NAVIGATION IN INDOOR VOXEL MODELS

Ben Gorte [a], Sisi Zlatanova [a], Fodil Fadli [b]

[a] GRID-UNSW, Fac. of Built Environment, Sydney, Australia, {b.gorte, s.zlatanova}@unsw.edu.au
[b] Architecture and Urban Planning (DAUP), College of Engineering, Qatar University, f.fadli@qu.edu.qa

Commission IV / 5

KEY WORDS: 3D building models, indoor navigation, 3d reconstruction, voxel models.

ABSTRACT:

The paper proposes to use voxel models of building interiors to perform indoor navigation. The algorithms can be purely geometrical, not relying on semantic information about different building elements, such as floors, walls, stairways etc. Therefore, it is possible to use voxel models from different data sources, in addition to vector-to-raster conversions. The paper demonstrates this on the basis of tree different input types: hand measurements, point clouds and images of floorplans. On the basis of these models, the paper shows how to determine the navigable space in a voxel model for a pedestrian actor, and how to compute paths from arbitrary sources to specified destinations.

1. INTRODUCTION

Indoor navigation is attracting the attention of many researchers in the last decade (Khoshelham and Zlatanova 2016). Indoor environments are usually much more complex and difficult for orientation compare to outdoor. Creating indoor models poses many challenges for indoor navigation (Zlatanova et al 2013). In most of the cases, there are no clear paths and directions for walking, as humans can move through the entire empty space. Commonly a predefined network model is created to be able to perform shortest path computations. However, such networks are too abstract and do not consider the entire space, where people can be located or can move through.

Another complication is brought by the dimension. Indoor environments are three-dimensional and, in many cases, the commonly used 2D dimensional floor plans fail to represent accurately the available space for navigation. Shapes and sizes of stairs are not modelled and are not included in the navigation models.

Various approaches for indoor navigation have been presented in the literature exploring 3D vector-based representations, as well as voxel representations. Several frameworks have been investigated that provide mechanisms for space subdivision of 3D vector models (Diakité and Zlatanova, 2018, Sithole and Zlatanova 2016), but the computational complexity for some types of representation is too high.

Voxel-based methods gain an increased interest due to their flexibility, simplicity and efficiency. In several papers we have investigated the suitability of voxel models for navigation (Lim et al 2018, Xiong et al 2016)

In this paper we present a voxel-based approach that allows to construct quickly a 3D model from point clouds, identify the navigable spaces and compute a path for navigation from any possible accessible point.

The idea to navigate actors in two- or three-dimensional gridded spaces was already being studied in robotics literature during the nineties of the last century (Huang and Ajuha, 1992). Some instances of this research were based on distance transforms, which we use as well (Bandi and Thalmann, 1998). At that time, efficiency, both in time and in space, was much more of an issue than it is nowadays, which led the research into various complications (Kitamura et al 1995, Vörös 2001).

Especially in GIS-environments the preferences went into the direction of vector-based modelling. We want to demonstrate in this paper that grid-based solutions nowadays are even more than before a feasible alternative, offering great flexibility with respect to the data sources (manual entry, point clouds or building floor plans) from which the 3D models originate. Furthermore, we used the results of (Koopman, 2016) to parameterize the sizes (heights and widths) of the actors – in the examples provided the values are 1.8m and 0.4 m, respectively. Finally, we have a novel methodology to have pedestrian actors make use of stairways in the building model, which take part in the routing process on the basis of their geometry only – in fact, our entire method is purely geometrical and no semantic information is required.

2. VOXELS MODELS

A voxel model of, for example, a building exists in a 3-dimensional, regular, rectangular array of cells (the voxels), which spans a ‘block’ of space that fully contains the building. All voxels have the same size, and each voxel can be addressed by an index, specifying its position in the block of space under consideration. Furthermore, each voxel has a value. The block of space is entirely filled with voxels, and the values are used to distinguish ‘air’ from ‘building’ voxels – in the simplest case.

This is the conceptual view at voxels. Physically, we can distinguish between dense and sparse representations. In the dense case *only* the voxel values of *all* the voxels are represented in a known order. Then the position in the block (the index) is uniquely determined by the position in the data sequence and does not have to be explicitly stored. In the sparse case, there is a default value (usually ‘air’). Only the other (non-air) voxels are stored, together with their indices. Retrieving a voxel at a certain

position involves looking it up by the index in the data structure or declaring it 'air' if it is not found.

