# Generating Smooth Virtual Reality Maps Using 3D Building Blocks

**B. J. Wimpey and W. D. Potter**
Computer Science Department, University of Georgia, Athens, GA, USA

**Abstract -** *In this paper, we describe a 3D modeling program called RMBL3D, Realistic Maps Built like Legos, which manipulates the architectural repetition in building structures to produce realistic 3D models. This paper details the features and aspects of the RMBL3D program and tests the viability of using 3D objects as building blocks to generate a virtual map. RMBL3D is designed to be the visualization program of an autonomous mapping robot. The program takes in a description of the map in the form of character symbols representing objects in the environment and reproduces the 3D version of the map, one that can be explored and navigated by a user.*

**Keywords:** Mapping, 3D Modeling, Robotics.

## 1 Introduction

The use of 3D maps and models has become quite important. Maps in 3D provide information that 2D maps contain, such as object locations and scale, while also adding depth and realism to the viewer. Foreign cities with corresponding 3D maps can provide a tourist with a virtual walkthrough of the land and building layouts before the person ever sets foot in the city. Maps and models in 3D can also be used in military training scenarios to prepare soldiers before their lives are put into any danger. The more accurate the textured images and locations of the digitized objects, the more realistic the experience is to the viewer.

It is common to find repeated structures in man-made architecture. Inside of buildings you find walls with long stretches of painted surfaces or column after column of bricks or cinder blocks. Occasionally these sequences of objects are interrupted by doors or hallway archways. Often the greatest distinguishing feature between repeated objects is where they are located in relation to one another. If one had realistic looking 3D models representing different objects found in real life, such as a column of red bricks, a column of white cinder blocks, or a door to name a few, and if one was able to place these objects into 3D space, then one could conceivably build accurate 3D representations of the environment.

### 1.1 Related Work

Mapping in 3D has been used to model environments dangerous for humans; Thrun et al. [14] used a sensor equipped cart and a mobile robot for 3D mapping of abandoned mines. Modeling in 3D has also been used for heritage site recording and archiving techniques. El-Hakim et al. [3] applied their methods to modeling churches in Italy. They described combining image based modeling for large structural elements with laser range scans for detailed modeling. Their image based modeling, however, was interactive. Parish and Müller [11] introduced CityEngine, a program that used a set of production rules to generate an urban model of buildings, roads, and highways. Müller et al. [9] described their shape grammar for modeling different building types, CGA Shape, which was added into CityEngine. They created production rules used to model Pompeii based on ground plans and architectural styles. The result is a stunning “What it might have looked like” visualization technique. In [10], Müller et al. procedurally recreated stone buildings of the ancient city of Xkipché in Mexico. CityEngine with CGA Shape produced amazing results, but this program is more for stochastic model generation for say, games or movies, as opposed to mapping. El-Hakim et al. [4] described copying and pasting repeated architectural components in image based modeling. The user first models one component, for example a window of a building, and then identifies other areas in the overall model that the component can be reused. The technique is interactive, however, and it requires a user to select points common between the component and the areas in the model it is to be reused.

A popular choice for modeling is the use of laser range finders, but different techniques exist for representing the model data. Howard et al. [7] used a mobile robot to map part of the University of Southern California campus, but the resulting 3D map was presented as point clouds, leaving it to the viewer to interpret objects and details. Surmann et al. [13] produced a mesh of range data in an octree representation. Hahnel et al. [6] modeled structures by identifying large planar shapes in the range data results, but no object separation or texturing was applied. Biber et al. [1] produced a 3D model of an office environment by using a mobile robot equipped with a panoramic camera and a laser range finder. The textured images were not very detailed, so the resulting 3D model was lacking in quality. Stamos and Allen [12] attempted to create CAD models from range and image data. They use some human interaction to combine the 3D data with the 2D image data. While the resulting models are CAD representations, they do not use repeated architectural objects in construction of the model. Haala et al. [5] created urban environment models by incorporating height data acquired from airborne laser range finders, but they also used information from existing ground plans. Our approach can be used to generate models without being locked into ground plan data.

