuous-space approaches. Apart from acting as data provider, the database furthermore (5.) allows for spatial predicate functions which may be used for analysis of a space program.

**Author Keywords**

Simulation; database; spatial representation; web services.

**ACM Classification Keywords**

H.2.8 DATABASE APPLICATIONS / Spatial databases and GIS

## 1 INTRODUCTION

Schemata form the basis for a wide range of simulations that need spaces to be represented simply as boundary polygons (continuous-space simulations) or collections of cells (discrete-space simulations). Such a representation could be derived from data-rich models (i.e. Building Information Models [BIMs] for architecture; Geographical Information Systems [GIS] and Web Mapping Services [WMS] for the urban case), however, these might not be available when dealing with existing buildings for which no digital plans exist. Improving the workflow for getting a digital schema from an analog plan was thus our primary goal – technically achieved by the implementation of a web-based building database that can process images into schemata and allows for elaborate queries via web interface and web services. In more detail, we have

- looked at the current workflow for obtaining schemata from analog plans, carefully restructuring and automating individual steps so as to minimize time and effort spent (see section 'Restructuring and Automating the Digitization Process'),
- incorporated the resulting schemata into a web-based building database which lets us enrich the data further by uploading attribute tables and other media (see section 'Database Representation'),
- devised a query and data retrieval interface in the form of a web interface and RESTful webservices, providing a way for users and applications to interact with the data (see section 'Query and Data Retrieval Interface').

In order to argue for the applicability of our method, we have tested both import and retrieval using a showcase setup (see section 'Showcase'). We further provide a discussion that outlines limitations of our approach (see section 'Discussion') before concluding.

## 2 RELATED WORK

Many authors have already tried to infer spatial representations from image data: Koutamanis' work [2] is occupied with semantic recognition of building elements from sketches while others employ a rule-based vectorization strategy in floor plan images [3, 4]. In contrast to these approaches, we use a polygon-based vectorization algorithm [5] which has no contextual knowledge specific to architecture. Since it works on monochrome bitmaps, the user needs to separate different features by at least one pixel, which is reasonable if we assume that the interior polygon is shown and walls are simply left out.

Apart from technique used for recognition, the actual representation used for schemata might differ: For example, Tabak [6] uses a circulation graph to which he attaches spaces as single nodes. Our representation uses spaces rather than a circulation network, however we can get the schema as raster and transform the midpoint of each cell into a network node. Once connected to its neighboring nodes, the network can then be used for navigation between different spaces. Dijkstra and Timmermans [1] have used a

similar approach (network of decision points), but these nodes served as intermediate goals for lattice-walking agents.

## 3 RESTRUCTURING AND AUTOMATING THE DIGITIZATION PROCESS

Depending on regulation, one may need a permission to access the submitted construction plans of a building. The current workflow shown left in Figure 1 thus begins with a supposedly lengthy task "obtain permission". Since we assume that we work with non-digital plans, the next step lies in a visit to the regulations office in which the plans have to be scanned or photographed and optionally stitched in a post-step. Next comes the vectorization part, which tries to identify and restore geometry. However, this process is lossy and might not be adapted to hand-drawn plans which are typical for older structures. It can thus safely be assumed that some geometry will be omitted or recognized only in part. We expect that a bit of manual work in restoring or completing features is part of the process, which also involves deleting parts of the geometry that are not relevant (task "extract relevant features"). With that, the vectorization is complete. One may still want to add attributes to certain features, e.g. room names (if not correctly recognized), functions and so on.