The idea of voxel analysis is that a voxel dataset can serve as input to an operation, which produces a modified (or an entirely new) dataset as its output. In this study we present operations to implement indoor navigation in voxel datasets that model building interiors. Three cases are shown: First we demonstrate the detailed analysis steps with a small apartment model that was made by hand. Then we repeat the analysis in another model of the same apartment, generated from a (simulated) laser point cloud. Lastly, we show a (larger) office building that was modelled based on floor plans.

3. METHODOLOGY

We have a manually created voxel model of an apartment. Empty (transparent) space has voxel value 0, 'hard' objects, such as floors, walls, ceilings and furniture have value 1 (but there is hardly any furniture). The model has been entirely created by hand, by subdividing the hard elements into 3d rectangular boxes and denoting their places and sizes in a script for a little program that inserts the blocks into a 3dimensional matrix, combining them using logical operators OR and SUBRACT (for the windows).

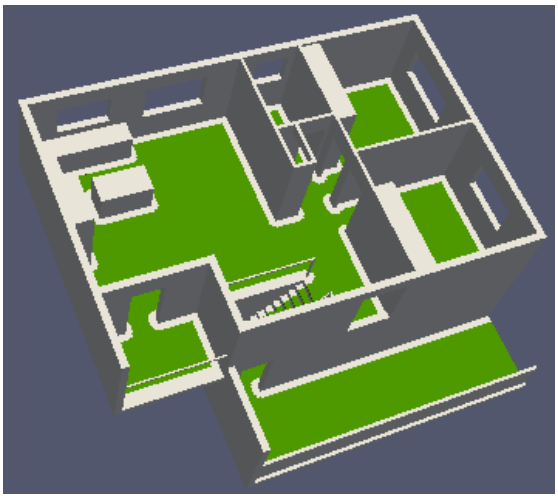

Fig. 1: Apartment voxel model after identifying navigable floor space.

The apartment has two floors. The entrance is on the ground floor, and stairs lead to the second floor, which has a combined living/kitchen, two bedrooms, a bathroom and a balcony. The voxel size in this example is 5cm and the 3-dimensional array has an extent of 208 x 240 x 98 voxels (see Fig. 1, while ignoring the green part for the time being).

In a voxel model we can do indoor navigation, consisting of several steps. The first step computes the navigable floor space: a subset of the 'hard' voxels that satisfies the following conditions.

- there is at least 1.8m of air above a navigable voxel
- there is 20 cm of air around the vertical line above a navigable voxel
- all navigable voxels are connected in 2D
- height jumps are max. 25 cm (allowing to use the stairs)

The first two of those conditions can be checked by 3D spatial filtering. This operation is an extension of the well-known filters for convolution and morphology for analysis of 2D images and other raster data sets – the 3D extension can meanwhile be considered *known* as well. In the current case we would have a 3D kernel (or structuring element) depicting a 1.55 m (31 voxel) high cylinder with a 45cm (9 voxels) diameter, standing on a stick of 25cm (5 voxels) high, with the origin of the kernel at the bottom end of the stick. We perform a normal convolution (i.e. count the 1-voxels that are covered by the kernel) and are only interested in 1-voxels of the model where the convolution result is equal to 1 – meaning that all the other covered voxels are 0. The trick with the stick is to allow for the height jumps of max 25 cm (the 4th condition).

The second step is to perform a dilation (the same spatial filtering algorithm) of the above-selected voxels by 6 voxels *up*. We will get a 6 voxels thick layer on top of all horizontal surfaces, in such a way that the top of the layer on one step at a stairway will be adjacent to the bottom of the layer on the next step. Under the assumption that any place in a building can be reached from any other place by moving through the navigable space, the dilation layer above this navigable space will be one single, connected voxel volume. This volume may be separated from other dilation layer pieces, for example lying on chairs, tables, cabinets and window sills. A 3D connected-component algorithm can assign unique numbers to all the disconnected pieces: every voxel in each piece will get the number of that piece as the voxel value. By making a histogram of those numbers, the largest piece of dilation layer can be identified, and the hard voxels underneath form the navigable space of the building. These are the green voxels in Fig. 1.

During the third step in the navigation process the identify one or more voxels in the navigable space the *destination* in a routing exercise. For example, one destination voxel can be placed at each exit of the building.