### 1.2 RMBL3D - Realistic Maps Built like Legos

The goal of RMBL3D is to be the visualization program of an autonomous mobile robot. The robot explores the environment and records turns and objects into a text file we call the symbol map, which is then fed into RMBL3D to view the resulting 3D map. Using RMBL3D, a user would then be able to traverse the virtual environment. To test our idea, we set out to use objects found within the Boyd Graduate Studies Research Center on the campus of the University of Georgia. Using images and size information of objects from the interior of Boyd, we designed 3D models of objects to be placed into the 3D world. Since RMBL3D is equipped with 3D building blocks representative of the objects one might find in Boyd, it is not only able to produce 3D maps of Boyd hallways and foyers, but it is also able to produce maps of non-existent locations like kilometer long hallways or square kilometer foyers, just with the objects common to Boyd. The entire system of the exploring mobile robot, object mapping and 3D modeling is ongoing research, but we present here the RMBL3D program in order to adequately explain its features and details.

## 2 Surveying Boyd for Digitization

Our motivation was to manipulate architectural repetition in order to produce realistic maps in three dimensions. In order to realize the goal of realistic 3D models and maps, we had to assess the environment to make sure we encode representative objects as well as realistic heights, widths and other measurements for our 3D building blocks. The Boyd building has an interior one might find in other office buildings, and since it has convenient access to researchers, we took it as our location for experimentation.

We had to decide on a segmentation size that would not be too big, which might lead to unrealistic 3D models, but also not be too small, eliminating the benefits gained from recognizing and mapping repeated structures. We found that we could map out a floor of Boyd by sectioning the environment into one meter segments, with each meter segment being a wall section, door, column or another object from the environment. An elevator was a special case object; it needed to be two meters wide. Using our modeling approach, object locations and sizes will be rounded to the nearest meter. For example, if a door along a hallway is separated from another door by only two feet of a brick wall object, the brick wall segment in the 3D map will be rounded to a whole meter. A column of Boyd is about 66 centimeters square, and we model it using a square meter column object. The widths of hallways were modeled as 2 meters across. Using these mapping rules, the models generated are not exact, but for our application, the approximate models are robust and the rounding error does not cause any major problems. Furthermore, Boyd walls were of different material, often bricks or cinder blocks. Doors do not go from the floor to the ceiling, so brick or cinder overhangs would also need to be modeled over doors. Hallway turns also needed to be represented in the model in the form of L or T junctions. In order to model a foyer, we needed segments of empty space. Furthermore, we needed floor tiles, ceiling tiles, and all of this took place on a floor of Boyd with a height (floor to ceiling) of 3.048 meters.

## 3 Representing the Symbol Map

Since the origin of the symbol map is of no concern to the RMBL3D program, as long as the symbols make sense to the program, a 3D model can be produced. This modularity allows us to develop and test the extent of the RMBL3D capabilities without it specifically relying on the mobile robot to, say, traverse a kilometer long building layout. Since this was a mobile robot project, however, we had to keep in mind the way in which the robot would detect the objects and how it should be represented in a symbol map. Therefore, the symbol map layout is formatted to support an incremental, robocentric mapping behavior.

TABLE I
IMPLEMENTED OBJECT SYMBOLS

| Symbol | Object Description |
|---|---|
| D | door with brick overhang |
| d | door with cinder overhang |
| b | brick wall |
| w | cinder block wall |
| E | elevator with brick overhang |
| e | elevator with cinder overhang |
| c | column |
| n | empty space |

### 3.1 Object Symbols

Table I lists the object symbols implemented for RMBL3D. According to our robocentric mapping behavior, for each represented meter segment, the objects the robot detects on its left and its right are recorded as pairs of symbols in the symbol map. The robot then moves and detects again. The robot updates the map incrementally and keeps the map as a string of characters. Therefore, the format for recording object symbols in the symbol map is $O_{R1}O_{L1}O_{R2}O_{L2}\ldots$ where for each object symbol $O_{Si}$ : O is one of the symbols {D,d,b,w,E,e,c,n}; S is the robocentric side the object is located (left or right); i is the segment number in which the object was detected. For example, a symbol map string that starts "DbbD" represents a door with brick overhang on the right and a brick wall segment on the left in meter number one, and the second meter of the map is of a brick wall on the right and a door with brick overhang on the left.