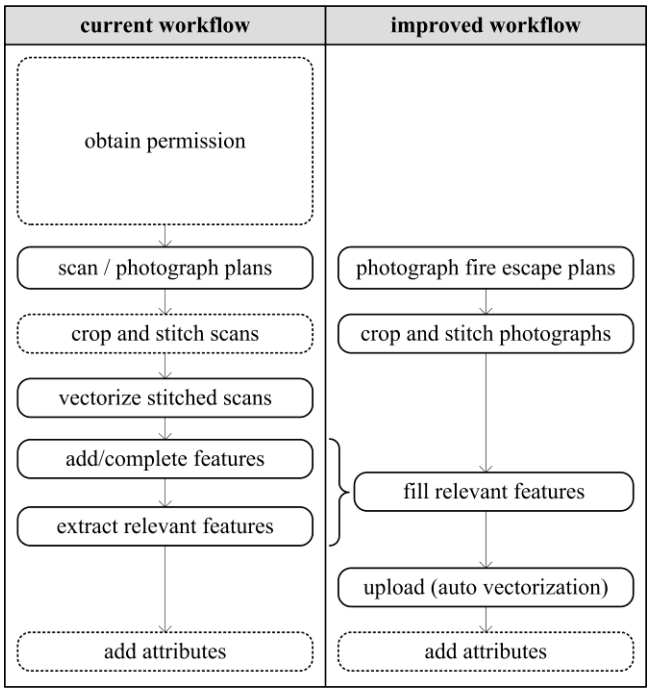



**Figure 1.** (left) current workflow (right) improved workflow.

When improving the current workflow, we saw our biggest challenge in the initial permission step. There are two ways in how to circumvent this, (a.) by photographing the fire escape plans without asking for permission or (b.) to scan floor plans shown in printed literature. However, the last option seems only feasible for historical architecture, since magazines and books publishing contemporary buildings tend to depict only certain areas or levels. Thus, we stick with the first case - to photograph fire escape plans (see right in Figure 1) since we aim at recent architecture. Because our photographs will likely contain areas and not a whole floor, we must stitch them together in a post step.

The biggest difference between the current and improved workflow comes in the next step: Rather than vectorizing and completing features in a post-step, we fill each extracted space (a simple flood-fill operation in a drawing package) with a known color (or a set of colors, if multiple space are to be marked as belonging to one common area). Details that are not relevant (e.g. doors, equipment) may need to be cleared beforehand since the schema is only concerned with boundaries. A further way in which superfluous details can be eliminated is to clear all pixels that are not in the range of colors used to signify spaces.

The color-coded images are now uploaded to the database which performs an automatic vectorization. In that process, it assigns an id and color to each resulting feature. The vectorization we use guarantees that all features are found if there is at least one pixel between separate elements. The result is registered as vector layer of a certain name (e.g. "schema") at a certain level (e.g. 1) of a certain building. We use naming conventions on the image file in order to extract this information.

In a post-step, one may assign attributes to individual spaces or areas by specifying a mapping of the form (color, *...attributes...*) or (feature id, *...attributes...*) interactively or by upload of a spreadsheet.

## 4 DATABASE REPRESENTATION

The database acts as a repository that stores uploaded schemata per building and level. It furthermore offers the possibility to upload other media (e.g. images, spreadsheets, documentation) which are registered on a per-building basis. An graphical outline is given in Figure 2.

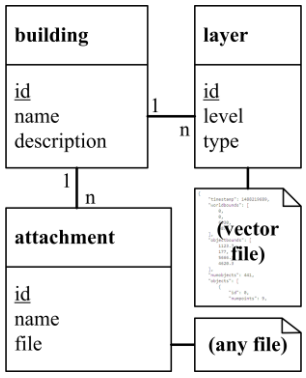
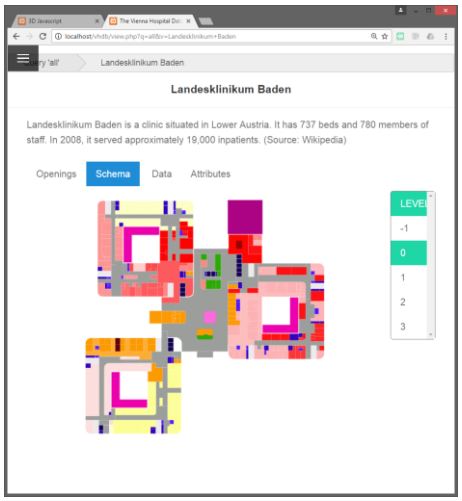
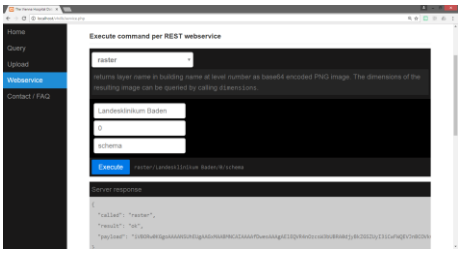



**Figure 2.** Database representation and linked files.

Technically we use the sqlite database to store all metadata concerning buildings, layers and attachments. The actual vector data is represented as files containing Javascript Object Notation (JSON), attachments are written out as files in their original formats.