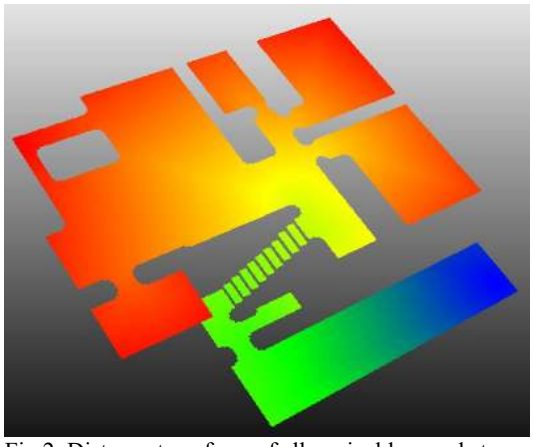

Fig 2. Distance transform of all navigable voxels to an exit voxel at the right side on the ground floor.

Next, we compute the distance of *all other* voxels in the navigable space to the nearest destination voxel (the nearest exit). This is done by a distance transform (DT), an image operation devised by Borgefors in the late seventies. The original DT computes the distance from each background (= source) pixel in a binary image to the nearest object (= destination) pixel, in a sequence of two (recursive) 'filters'. This can be expanded to a

three-valued case, which has 'obstacle' pixels in addition to source and destination pixels, which are excluded from the operation. This makes the process iterative: it continues until it stabilizes. Furthermore, the extension from 2D to 3D has been made long ago. In our case, a destination voxel is placed at the exit on the ground floor, the blue area in Fig. 3. The navigable-space voxels and the dilation layer are made 'source voxels', and all other voxels are marked 'obstacle'. The resulting distances (for the navigable space voxels) are shown colour coded in Fig. 2. DT gives an approximation of Euclidian distances with a maximum error of a few percent and is very efficient computationally.

In the final step of the navigation process one or more source voxels are chosen. From those, a path to the (nearest) destination can be found by 'swimming' downstream through the distance field (Fig. 3).

Fig. 3 : Paths to the exit from a number of randomly selected source points

4. FLEXIBILITY, FURTHER EXPERIMENTS

A preliminary conclusion from the above result might be that the operations so far are purely geometrical. No semantic information concerning floors, walls, stairs, etc. was required to establish the navigable space and test for its requirements, nor to perform the rest of the computations. The good news of this is, that it makes the methodology quite independent of its data sources. We will show two more case studies to illustrate this.

Voxel Model Reconstruction from a point cloud

The first of those concerns the same apartment as above, but now using a model that is reconstructed from a point cloud – albeit one that was not obtained by laser scanning.

Using the voxel model, we simulate laser scans (range images and point clouds), recorded at different scanner positions. From a defined position, the (virtual) scanner emits laser beams into many discrete (theta, phi)-directions.

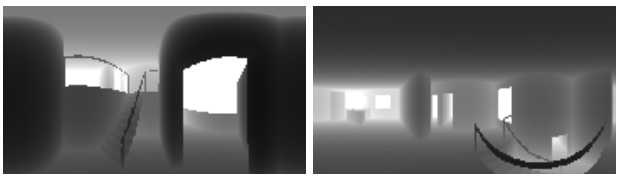

Fig. 3 Two sample range images

We compute for each beam which voxels (relative to the scanner position) it eventually traverses. The nearest one of those with value 1 defines the distance measurement at that direction, which is stored at that (theta, phi) position in the range image. Beams that are not hitting any 1-voxel disappear into infinity and remain undefined in the range image; they will generate no points in the point cloud.

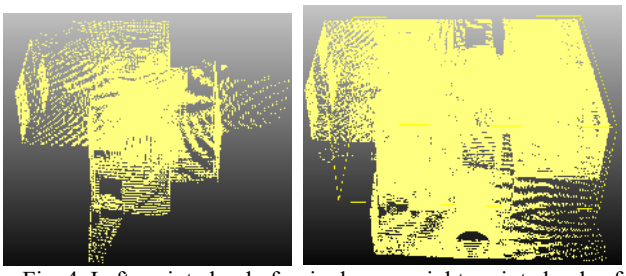

Fig. 4. Left: point cloud of a single scan; right: point clouds of seven scans combined