## 3.2 Representing Turns and Multiple Floors

To represent a single 90 degree turn in the symbol map, we used the five symbol format $DO_{E1}O_{E2}O_{B1}O_{B2}$ where D is the robocentric direction of the 90 degree turn ('l' or 'r' for 'left' or 'right', respectively), $O_{Ei}$ is an object symbol representing one of the two end cap objects, and $O_{Bi}$ is an object symbol representing one of the two beginning cap objects. We use the terms "end cap" and "beginning cap" to represent the space to be filled in by objects when the robot makes a turn or when two hallways join. Since our modeled hallways are two meters across, if a robot is mapping from the center of the hallway and needs to turn to start mapping an intersecting hallway, the "caps" are the extra objects to be mapped that join one hallway with the next. Alternatively, the caps are also the objects to be mapped when a robot makes a turn in a foyer. Fig. 1 highlights this area of interest. An example of the five symbols to represent a 90 degree turn in the symbol map would be "rdwwd", which would mean a 90 degree turn to the robot's right, with the end cap objects being 'door with cinder overhang' and 'cinder block wall' and the beginning cap objects being 'cinder block wall' and 'door with cinder overhang'. This is a typical example of an L intersection. Should the robot encounter a turn, and it is mapping a foyer or a T intersection in which there are two directions it could take, it would need to insert empty space into the symbol map. A five symbol example of a T intersection in which the robot chooses to continue mapping to the left but leave the right area open for future mapping might be "lnnww", where the end caps are left as empty space and the beginning caps are two cinder block wall segments. Visual examples of intersections and turns that needed to be represented are shown in Fig. 2, and how they were modeled in RMBL3D can be seen in Fig. 3. The '+' symbol was implemented such that we could map multiple

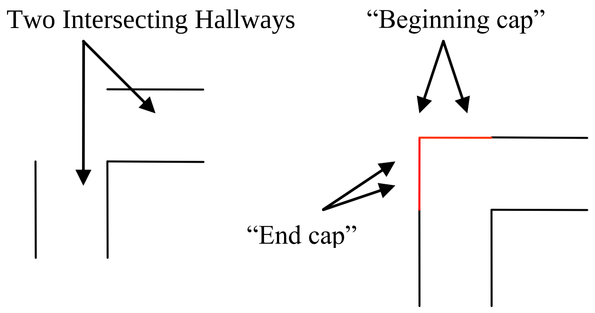

Fig. 1. Hallway caps are the objects that need to be modeled so that two intersecting hallways meet and leave no open gaps.

levels of the same building into one 3D model map. The '+' symbol would move the mapping plane 4 meters up.

Whereas using laser range finders on robots will allow the robot to map areas from a greater distance away, we decided that object detection via computer vision combined with this grid-like mapping allows for more methodical and therefore more reliable object detection and mapping. The

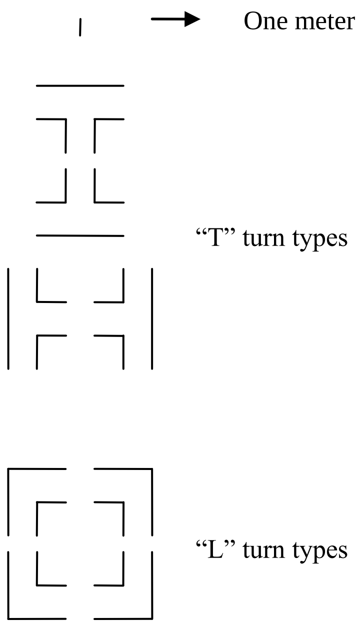

Fig. 2. 90 degree turn types take the form of Ts or Ls oriented in different ways as shown in the examples above. We needed to be able to represent all of them.

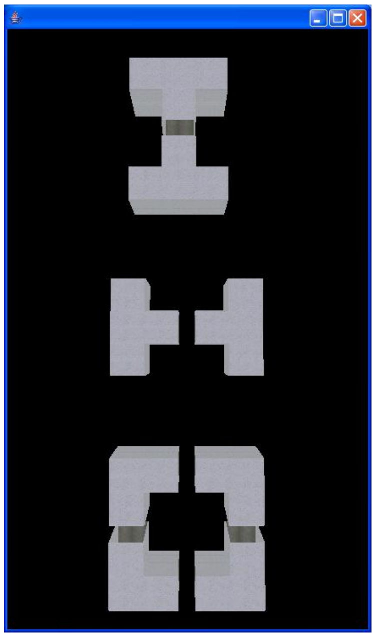

Fig. 3: This is a bird's eye view of how the 90 degree turn types were represented in RMBL3D.

robot would be able to map areas in a snake-like pattern, two meters in width at a time.

## 4 Implementation

The RMBL3D program was written in Java and used Java 3D [8] and the JWSU Virtual Reality toolkit [2]. The Java 3D package is available for free from Sun Microsystems. Java 3D has been developed for years and includes a robust API and many tutorials. The JWSU toolkit is a powerful and free utility that allows for mobility in the 3D world and also includes support for virtual reality devices such as head mounted displays, powerwalls, and PinchGloves or CyberGloves.