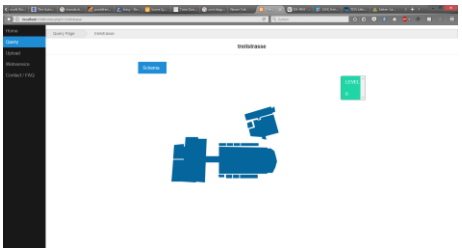

(a)




(b)

**Figure 3.** (a) Web-based query interface with 2D content viewer. (b) Web service call with results shown online.


(a)



(b)

**Figure 4.** (a) Touched-up fire escape plans and (b) 2D viewer displaying the ground floor of our institute.

## 5 QUERY AND DATA RETRIEVAL INTERFACE

Until now the changed workflow proposed by the authors does little more than to vectorize image data with some added constraints (1 pixel separation between spaces) such that these become distinguishable. The true power of the building database lies in leveraging the information for producing extracts in both raster and vector form, offer spatial predicates based on the imported data and to be interfaced in multiple forms, either

- through a web interface which allows end-users to query and view content in 2D (see Figure 3a) or
- a web service which enables applications to interact with the database (see Figure 3b for a web-based service call and response).

The interaction lies in mapping/transforming geometries to the client coordinate system (by scaling the whole geometry or extracting a portion, given by a window) in both vector and raster forms (by rendering geometries to a raster map), and to map colors and indices to attributes (through attachments in spreadsheet form listing [color, *...attributes...*] or [feature id, *...attributes...*]). Another area of queries lies in spatial predicates which are concerned with topological queries (e.g. neighboring spaces), realized internally through the use of an adapted form of the Dimensionally Extended nine-Intersection Model (DE-9IM) which is common in GIS systems.

The query interface does not carry a state, which is assumed to be supplied by the client. In more technical detail, our web service expects to be called in RESTful form, through

*{function name}/parameter1/parameter2/.../parameterN*

which is received by a PHP stack and mapped to a range of service routines. The results of each call are given in JSON, giving a wide range of web-based clients the opportunity to interact with our database.

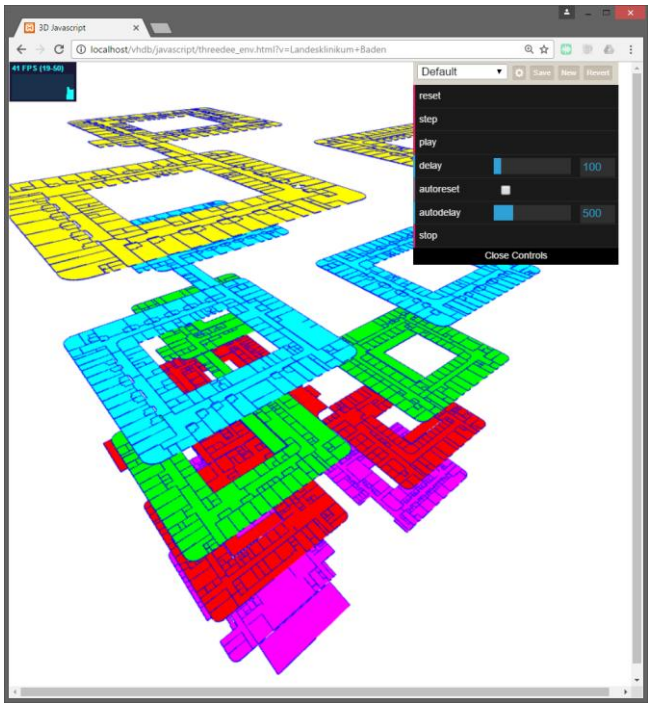
## 6 SHOWCASE

In order to argue for the applicability of our method, we have gone through all of the proposed steps and got a benefit in terms of time spent digitizing and effort involved in getting the gathered data ready for simulation. Our test was split into two phases, (1.) gathering input using fire escape plans, which were digitized automatically and (2.) writing client programs that would access data stored in the database.