On the fly, we propose a method to create a voxel model from a set of range images made by (simulated) laser scanning, if the orientation parameters are known - in our simulated scans they are. Given a scanner position, plus a direction and a distance of one beam, exactly one voxel will become *hard* in the generated voxel space. Furthermore, all voxels on the line between the scanner position and that *hard* voxel are apparently air (or perhaps a window) and will be marked *transparent*; undefined range-image elements only generate *transparent* voxels, without a *hard* voxel at the end of the beam. Repeating this process for all laser beams of all the range images, starting off with an empty space where all voxels are *unknown*, we will end up with three classes of voxels: *unknown*, *hard*, and *transparent*. When processing all the beams of a set of scans, very many voxels will be assigned a value (*transparent* or *hard*) several times. In those cases, *hard* gets the priority.

Fig. 5. Reconstructed voxel model with three classes. Left: *hard*, centre: *transparent*, right: *hard* + *unknown*

After this, the transparent space is considered navigable (preventing routes through windows remains the operator's responsibility: close curtains to avoid confusion). This as opposed to more traditional approaches, where the point cloud only defines the *hard* voxels, while it remains uncertain whether the remaining space is either completely transparent or perhaps partially occluded or otherwise not scanned. It may contain other *hard* objects. In the presented method, the union of the *hard* and the (remaining) *unknown* voxels bounds the navigation space and

can be used to evaluate navigation constraints (overhead space, passage width, height jumps).

We perform the same navigation operation as in section 3 in the reconstructed model and compare the results.

The above workflow makes it easy to design and evaluate different scanning strategies and station setups. The developed system seems helpful if one intends to scan a building where a (perhaps outdated) BIM model is already available. (BIM-to-voxel conversion is on its way.)

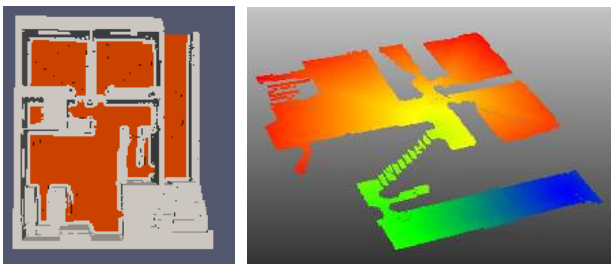

Fig. 6. Navigable floorspace (left) and distances to exit point (right) in voxel model reconstructed from point clouds

The more important result, however, is the reconstruction using three classes. In general, with "better" scanning (more scans, better coverage of the scene) one receives more *hard* voxels, which in a traditional approach would lead to a reduction of the navigable space: the number of obtained possible routes goes down, but their reliability (the likelihood that a route is traversable in reality) increases. In our three-class approach, "better" scanning from more positions increases the number of *hard*, but also of *transparent* voxels, and thereby the 'amount' of navigable space. Therefore, the number of possible routes goes up as well. Meanwhile, the reliability of the obtained routes was already high from the beginning. We think this is the preferred methodology.

And the even more important result in the current context is that good navigation can be performed based on a point cloud, after transferring it into the voxel domain. The main reason for this is the purely geometric nature of the methodology, requiring no recognitions of semantical classes or objects. If the voxel are more or less in the right positions, the results will be decent.

Arboretum Professional Centre, reconstruction from floorplans

As a last case study, we present a somewhat larger office building, reconstructed from floorplans. The floorplans were sort of randomly selected from Internet, in order to show we could use 'any' floorplan to do the analysis. They concern an office building in Seattle, WA, USA, called the Arboretum Professional Centre.

Floorplans of the three floors were given (ground, second and third floor). We set up a small work flow to generate a voxel model out of these, requiring a bit of manual editing (Fig. 7). First, we remove annotations, ornaments, furniture, doors, indications how windows open, etc., and basically only keep the walls. The most challenging part is obviously to reconstruct the stairs: for this we edit the floorplans again to create an impression of the "slab" of concrete underlying each floor as a black polygon, on which we indicate the heights of the different steps of the stairways in grey scale – all this using the flood-fill operation of xpaint in Linux.

The floorplans we found on Internet had room size annotations, from which we derived a 'scale' of 20 pixels per foot (30.5cm), or a resolution of 1.51cm. We reduced the images by a factor 3 to get a 4.55cm pixel (and voxel) resolution (Fig. 7).