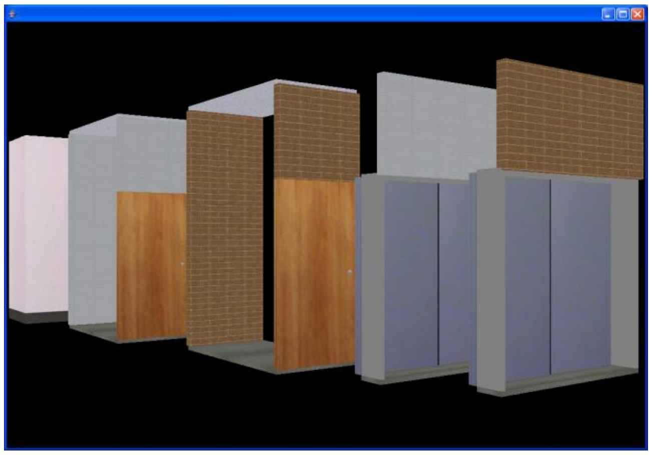

Fig. 4. A view of the different building blocks used in RMBL3D.

### 4.1 The Java 3D World and Object Models

Since the spatial environment in Java 3D is represented in meters, modeling the objects and specifying coordinates based on our previous assumptions was straightforward. We use the Java 3D geometry primitive classes for boxes and spheres in conjunction with Java 3D texture mapping classes to texture the objects with detailed images. A RMBL3D view of the object models can be seen in Fig. 4. Using Java 3D, it was also easy to create an object that runs in different directions by simply switching around two of the object's dimensions. For example, suppose a viewer was facing a three dimensional wall object that was one meter wide, three meters tall, and ten centimeters in depth. One might also describe this object, from the viewer's perspective, as running east to west in front of the viewer. To create an object running north to south, using Java 3D it was simply a matter of switching the object's width and depth properties, producing an identical object flowing in a perpendicular direction.

There were some slight oddities in describing object sizes in Java 3D. In creating a box object, it would seem the intuitive way to describe it would be to provide the object's width, height and depth. Using the box geometry class of Java 3D, however, the object is centered at the origin and stretched to the given X, Y, and Z dimension parameter values. Thus, if one does not keep this in mind, one might unintentionally end up with object dimensions double the intended size.

After the symbol map is read into the RMBL3D program, the map is parsed into pairs of objects, the five symbol set representing a turn, or into single symbols such as the '+' symbol. During the parsing, the program simulates the movements the robot took in mapping the environment by keeping an updated directional variable and its total area traversed, allowing it to correctly place objects into the 3D world with updated (X,Y,Z) values. The program starts building the 3D virtual map at the origin (0,0,0), and it assumes the robot started out facing north. The symbol map is robocentric, meaning left and right are always in relation to the robot and the direction it is traveling. The program therefore had to account for which direction the robot was traveling when it recorded the symbols. The program updates the direction as turns are encountered in the symbol map. Also, since we assumed objects were in meter segments, and since the Java 3D universe is in meters, as a new pair of symbols is parsed by RMBL3D and needs to be inserted into the 3D map, RMBL3D simulates taking a meter step in the environment by incrementing or decrementing the appropriate (X,Y,Z) coordinate variables. Also, updating the compass direction variable follows the graph in Fig. 5, with the nodes being the up-to-date direction and the edge labels R or L being the direction symbol encountered in the symbol map and the new compass direction which results from such a turn. When RMBL3D encounters the '+' symbol in the symbol map, which signifies to start mapping on a new floor, the program jumps the Y variable up 4 meters so that now it simulates the robot on a new plane in the 3D world. Using this, we can map multiple levels of the same building in one 3D map.

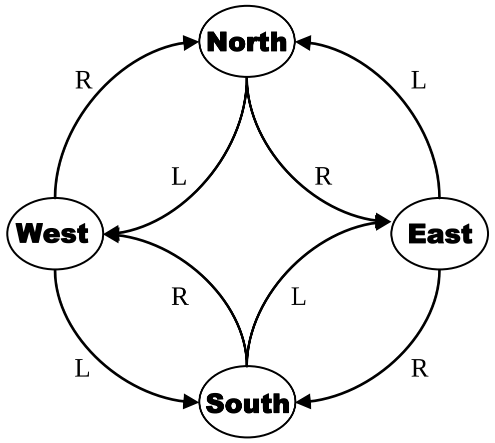

Fig. 5. This graph illustrates direction change in the 3D world depending on which way one turns. The edges indicate the turn direction symbol encountered in the symbol map, while the nodes are the up-to-date directions resulting from such a turn.

