MapGENIE: Grammar-enhanced Indoor Map Construction from Crowd-sourced Data

Damian Philipp, Patrick Baier, Christoph Dibak, Frank Dürr, Kurt Rothermel
Institute of Parallel and Distributed Systems
University of Stuttgart, Stuttgart, Germany
Email: first.last@ipvs.uni-stuttgart.de

Susanne Becker, Michael Peter, Dieter Fritsch
Institute for Photogrammetry
University of Stuttgart, Stuttgart, Germany
Email: first.last@ifp.uni-stuttgart.de

***Abstract*—While location-based services are already well established in outdoor scenarios, they are still not available in indoor environments. The reason for this can be found in two open problems: First, there is still no off-the-shelf indoor positioning system for mobile devices and, second, indoor maps are not publicly available for most buildings. While there is an extensive body of work on the first problem, the efficient creation of indoor maps remains an open challenge.**

We tackle the indoor mapping challenge in our MapGENIE approach that automatically derives indoor maps from traces collected by pedestrians moving around in a building. Since the trace data is collected in the background from the pedestrians' mobile devices, MapGENIE avoids the labor-intensive task of traditional indoor map creation and increases the efficiency of indoor mapping. To enhance the map building process, MapGENIE leverages exterior information about the building and uses grammars to encode structural information about the building. Hence, in contrast to existing work, our approach works without any user interaction and only needs a small amount of traces to derive the indoor map of a building. To demonstrate the performance of MapGENIE, we implemented our system using Android and a foot-mounted IMU to collect traces from volunteers. We show that using our grammar approach, compared to a purely trace-based approach we can identify up to four times as many rooms in a building while at the same time achieving a consistently lower error in the size of detected rooms.

I. Introduction

Location-based services (LBS) such as navigation services, point-of-interest finders, or geo-social networks enjoy ever-growing popularity. This trend is driven by the availability of modern smartphones equipped with accurate positioning systems like GPS and the availability of detailed map information either from commercial providers or communities like the OpenStreetMap (OSM) project. However, so far these systems are mostly restricted to outdoor scenarios.

Looking at the current development of LBS, we see a strong trend to extend LBS to indoor scenarios such as indoor navigation or shopping assistants. An essential prerequisite for such services is the availability of detailed and accurate indoor maps. Although first map services such as Google Maps allow users to upload their indoor maps [1], creating these maps is still challenging and often involves labor-intensive manual tasks. Therefore, there is a strong incentive to automate the task of indoor mapping.

So far, only a few approaches for the completely automatic creation of indoor maps have been reported in the literature [2], [3]. Their basic idea is to automate the modeling of floor plans by using a set of pedestrian traces as input. Although these approaches are promising and show the general feasibility, they either rely on a large set of traces to compensate for the inaccuracy of indoor positions [2], [3] or they require the active collaboration of users collecting traces, for instance, to "explore" the borders of a room [2].

However, automatically constructing a floor plan from few, possibly inaccurate traces "on the fly" without user intervention remains a great challenge which we tackle in this paper. To this end, we propose the MapGENIE system for the automatic indoor modeling of floor plans from a small number of traces gathered opportunistically by mobile users. The basic idea of MapGENIE is to utilize different kinds of *structural information* to support the mapping process and, in particular, compensate for partially missing traces and positioning inaccuracies without relying on a large set of traces. As already shown by existing work [4], the outline of the building exterior can be utilized to infer the boundaries of floor plans and to align traces since corridors and rooms are often aligned along the building axes. However, additionally we are the first to utilize *indoor grammars* to support the mapping process. Indoor grammars are powerful tools which may also have an impact on the set-up of Building Information Models (BIM) in the near future. It has been shown in existing work that such grammars can be used to encode structural information for different kind of architectural domains, like buildings [5] or street networks [6]. Typically a floor plan also follows certain architectural principles. Considering an office building, an assistant's office is very likely located next to an executive's office and both rooms have specific dimensions. If the floor plan of a building contains such structural features, we can use them to correct and complete (as far as possible) a floor plan derived from traces which are typically inaccurate and might not cover a complete floor. To encode such structural information we use a grammar that includes, for instance, the dimensions of rooms, the number of rooms, the relative room ordering, geometric constraints, etc.

In detail, we make the following contributions: (1) We define an architectural framework and multi-step process for the automatic modeling of indoor maps supported by structural information. (2) We present an algorithm that, as a first step, derives an initial plan from a set of inaccurate traces. In this initial step, we use information about the building exterior to filter and correct traces. (3) We define a formal grammar

Published in Proceedings of the 12th IEEE International Conference on Pervasive Computing and Communications (PerCom 2014), Budapest, Hungary, March 24-28, 2014. ©IEEE 2014
http://dx.doi.org/10.1109/PerCom.2014.6813954

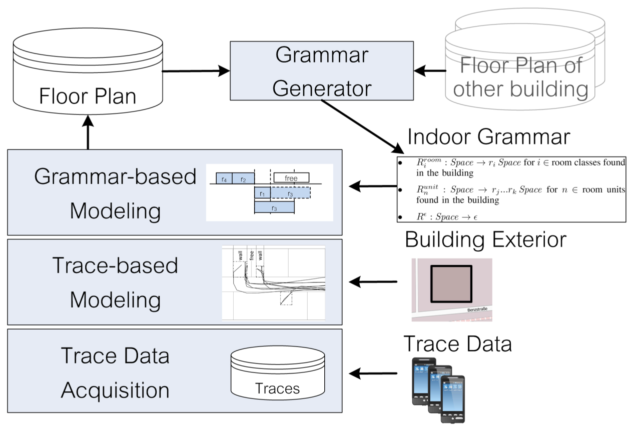

Floor Plan
Grammar Generator
Floor Plan of other building
Indoor Grammar
R_i^{room} : $Space \rightarrow r_i\, Space$ for $i \in$ room classes found in the building
R_n^{unit} : $Space \rightarrow r_j ... r_k\, Space$ for $n \in$ room units found in the building
R^{ϵ} : $Space \rightarrow \epsilon$
Grammar-based Modeling
free
Building Exterior
Benzstraße
Trace-based Modeling
Trace Data
Trace Data Acquisition
Traces