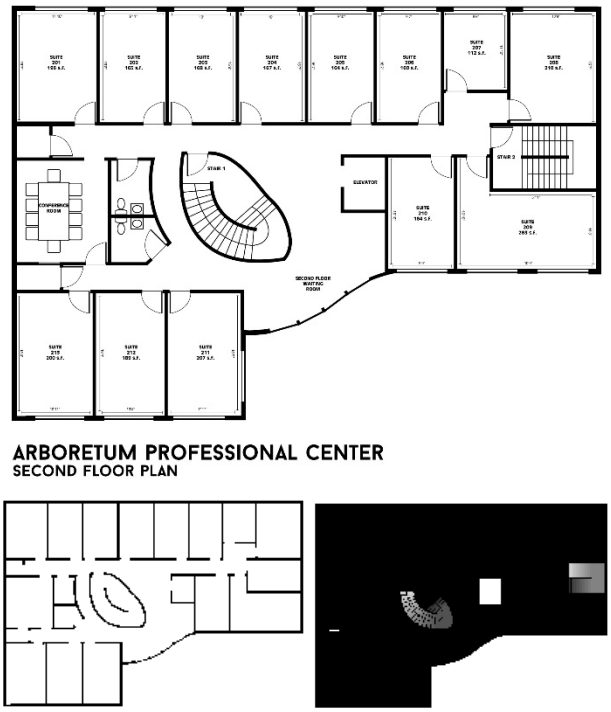

Fig. 7 Original floorplan (top) and edited versions for walls (bottom left), slab and stairways (bottom right)

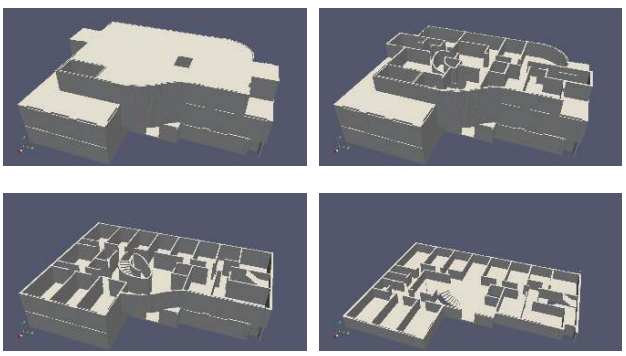

Fig 8: the entire building, including the roof (top left) as well as 'cut-off' models at three different heights, showing the layouts of the three floors.

All the edited floorplans (three sets of walls and four slabs, two of which have stairways), are read into a little piece of software that reconstructs the voxel model in a straightforward fashion, while using some assumptions of various heights and floor thicknesses. The hole in the centre of the model concerns the elevator, which was left out of all considerations (Fig 8). The model occupies a space of 519 x 674 x 196 voxels.

To start navigation processing, we select a 5-voxel thick layer of 'air' voxels that are located above 'hard' voxels and use the histogram on the connected component labelling to find the

navigable space (Fig. 9) just above the floors and the stairways (in this example, we omitted taking the overhead height into account, because the model was reconstructed with sufficient height all over; the only exception might be under the stairs on the ground floor).

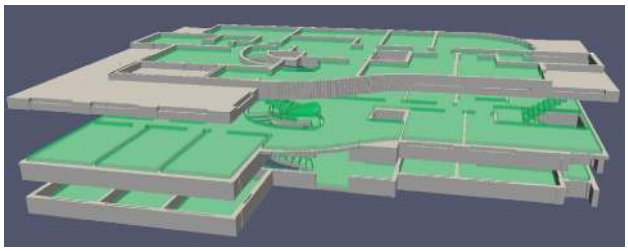

Fig 9: Navigable space as the largest connected component in the air layer just above floors and stairways

To continue the navigation process, we selected one destination voxel near the main building entrance, and another using the emergency exit at the right front side of the building. The two resulting distance fields, denoting the distances from anywhere in the building to the nearest destination voxel, are shown in Fig. 10. Next, specific voxels can be selected as starting points to compute routes to the nearest of the exits, by floating downward through the distance fields.