Once the program finishes reading through and placing all of the objects, turns, and level changes into the 3D world, it displays the result as a smooth flowing 3D map that one can navigate and explore. Many of the traversal behaviors are enabled by using the JWSU toolkit and the keyboard arrow keys and mouse, but Java 3D also allows for implementation of custom keyboard navigation features. To test this and to allow for some precise camera viewing locations, RMBL3D implements different camera views by translating the user's viewing location to the origin, to +100 meters in the Y direction above the origin (which would move the user to a flyover location), or +100 meters in the Z direction from the origin (which would move the user's view 100 meters backwards from the origin).

## 5 Results

By identifying and using repetition and creating a virtual 3D map from a string symbol map, we achieved excellent results (see Fig. 6). Fig. 6a shows the Boyd foyer we attempted to recreate, and Fig. 6b is a view of the foyer modeled in RMBL3D. This is a navigable model, so a user can walk around in the 3D map and experience it similar to a first person point-of-view video game. Fig. 6c shows the result of a one kilometer long hallway modeled in RMBL3D with Boyd objects. Fig. 6d shows a hallway in Boyd, and Fig. 6e shows the resulting model in RMBL3D. Fig. 6f illustrates the use of the '+' symbol and the results one can get from modeling different floors in the same 3D map. RMBL3D was designed to take a map of turns and object symbols from an exploring mobile robot, but to test the capabilities and limitations of the modeling program, these examples were created by typing the symbols into the symbol map and running it through RMBL3D to view the results.

## 6 Conclusions and Future Work

Virtual reality offers a rich, realistic experience for exploring and navigating 3D maps. 3D maps have become useful in site archiving, in training, and in virtual tourism. RMBL3D is a program developed for the 3D visualization of an object map. One can test any symbol map with the RMBL3D program to see the 3D map it creates, within the limits of the defined symbols and objects. Future work includes adding the autonomous robot into the system to explore the environment and autonomously acquire the object symbol map. Furthermore, we need to identify the appropriate object recognition techniques to apply to the autonomous robot, perhaps exploring the use of stereo correlation to help supplement odometryinformation. Also, continued work is needed to refine and enhance RMBL3D. Though the assumptions made were still useful in creating realistic looking 3D maps, RMBL3D is still limited to meter segments.