Fig. 1. System architecture showing the backend system and its inputs to the individual components.

describing structural information about room layouts. (4) We present a probabilistic algorithm that, in a second step, derives the most likely room layout from the initial (inaccurate and incomplete) plan. (5) We present a proof-of-concept implementation and evaluation in a real-world scenario. To this end, we collected more than 22 km of indoor traces (which are available to the community[1]), and we implemented an Android App for trace collection using a foot-mounted inertial measurement unit (IMU). Furthermore, we implemented a backend service for trace processing. We show that from only few traces our approach can derive hallways of a floor without user interaction and a complete floor plan including rooms using an indoor grammar. In comparison to a purely trace-based approach, we can identify up to four times as many rooms in a building while at the same time achieving a consistently lower error in the size of detected rooms.

The remainder of this paper is structured as follows: Section II presents the system model and architectural framework of our system. Section III describes the acquisition of the trace data in detail. In Section IV we describe the processing of the trace data including information from the building exterior, before Section V introduces the grammar which supports the trace processing step. In Section VI we present our real-world evaluation, including our prototype implementation that we built for the acquisition of traces with mobile devices, and discuss the results. Finally, Section VII presents related work before Section VIII concludes with an outlook on future work.

II. System Model and Architecture

Our system consists of a server-based backend system containing four different components which are described in the following in detail. Figure 1 gives an overview over all components with their respective input.

The *trace data acquisition* component receives trace data from mobile devices via a wireless communication network such as a WiFi network or a 3G/4G mobile network. To provide the trace data, we assume to have a number of trace collectors (*users*) that record traces with their mobile devices. The users record the traces in an opportunistic fashion, i.e., traces are recorded on the fly without requiring user interaction. Details on the trace recording process are given in the next section.

The *trace-based modeling* starts the processing of the uploaded traces by applying trace correction and alignment techniques. Unreliable and corrupted traces are filtered out, before further processing on the trace data is done. Subsequently, hallways are identified from the trace data and the spaces of the rooms are derived. These steps incorporate information about the building exterior in order to derive bounds for the indoor model. After having separated hallways from room spaces, individual rooms are identified on base of the trace data. The result of this processing step is a *trace-based indoor model*.

The resulting indoor model from the previous processing step is only based on trace data and, therefore, still inaccurate and incomplete. Hence, the *grammar-based modeling* applies structural knowledge to this model to derive a more accurate and complete model. A grammar defines constraints, for instance, for the size of rooms, and is tailored to individual building types. Based on the information contained in the grammar, a room layout proposal that contains the most likely room layout based on the trace data is derived and the system outputs a *grammar-based indoor model* that is based on all evidence from the available data and structural information.

In our paper, we assume that the grammar used for modeling is provided by a *grammar generator*. Such a grammar generator creates an indoor grammar by parsing a known indoor map of another floor of the same building or an indoor map of another building that has a similar architectural style. The detailed implementation of this grammar generator can be found in [7] and is beyond the scope of this work. For future work we envision that such a grammar can be automatically constructed on-the-fly based on areas of indoor maps that are already sufficiently validated by traces. Similar approaches have also been shown to be feasible in our work on 3D facade reconstruction [8].

In the next sections, we will describe each component of the system architecture in more detail.

III. Trace Data Acquisition

The goal of the *trace data acquisition* component is to provide a *set of odometry traces* from mobile users, which is used by the trace-based modeling component to derive the trace-based indoor model. To collect data, users are provided with an App they can install on their smartphone. When enabled, the App will collect the odometry trace data and periodically upload it to the backend service. (In our evaluation, the App is connected to a foot-mounted IMU that detects steps by using a Zero-Velocity-Update protocol [9].) Formally, we define an *odometry trace* $t = (\overrightarrow{\pi}, S)$ where $\overrightarrow{\pi}$ is the initial position and $S = \{s_1, ...s_n\}$ is an ordered set of steps. Each s_i is given as a (relative) 2D vector (x_i, y_i), annotated with a timestamp $s_i.time$ denoting the time the step began. The position $\overrightarrow{p_k}$ after the k-th step is calculated as $\overrightarrow{p_k} = \overrightarrow{\pi} + \sum_{i=1}^{k} s_i$, annotated with a timestamp $\overrightarrow{p_k}.time = s_k.time$.

To record an indoor trace we rely on inertial positioning, as absolute positioning systems (such as GPS) are not available with sufficient accuracy [10] in indoor settings. Their infrastructure-based indoor counterparts (using, e.g., WiFi beacons) are not usable for the reconstruction task at hand, as

[1]see http://www.comnsense.de/downloads

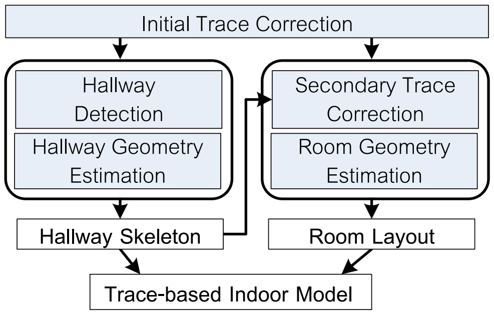


Fig. 2. Overview of the Trace-based Modeling component

their set-up itself involves mapping, such as fingerprints of the received signal strength or the locations of the beacons. Therefore, we can only use GPS to determine the initial position $\overrightarrow{\pi}$ before entering a building [2].

Although, we assume the use of a foot-mounted IMU, the relative positions of traces can also be derived from a variety of other sensors. Every modern smartphone has a built-in accelerometer, which can be used to track user movement. Since these built-in sensors are subject to high sensor noise [11], a large set of traces is necessary to cancel out the drift errors [2]. The use of an external IMU using a foot-mounted strap-down system with drift correction (cf. Fig. 11b) can reduce drift errors and, thus, perform well even with a small set of traces. However, this requires the deployment of additional hardware that is rather expensive [9]. Our system is agnostic of the actual inertial positioning system that is used for recording traces. To deal with drift errors, it includes an initial trace correction step that automatically preprocesses inertial traces, as described in the next section. In this paper, we consider the use of an IMU to record pedestrian traces. Implementing and evaluating the system with more noisy sensors is part of our future work.