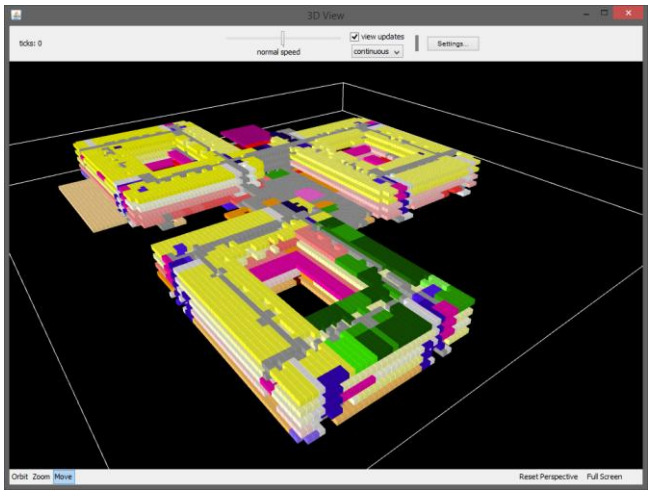
For the first task, we scanned the fire escape plan of our own institute, which took 28 minutes for outlining a single floor. The touched-up data is shown in Figure 4a and 4b and is discussed in due course (see 'Discussion').

For the second task, we imported a building from existing floor plans. A data query in vector and raster form was instantaneous, in the order of milliseconds. It enabled programs (see Javascript and Netlogo 3D in Figures 5a and 5b) to utilize a rich scale of options to choose from when

mapping into an own coordinate space and, at the same time, use the set of spatial predicates available via the data interface of the database.




(a)




(b)

**Figure 5.** (a) Javascript [three.js] and (b) Netlogo 3D viewer accessing the database via web servives.

## 7 DISCUSSION

Apart from all technical assumptions, it is clear that we presume that we can enter all levels of a building for photographing fire exit plans. This optimistic strategy is driven by the fact that a simulation study is typically conducted in a project context where the client actively supports the retrieval of plans.

The second discussion point lies in the contribution of the method. It is clear that some of the mentioned concepts could be achieved by redrawing features in a vector program, however, the proposed service architecture offers additional functions that make the proposed system valuable as content provider, freeing simulation of having to implement certain functionalities on their own. With that, we are aiming at a service architecture enabling client programs to scrap some of their code when connecting to databases, which is a good outlook for the development of the field.

## 8 CONCLUSION

We have presented an improved workflow for obtaining digital schemata from analog images. The proposed process is integrated into a building database that offers additional functionalities for querying, transforming and annotating that data and is accessible via a web interface and web services. The applicability of our method was showcased by conducting all steps from data acquisition to usage of the data in a simulation.

## REFERENCES


1. Dijkstra, J., Timmermans, H. Towards a multi-agent model for visualizing simulated user behavior to support the assessment of design performance, *Automation in Construction 11*, 2 (2002), 135–145
2. Koutamanis, A. Recognition of Building Elements in Free-Hand Sketches. *Proc. 26th eCAADe*, 2008, 419-426.
3. Lu, T., Yang, H., Yang, R., Cai, S. Automatic analysis and integration of architectural drawings, *Int. Journal of Doc Analysis and Recognition 9*, 1 (2007), 31-47.
4. Macé, S., Valveny, E., Locteau, H., Tabbone, S. A System to Detect Rooms in Architectural Floor Plan Images, *9th IAPR Int. Workshop on Document Analysis Systems*, 2010, 167-174.
5. Selinger, P. Potrace: a polygon-based tracing algorithm (2003). http://potrace.sourceforge.net/potrace.pdf. As of 1st March 2017.
6. Tabak, V. *User Simulation of Space Utilisation*, PhD Thesis, Technical University Eindhoven, 2008.