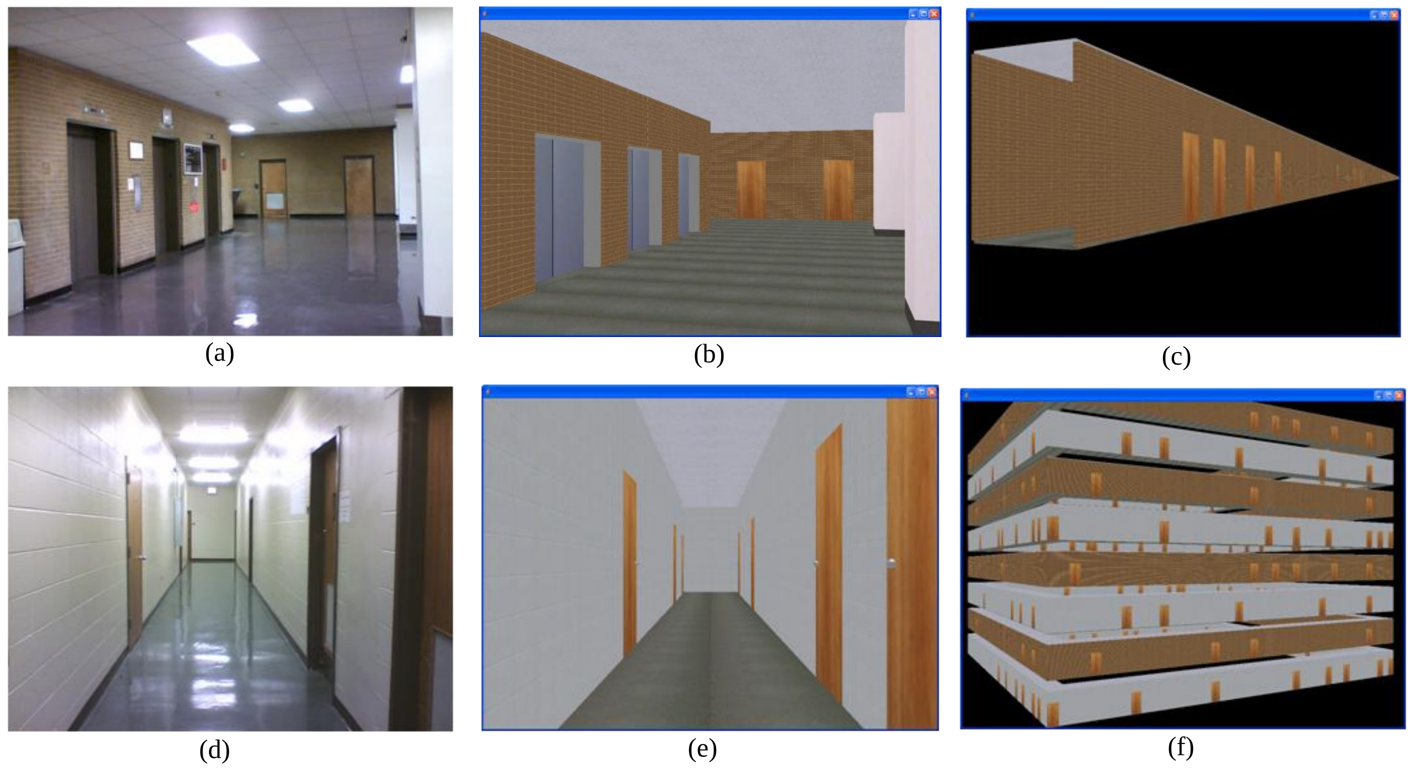

Fig. 6. (a) Part of a foyer in the Boyd Graduate Studies Research Center building to be mapped. (b) The view of the foyer as represented in RMBL3D. (c) RMBL3D representation of a one kilometer long hallway with Boyd objects. (d) Another hall in Boyd to be mapped. (e) The hall as viewed in RMBL3D. (f) RMBL3D representation of multiple floors with Boyd objects.

## 7 References

[1] Biber, P., Andreasson, H., Duckett, T., Schilling, A. "3D modeling of indoor environments by a mobile robot with a laser scanner and panoramic camera," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 4, pp. 3430-3435, 2004.

[2] Deligiannidis, L., Weheba, G., Krishnan, K., Jorgensen, M. "JWSU: A Java3D framework for virtual reality," in *Proc. of the International Conference on Imaging Science, Systems, and Technology*, pp. 312-319, June 2003.

[3] El-Hakim, S., Beraldin, J., Picard, M., Vettore, A. "Effective 3D modeling of heritage sites," in *Proc. Fourth International Conference on 3-D Digital Imaging and Modeling*, pp. 302-209, 2003.

[4] El-Hakim, S., Whiting, E., Gonzo, L. "3D modeling with reusable and integrated building blocks," in *Proc. 7th Conference on Optical 3-D Measurement Techniques*, Oct 2005.

[5] Haala, N., Brenner, C., Statter, C. "An integrated system for urban model generation," in *Proc. ISPRS Congress Commission II Symposium*, pp. 96-103, July 1998.

[6] Hahnel, D., Burgard, W., Thrun, S. "Learning compact 3D models of indoor and outdoor environments with a mobile robot," in *Robotics and Autonomous Systems*, Vol. 44, pp. 15-27, July 2003.

[7] Howard, A., Wolf, D., Sukhatme, G. "Towards 3D mapping in large urban environments," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 419-424, 2004.

[8] Java 3D homepage, https://java3d.dev.java.net/

[9] Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L. "Procedural modeling of buildings," *ACM Transactions on Graphics (TOG)*, Vol. 25, Issue 3, pp. 614-623, July 2006.

[10] Müller, P., Vereenooghe, T., Wonka, P., Paap, I., Van Gool, L. "Procedural 3D reconstruction of Puuc buildings in Xkipché," *7th International Symposium on Virtual Reality, Archaeology and Cultural Heritage*, pp. 139-146, 2006.

[11] Parish, Y. and Müller, P. "Procedural modeling of cities," in *Proc. of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 301-308, 2001.

[12] Stamos, I. and Allen, P. "3-D model construction using range and image data," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 531-536, June 2000.

[13] Surmann, H., Nuchter, A., Hertzberg, J. "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, Vol. 45, pp. 181-198, Dec 2003.

[14] Thrun, S. et al. "A system for volumetric robotic mapping of abandoned mines," in *Proc. of the IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 4270-4275, May 2003.