IV. Trace-based Modeling

The goal of the *trace-based modeling* is to derive an indoor model for an observed floor of a building. Given an exterior model of the building and a set T of (noisy) odometry traces collected by the trace data acquisition component, we compute a set of hallways (the *hallway skeleton*) and a set of rooms (the *room layout*) that together form the *trace-based indoor model*.

As depicted in Fig. 2, detecting the hallway skeleton and detecting the room layout both are multi-step processes. As a preprocessing step to both processes, we first reduce the noise of the traces in the *initial trace correction* step. For the detection of the hallway skeleton, in the *hallway detection step*, we extract *hallway segments* from all traces and group these segments per hallway. These segments define the walkable area of each hallway, which is converted into a 2D rectangle representation in the *hallway geometry estimation* step, thus forming the hallway skeleton. To detect the room layout, we first reduce the noise of odometry traces further using the hallway skeleton in the *secondary trace correction* step. Finally, we find the room layout in the *room geometry estimation* step by analyzing all steps that were taken inside the building exterior but outside of a hallway. We present these steps in detail in the following subsections.

```
Require: Trace $t$, Exterior Model $E$, $minLength$, $\tau$
  $HS_t \leftarrow$ FINDHALLWAYSEGMENTS$(t, minLength, \tau)$
  for all $hs \in HS_t$ do
      $hs.chord \leftarrow (hs.start, hs.end)$
      $minAngle \leftarrow \min(\{\text{angle}(hs.chord, e) \mid \text{Exterior Wall } e \in E\})$
      rotate($t[p_s, p_{last}]$, $minAngle$)
  end for
```

Fig. 3. Initial Trace Correction

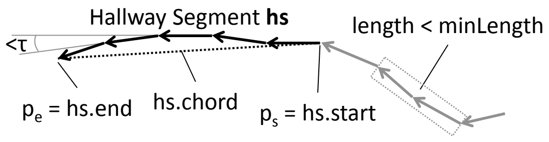


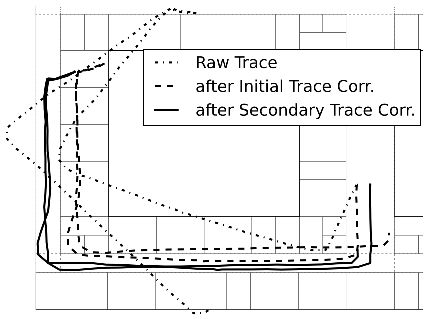
Fig. 4. Hallway Segments are defined as maximum-length sequences of steps in a trace, where the angle of any two subsequent steps is $< \tau$ and that are at least *minLength* long.

A. Initial Trace Correction

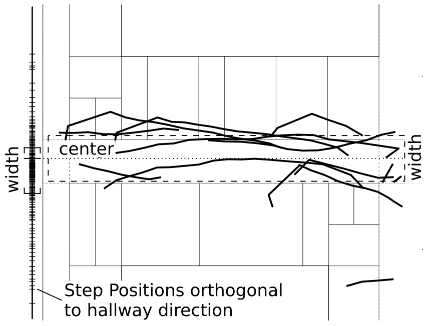
As detailed in Section III, readings from inertial measurement systems contain errors. As an example, consider the trace depicted in Fig. 5a, taken using a foot-mounted inertial sensor: Turning angles were recorded incorrectly, and due to sensor drift, the trace is slightly bent even though the user walked in a straight line. To reduce sensor noise, we leverage two observations: First, users commonly walk along the long axis of a hallway, and, secondly, hallways are commonly built parallel to an exterior wall of the building. Thus, given the exterior model of the building—as can be readily obtained from, e.g., OpenStreetMap—we rotate straight segments of a trace parallel to an exterior wall as depicted in Fig. 3 and 5a: We first extract a set of hallway segments HS_t (cf. Fig. 4) of at least *minLength* length from each trace t. The value of *minLength* must be chosen longer than the longest room that is presumably present in the building, so that no hallway is detected when walking along the length of such a room. Starting from the first hallway segment, for each hallway segment $hs \in HS_t$ we rotate the remainder of the trace including hs around $hs.start$ by the smallest possible angle so that $hs.chord$ is parallel to an exterior wall. Note that if a trace is extremely noisy, it may be rotated towards the wrong exterior wall. In this case, we say that a trace has been *corrupted*. Corrupted traces are detected and handled in the next step.

B. Hallway Detection & Hallway Geometry Estimation

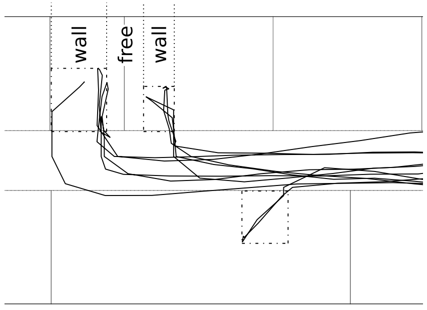
To derive the hallway skeleton, we begin by filtering out corrupted traces and find hallways that exist in the building from the remaining traces in the *hallway detection step*. Intuitively, we would say that a hallway has been found wherever a hallway segment is detected on a trace. However, there are two problems with this simple criterion: First, if multiple users traveled through a hallway, that hallway is detected multiple times. Secondly, non-existing hallways are detected from corrupted traces. We solve these problems by iteratively building a *hallway skeleton* and matching additional traces with the existing hallway skeleton using a *hallway relation*. Traces that cannot be successfully matched with the hallway skeleton are considered to be corrupted and are subsequently excluded from further computations.



(a) Raw and corrected traces shown on the actual building layout.



(b) Hallway geometry estimated from the bounding box, interquartile range and centerIndex.



(c) Rooms detected from traces and constraints for grammar-based modelling (free/wall).