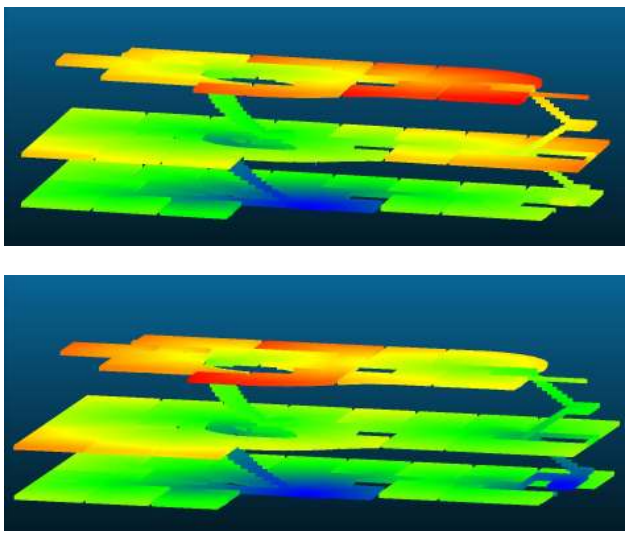

Fig. 10: Distance fields from anywhere in the navigable space to a single building exit (top) or multiple (bottom) exits (in blue).

5. CONCLUSION

We have shown three ways to reconstruct voxel models from different data sources: a set of hand measurements of a small apartment, a (simulated) set of laser scans of the same apartment, and a set of floorplans of a three-story office building.

As a by-product, we have demonstrated laser point cloud simulation from voxel models, as well as its inverse: to translate a point cloud voxel model with three classes: object, air and 'unknown'. The main goal of this study was to illustrate that voxel models are a highly suitable basis for performing indoor navigation, irrespective of the origin of the models. The reason for this is, that the algorithms are purely geometrical, and do not rely on the correctness of classification or object recognition.

The steps in the navigation process are 3D versions of well-known 2D raster operations, such as convolution, morphology, distance transform and connected component labelling. In our opinion, the paper once more illustrates the general usefulness of representing 3D spatial information using voxels.

REFERENCES

Diakité A. A. and S. Zlatanova, 2018, Spatial subdivision of complex indoor environments for 3D indoor navigation, International Journal of Geographical Information Science, 32(2), pp. 213-235

Khoshelham, K. and S. Zlatanova, 2016, Editorial to Sensors for Indoor Mapping and Navigation, Sensors 2016, 16(5), 655

Li, F., S. Zlatanova, M. Koopman, X. Bai, A. Diakité, 2018, Universal path planning for an indoor drone, Automation in construction, vol. 95, November 2018, pp. 275-283

Sithole, G. and S. Zlatanova, 2016, Position location, place and area: an indoor perspective, *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., III-4, 89-96, doi:10.5194/isprs-annals-III-4-89-2016*

Xiong, Q, Q. Zhu, Z. Du, S. Zlatanova, Y. Zhang, Y. Zhou, Y. Li, 2016, Free multi-f-loor indoor space extraction from complex 3D building models, Earth Science Informatics, 9 (32) , pp. 1-15

Zlatanova, S., G. Sithole, M. Nakagawa, and Q. Zhu, 2013, Problems In Indoor Mapping and Modelling, *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XL-4/W4, 63-68, https://doi.org/10.5194/isprsarchives-XL-4-W4-63-201*

Hwang, K. and N. Ahuja, A potential fied approach to path planning, 1992, Robotics and Automation, IEEE Transactions on 8 (1) 2332.

Vörös, J., Low-cost implementation of distance maps for path planning using matrix quadtrees and octrees, 2001, Robotics and Computer-Integrated Manufacturing 17 (6) 447-459.

Bandi, S. and D. Thalmann, Space discretization for efficient human navigation, 1998 in: Computer Graphics Forum, Vol. 17 of Computer Graphics Forum, Wiley Online Library, pp. 195-206.

Kitamura, Y., T. Tanaka, F. Kishino, M. Yachida, 3-d path planning in a dynamic environment using an octree and an articial potential Field, 1995, in: Vol. 2 of Intelligent Robots and Systems 95, Proceedings. 1995 IEEE/RSJ International Conference on, IEEE, 1995, pp. 474{481.

Koopman M., 3D path-finding in a voxelized model of an indoor environment, 2016. MSc Thesis Geomatics TU Delft, Netherlands.
http://repository.tudelft.nl/islandora/object/uuid:13788271-e19d-41e1-b827-fe7535a66281/datastream/OBJ/download