Fig. 5. Example Steps of the Trace-based Modeling component

Seeding the Hallway Skeleton We first pick a set of *seed traces* $T_{seed} \subseteq T$, where $t \in T_{seed}$ iff (1) t was rotated by less than half of the minimum angle between any two exterior walls and (2) the first hallway segment $hs_0 \in HS_t$ occurred withing *maxSteps* steps and *maxTime* from the start of t, thus ensuring that the overall drift error of each seed trace is limited. From T_{seed} we compute an initial set of hallways by grouping hallway segments according to the equivalence classes $\{[hs]\}$ of the *hallway relation*, defined as follows: $hs_i \in [hs]$ iff $\exists hs_j \in [hs]$: *hallway segment areas of* hs_i, hs_j *overlap*. The hallway segment area of hs_k is the boundig box of hs_k, extended by a width of *bbExtension* in each direction orthogonal to $hs_k.chord$. This accounts for the fact that two traces in the same hallway never overlap completely, as users might walk on different sides of the hallway. We empirically determined 0.5 m to be a suitable value for *bbExtension*.

Completing the Hallway Skeleton Next, we match traces from $T \setminus T_{seed}$ onto the initial set of hallways. A trace $t \in T \setminus T_{seed}$ matches iff (1) the first and last hallway segment of t do not add new equivalence classes and (2) at least 75 % of steps in hallways segments of t do not add new equivalence classes, i.e., t fits into the current set of hallways. New equivalence classes defined by a matching trace are added to $\{[hs]\}$. This process is repeated until no additional traces can be matched to the hallway skeleton. Note that even due to our strict criteria, corrupted traces might still be matched onto the existing hallways. To fully exclude corrupted traces, we require that each hallway is *verified* multiple times, i.e., $[hs]$ contains either (1) at least four hallway segments from traces in T_{seed} or (2) at least two hallway segments, one of which is from a trace in $T \setminus T_{seed}$. Non-verified hallways are present in $\{[hs]\}$ but not used in matching traces or the following hallways geometry estimation step until they are verified.

Hallway Geometry Estimation To find the geometrical outline of each hallway, we compute four values: Length (the dimension along the direction), width (the dimension orthogonal to the direction), the center line (location of the hallway) and the topological extension (connecting topological neighbors). The *length* is taken from the bounding box around all steps of all hallway segments. The *width* is computed as the interquartile range of step positions, i.e., 25 % of steps are not included on either side of the hallway. This is motivated by the observation that traces exhibit a normal-distributed error orthogonal to the direction of a hallway (see Fig. 5b). Furthermore, we observed that the position of hallway segments with respect to the true position of a hallway trends towards the side of entry. When users can enter a hallway from only one side, e.g., at the edge of a building, the center of the hallway is found too close to the center of the building. Thus, we compute the ratio of entrances into a hallway from either side and push the center towards the side with fewer entries. To this end, we set *centerIndex* $= (\frac{enter_{left} - enter_{right}}{2(enter_{left} + enter_{right})} + 0.5) *$ *number of steps* and choose the *center line of the hallway* so that it passes through the *centerIndex*'th step from the left. Finally, for each hallway h_i we adjust the length and width so that if a trace passes from h_i to h_j, h_i and h_j intersect (*topological extension*) and no space of depth *minRoomDepth* is left between a hallway and an exterior wall, i.e., the space to place rooms in must have sufficient size to at least open the door. The set of all rectangles computed in this step forms the *hallway skeleton*.

C. Secondary Trace Correction & Room Geometry Estimation

Using the hallway skeleton computed in the previous section, we now determine the room layout. As explained in the hallway geometry estimation step, in each hallway, traces exhibit a normal-distributed error orthogonal to the direction of the hallway. Therefore, we add a *secondary trace correction* step, where we move the hallway segments of each trace onto the center line of their corresponding hallway (see Fig. 5a), analogous to the rotation process shown in Fig. 3.

These doubly-corrected traces are used in the *room geometry estimation* step to find the room layout. We first extract room segments, i.e., maximum-length continuous sequences of steps that do not overlap with the hallway skeleton. In the following, we limit our discussion to rectangular rooms for simplicity. Support for non-rectangular rooms may be added using, e.g., clustering and alpha shapes [2]. We first detect *initial rooms* for each room segment by creating a bounding box around the room segment and placing the door at the point where the trace crosses a hallway wall as shown in Fig. 5c. As rooms may have been detected by multiple traces, we find the actual rooms as *merged rooms*, based on the door position and the geometrical center of the rooms: All rooms where (1) doors are no more than 1 m apart and (2) the distance between the center points is no more than 2 m are replaced by a single room comprised of a bounding box around the area of all individual rooms and a door placed at the mean position of all merged doors.

	R_1^{room}	R_5^{unit}
Rule	$Space \rightarrow r_1\, Space$	$Space \rightarrow r_3 r_2 r_3\, Space$
Width	2.4 m	19.2 m
A-Priori	0.06	0.04
Type	Small Office	Two executives with assistant's office

Fig. 6. Example for room rules and room unit rules

Finally, the hallway skeleton together with the detected rooms forms the *trace-based indoor model*.

V. Grammar-based Modeling

The information in the trace-based indoor model can be incomplete for areas where we do not have traces, or the map can be inaccurate due to the inaccuracies in the trace data. In the *grammar-based modeling* component, we try to improve the quality of the trace-based indoor model by filling white spots in the map and by trying to correct these inaccuracies by imposing constraints on valid indoor maps. For instance, in the trace-based model, rooms of the same type have slightly different geometries, and some walls might have been missed due to missing traces in a room. The grammar-based modeling will try to align walls such that these errors are corrected.

In the following, we first present the formal definition of our grammar that encodes structural knowledge of the building. Subsequently, we present an algorithm to derive a room layout with the help of this grammar. Note that for simplicity, we limit our discussion to rectangular rooms. The full grammar that supports arbitrarily shaped rooms is presented in [7].

A. Room Grammar

In general, the structure of a building can be separated in two different areas: Hallway spaces and non-hallway spaces. While hallway spaces are traversed by users to reach different rooms, non-hallway spaces contain the rooms which are ordered in a certain sequence along each non-hallway space. However, the size and relative ordering of these rooms is not created by random composition of walls, but follows architectural principles and semantic relationships. For instance, public buildings often feature a very limited set of room sizes. Furthermore, individual rooms may be grouped into superior room units by their semantic relationship, e.g., the office of an assistant is very likely next to the office of an executive. Thus, once an executive office has been detected from traces, we implicitly detected the neighboring assistant's office along with it.

Formal Grammar We encode such structural information for non-hallway spaces by a formal grammar for a regular language of the form $G = (N, T, P, S)$, where $N = \{Space\}$ is the set of non-terminal symbols, $T = \{\epsilon, r_a, r_b, r_c, ...\}$ is the set of terminal symbols, P is the set of production rules and $S = Space$ is the axiom. Each terminal $r_i \in T$ represents a class i of rooms, identified by their geometric extent (see Fig. 6, 10). For instance, assistant's office and executive office are two different classes of rooms that can occur more than once in one floor plan. Furthermore, we encode knowledge about room units as fixed sequences of rooms that can be produced. Therefore, we define the production rules P as follows (cf. Fig. 6):

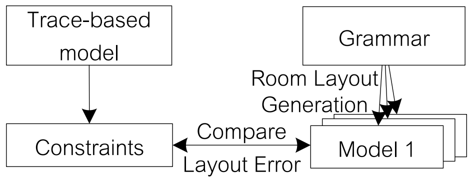


Fig. 7. Overview of the grammar-based room layout generation

```
Require: Rule Sequence $w$, Wall Constraints $WC$, Free Constraints $FC$
    $errors \leftarrow \emptyset$
 2: for all $[start, end] \in WC$ do
        if $w$ generates $wall \in [start, end]$ then
 4:         $errors \leftarrow errors \cup \{0.0\}$
        else
 6:         $x \leftarrow$ wall with minimum distance to $[start, end]$
            $errors \leftarrow errors \cup \{\textsc{minDist}(x, [start, end])\}$
 8:     end if
    end for
10: for all $[start, end] \in FC$ do
        if $w$ does not generate any $wall \in [start, end]$ then
12:         $errors \leftarrow errors \cup \{0.0\}$
        end if
14:     for $wall \in [start, end]$ generated by $w$ do
            $errors \leftarrow errors \cup \{\textsc{minDist}(wall, [start, end])\}$
16:     end for
    end for
18: if $|errors| = 0$ then return $\perp$
    end if
20: return $\sum_{e \in errors} e^2 / |errors|$
```

Fig. 8. Algorithm for computing the layout error of a rule sequence

- R_i^{room} : $Space \rightarrow r_i\, Space$ for $i \in$ room classes found in the building
- R_n^{unit} : $Space \rightarrow r_j ... r_k\, Space$ for $n \in$ room units found in the building
- R^{ϵ} : $Space \rightarrow \epsilon$

For instance, one possible sequence of rules to fill a non-hallway space with this grammar is: $Space \rightarrow r_3\, Space \rightarrow r_3 r_1 r_5$.

Probabilistic Grammar While this grammar encodes knowledge about existing room classes and room units, it does not contain knowledge about the typical neighborhood of these elements. For example, a combination of assistant's office and executive office may typically occur in combination with other office rooms, but hardly ever in combination with maintenance access rooms. Therefore, we define two probability models for the grammar: (1) The *a priori probabilities* of rules and (2) the *relationship probabilities* between rules. The a priori probability $P_a(R_i)$ encodes the relative frequency of occurrence of a room or room unit defined by rule $R_i^{\{room,unit\}}$. The relationship probability $P_{rel}(R_i|R_j)$ is a conditional probability which models neighborhood relationships between rooms or room units. For instance, $P_{rel}(R_j^{room}|R_i^{room}) = 0.5$ states that with a probability of 50%, room $r_{next} = r_j$ in any sequence $...r_i r_{next}$. To use these probabilities, we translate the grammar into a Markov chain. Room rules R_m^{room} and unit rules R_n^{unit} form the nodes of the Markov chain. The probability for a transition from R_i to R_j is defined as $\frac{P_a(R_j)}{P_a(R_i)} \cdot \frac{P_{rel}(R_i|R_j)}{P_{rel}(R_j|R_i)}$ according to [12].

Layout Error The grammar can create a multitude of

```
Require: Current Rule Sequence $w$, Set of Rules $P$, Wall Constraints $WC$,
    Free Constraints $FC$
    $P \leftarrow \{R \in P | wr$ fits in the available space$\}$          ▷ Annotate
 2: for all $R \in P$ do
        $R.RE \leftarrow$ GETERROR$(w, R, WC, FC)$
 4:     $Q \leftarrow$ last rule in $w$
        $R.prob \leftarrow \frac{P(R)}{P(Q)} \cdot \frac{P(Q|R)}{P(R|Q)}$
 6: end for
    $P_{error} \leftarrow \{R \in P | r.RE \neq \perp\}$          ▷ Filter
 8: $P_{prob} \leftarrow \{R \in P | r.RE = \perp\}$
    $minRE \leftarrow$ MIN$(\{R.RE | R \in P_{error}\})$
10: $P_{error} \leftarrow \{R \in P | R.RE \leq minRE\}$
    $P \leftarrow \{R \in P_{error} \cup P_{prob} | R.prob > 0\}$
12: if $P$ is empty then          ▷ Recover
        $P \leftarrow P_{error}$
14:     for all $R \in P$ do
            $R.prob \leftarrow 1/|P|$
16:     end for
    end if
18: normalize all $R.prob$ to 1.0          ▷ Select
    return $R$ randomly selected according to $R.prob$
```

Fig. 9. Constraint-Augmented Random Walk

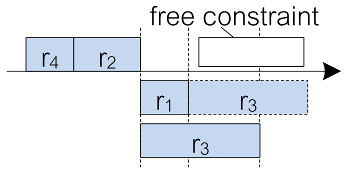


Fig. 10. Example application of rules. Combining Error- and Probability-Rules avoids unnecessary increases of the layout error.

different indoor models. To determine the model that best fits the observations from the trace-based indoor model, we derive a set of constraints on the grammar-based room layout from the trace-based room layout and define the *layout error* (LE, Fig. 8) to determine how well a given model fits the constraints (see Fig. 7). Constraints and individual errors are derived as follows: (1) For each room in the trace-based model, we introduce a *free constraint* as an area where no wall should be placed. The error is the distance of each wall placed inside the constraint to the respective nearest end of the constraint (l. 10–17). (2) For each neighboring pair of rooms, we introduce a *wall constraint*, placed in between these rooms, indicating that at least one wall should be placed in between these rooms. The error is the smallest distance from either end of the constraint to the closest room wall placed by the grammar (l. 2–9). The mean square over these individual errors is returned as the layout error (l. 20).

Finally, note that the grammar can produce room layouts that do not completely fill (or may overflow) the available area. Avoiding such layouts is the responsibility of the room layout generation algorithm presented in the next section.

B. Room Layout Generation

The goal of the room layout generation algorithm is twofold: (1) Find a room layout with minimal layout error and (2) provide a probable room layout for unobserved areas that completely fills the non-hallway space. Note that achieving a LE of 0 may not be possible: due to noisy observations, e.g., rooms detected at a short distance left or right of their actual position, it may not be possible to perfectly recreate the trace-based room layout from the grammar. Finding the minimum-error room layout from the grammar is an NP-hard problem that can be reduced to the Knapsack problem. Therefore, we employ a heuristic solution: We perform a *constraint-augmented random walk* on the Markov chain to find different room layouts. The random walk is repeated *numRandomWalk* times and the room layout that minimizes the LE is chosen. The algorithm, depicted in Fig. 9, iteratively performs the four steps of *Annotate*, *Filter*, *Recover* and *Select* as follows.

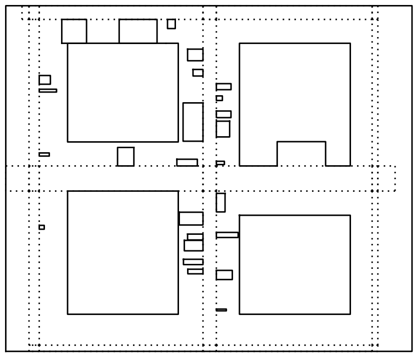
Annotate: Initially, rules are annotated with their transition probability and error they will cause. Given an (initially empty) sequence of rooms and the rules w that were used to produce these rooms, we first determine which rules can be applied without overflowing the current non-hallway space (l. 1). For the remaining rules, we compute a rule error (***R.RE***, l. 3) and set their transition probability from the Markov chain (l. 5). The RE is computed analogously to the LE, limited to the area of the non-corridor-space that is affected by R. If no constraints are encountered in the affected area, e.g., when computing a layout in a part of the building that has not been observed at all, no RE value ($\perp$) is set for R (cf. Fig. 8).

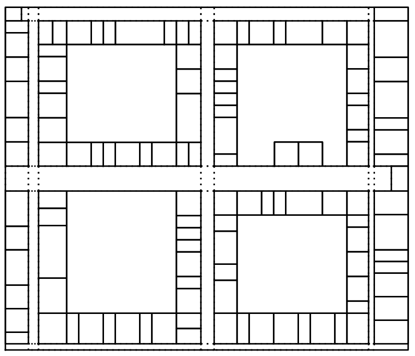
Filter: Next, rules are filtered for their applicability, i.e., whether they have a non-zero transition probability and minimize the resulting error. To this end, we split rules into two disjoint subsets: Error-Rules P_{error} (rules that have an RE value set, l. 7) and Probability-Rules P_{prob} (all remaining rules, l. 8). Error-Rules are filtered, keeping only the rules with the minimal RE value (l. 10). Note that we cannot limit the selection to either Error- or Probability-Rules, as illustrated by the example depicted in Fig. 10. $r_1 \in P_{prob}$ as r_1 does not touch the free constraint, whereas $r_3 \in P_{error}$. For instance, if we were to focus only on P_{error}, we lose the chance to select the room sequence $r_1 r_3$ which would improve the LE over selecting r_3. Furthermore, the grammar would be biased towards choosing rules that place more and larger rooms, as all rules in P_{error} occupy more space than any rule in P_{prob}. The reverse is also true if we were to focus only on P_{prob}. Therefore, both sets are rejoined, removing all rules with a transition probability of 0 in the Markov chain, i.e., rules that cannot occur in this neighborhood (l. 11).

Recover: As the grammar may be incomplete, it may not account for all combinations of rooms that occur in the building. Thus, we could encounter a situation where none of the remaining rules in either set has a transition probability > 0 in the Markov chain, effectively deadlocking our algorithm. In this case, we override the Markov chain by using the set of Error-Rules with minimal RE value and temporarily adjusting their transition probabilities to a uniform value (l. 12–17).

Select: Finally, we choose a rule out of the remaining ones at random according to the (adjusted) transition probabilities (l. 19). When no space is left in the current non-corridor space to use any additional rule, R^{ϵ} is applied and the resulting sequence of rooms is returned as the room layout for this non-corridor-space.

The hallway skeleton from the trace-based indoor model together with the set of grammar-based room layouts for all non-corridor-spaces forms the *grammar-based indoor model*, which is the final output of our system.


(a) Trace-based Indoor Model


(b) Grammar-based Indoor Model

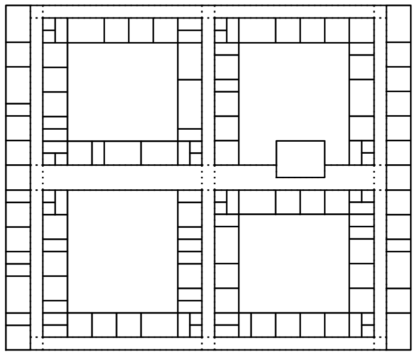
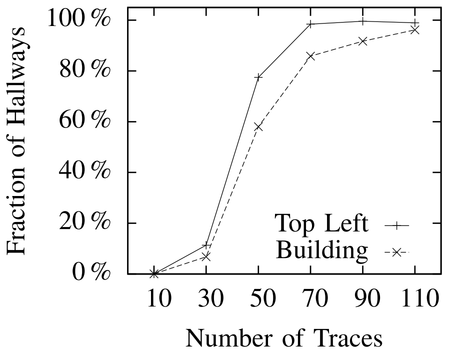
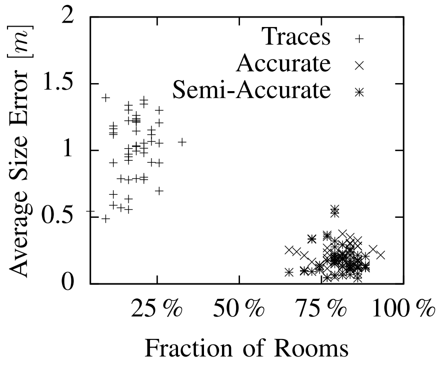

(c) Ground-Truth Indoor Model

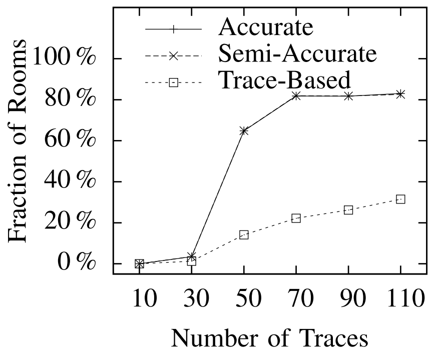
Fig. 12. Dense Scenario: Indoor-Models derived using all recorded traces



(a) Average fraction of detected hallways over the number of traces used.



(b) Fraction of matched rooms vs. average error in room size. Values for 50 traces.



(c) Average fraction of matched rooms over the number of traces used.

Fig. 13. Sparse Scenario: Quality result of models derived from a varying number of traces

(a) Android App

(b) Foot-mounted IMU

Fig. 11. Utilities used for recording the pedestrian traces: A foot-mounted IMU that sends relative position changes to an Android App which merges these steps to a trace and shows it on a map.

System Parameter	Value	System Parameter	Value
minLength	4 m	bbExtension	0.5 m
maxSteps	1	minRoomDepth	2 m
maxTime	1 s	numRandomWalk	10

Fig. 14. Parameters used in the evaluation

VI. Evaluation

To demonstrate the effectiveness of MapGENIEs' grammar-based approach, we tested our system in a real-world scenario. We first describe the setup of our evaluation, before we discuss evaluation results for *sparse* and *dense scenarios*, showing the performance of our system during trace acquisition and when a full set of trace data for the building is available.

A. System Setup

To set up the inertial positioning system, we designed an Android App, communicating with a foot-mounted inertial measurement unit via Bluetooth (see Fig. 11). We used a Zero-Velocity-Update protocol [9] to limit the sensor drift of the foot-mounted unit. All other components are implemented as a backend-service on a Linux server (cf. Section II). The parameters used in our evaluation are shown in Fig. 14.

To test our implementation, we collected 250 odometry traces, measuring a total of over 22 km from four volunteers. Traces are collected from the 2nd floor of the computer science building in Stuttgart (see Fig. 12c). Volunteers could freely walk all hallways in the building. Room access was limited to only a subset of office rooms, mainly located in the top left quadrant. Thus, hallways in this quadrant are well traveled, whereas only very few traces passed through hallways at the right and lower edges of the building. Furthermore, users did not visit maintenance rooms, which are not publicly accessible.

We evaluate our system in two scenarios. In the *dense* *trace scenario*, we show the output of the system when using all available information, i.e., we derive the indoor-model using all traces and using an accurate grammar for the floor. This demonstrates how even a well-mapped area can benefit from using a grammar-supported approach. In the *sparse trace scenario*, we analyze the performance of the system with incomplete information, i.e., using only a fraction of the traces, showing how the grammar-based approach is used to enhance the incomplete information of these traces. Furthermore, in the sparse scenario we use two grammars for the room layout generation: An *accurate grammar* derived from the floor plan of the 2nd floor and a *semi-accurate grammar* derived from the 1st floor of the same building, which, while overall similar to the 2nd floor, features different room types, room units and neighborhood relationships. Neither grammar includes knowledge of maintenance access rooms, which we cannot find without having an accurate floor plan in the first place. The semi-accurate grammar illustrates the performance of our system when using data obtained from a similar building. Note that using a fully inaccurate grammar will lead to fully inaccurate floor plans.

B. Dense Trace Scenario

Figure 12a shows the trace-based indoor model from the dense trace scenario, i.e., using all traces. Hallways, on the one hand, were estimated almost perfectly in areas covered by a large number of traces, i.e., the top-left quadrant and the center hallways. The rightmost and bottommost hallways are derived from only four traces each, which do not contain a sufficient number of samples to accurately find their width and position. The room layout, on the other hand, is very poor. Out of a total of 74 office rooms in the building, only 26 were found. Rooms are only of the right size if they are either very narrow or were detected from multiple (in our case three) traces.

Next, we look at the grammar-based indoor model, depicted in Fig. 12b. When comparing the grammar-generated room layout to the trace-based indoor model and the ground truth, we see that in areas where no rooms were found, the grammar provides a plausible but not necessarily perfectly accurate layout of rooms. In areas with observations, the observed rooms are reproduced by the grammar. Note that while the grammar-based layout reduces the error of the room layout, it is faithful to the observations. For instance, when a single room is detected as multiple rooms due to an overall insufficient number of traces entering the room, the incorrect room is recreated by the grammar. Overall, the basic structure of the building is reflected in the model.

C. Sparse Trace Scenario

In the sparse trace scenario, we analyze the performance of our system when operating on a limited set of traces only. For varying numbers of traces n, we repeatedly selected random subsets of n traces out of the set of all traces. For each set of traces we construct the trace-based indoor model and the grammar-based indoor model.

We first compare the hallway skeleton from the computed indoor model to the ground truth hallway skeleton. Figure 13a shows the average number of hallways detected for each number of traces. Using 30 traces, we can find 12 % of hallways in the building (“Building”). Finding more than 80 % of hallways requires at least 70 traces. This is due to our strict verification criteria. We require a number of high-quality seed traces, which are usually not present in a smaller number of traces. Loosening these parameters leads to finding more hallways. However, it also leads to false positives, i.e., finding non-existent hallways, which we avoid with our strict criteria. When limiting the analysis to the top left quadrant (“Top Left”), results show a similar trend but are overall better, since traces concentrate in that area, i.e., we are more likely to find hallways here than in any other part of the building.

Next, we analyze the accuracy of the room layout generation. For each computed indoor model, we match detected rooms to rooms from the ground truth model. Matching rooms lie on the same side of the same hallway and have similar size. Furthermore, we respect the ordering of rooms, i.e., if room b is located to the right of room a in the computed model and a was matched to x from the ground truth, then b cannot be matched to a room left of x. From here on we limit our analysis to the top left quadrant, where room information from traces is available. Note that maintenance access rooms cannot be detected from traces nor the grammar-based room layout, which account for about 19 % of rooms in the top-left quadrant.

Figure 13b shows the average error in the size of matched rooms compared to the fraction of rooms that have been found on detected hallways for all indoor models built from 50 traces. The trace-based indoor model (“Traces”) finds only very few rooms (up to 35 %) and shows a large error in the room size (0.48 m to 1,5 m), whereas using the accurate grammar (“Accurate”), many more rooms (at least 72 %) with a much lower size error (at most 0.37 m) are found. Furthermore, the semi-accurate grammar (“Semi-Accurate”) finds the same number of rooms, with a slightly higher average size error. Thus, using grammar support greatly increases the accuracy of the floor plan even when the grammar is not perfect.

To analyze the impact of the grammar further, we look at the overall fraction of matched rooms. Figure 13c shows the average number of matched rooms for each sample size of traces. Using 110 traces, we find only 31 % of rooms from traces alone, while using either grammar we detect 81 % of rooms, i.e., all rooms that can be found, with only 70 traces. Overall, we find up to 4 times as many matching rooms using the grammar-based model. Note the steep increase in the number of matching rooms from 30 to 50 traces. As argued before, more traces lead to detecting more hallways. For the trace-based model, the number of detected rooms grows with the number of traces. For the grammar-based model, the number of detected hallways is the important factor, as on each hallway, multiple matching rooms are detected. With only a few hints in existing rooms from the trace-based model, the grammar provides a far more accurate indoor model.

VII. Related Work

Most approaches for indoor mapping build on different flavours of the Simultaneous Localization and Mapping (SLAM) principle, originating from robotics. Recently, also human-operated SLAM approaches have emerged which employ miniature laser scanners [13], or the Microsoft Kinect [14]. Instead of using scanning techniques that require manual effort, the idea of using position traces, which may be acquired using low-cost ubiquitously available hardware, was proposed. Systems employing this data as the sole base for the reconstruction of building interiors are described in works like CrowdInside [2], FootSLAM [3] and SmartSLAM [15]. FootSLAM—as in our approach, using a foot-mounted IMU—reconstructs the building's interior as a map of walkable areas, not distinguishing between hallways or rooms. Instead of being constrained to a dedicated external IMU, CrowdInside and SmartSLAM employ the set of sensors found in modern smartphones. However, SmartSLAM merely reconstructs hallway structures from the resulting data. CrowdInside, on the one hand, requires a large amount of traces following the room walls in order to function well and, on the other hand, reconstructs the rooms as alpha shapes, not taking knowledge about common features of interior architecture like parallelism, rectangularity, or repetition into account. Following the evaluation, CrowdInside needs 290 trace segments to fully reconstruct all corridor areas and the 12 rooms in the floor plan used in the testbed. The system presented in [16] uses a combination of smartphone sensors and WiFi fingerprints to learn hallway layouts as well as to distinguish different rectangular rooms. The authors state that the system needs only 20 data points per floor plan to converge, while delivering an average room position accuracy of 91%, but a room area estimation error of 33% and a room aspect ratio error of 24%. The exact reconstruction of, e.g., repetitive structures is also not tackled by this approach.

Due to parallelism and rectangularity being the most prominent rules used in man-made construction, many reconstruction approaches build on the Manhattan World constraints [17]. To further include, e.g., repetition or to model more general constraints, formal grammars have been applied successfully to the modeling of geometric structures for several years. While [18] focuses on line structures by, e.g., simulating growth processes of plants through Lindenmayer-systems (L-systems), [5] and [6] proved the usability of grammars for the reconstruction of street networks and building shells. In terms of procedural modeling of building interiors, the authors of [19] present an appropriate split grammar, however, without the possibility of its use in the reconstruction from erroneous observation data.

VIII. Conclusion & Future Work

In this paper, we presented the MapGENIE approach that automatically infers indoor maps from pedestrian traces and structural building information. First, we presented a trace-based modeling approach, which is then improved by structural knowledge encoded in a formal indoor grammar. To show its performance, we conducted a large-scale experiment by collecting pedestrian indoor traces. The results show that MapGENIE enhances the indoor mapping process significantly, i.e., producing detailed indoor maps from only a small set of traces.

So far, we require an input grammar to derive a grammar-based indoor model. In the future, we will analyze automatic derivation and improvement of the grammar by using trace information. Eventually, we aim at a closed-loop system where the traces feed the grammar derivation while the knowledge stored in the grammar supports the acquisition of the traces. Furthermore, we aim at improving the hallway derivation by a hallway grammar. This will speed up the hallway creation step and improve the accuracy of the derived hallway skeleton.

Acknowledgment

The authors would like to thank Naresh Nayak for his help in the implementation.

This work is supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG), grants FR 823/25-1 and RO 1086/17-1, and the Ministry Of Science, Research and the Arts Baden-Württemberg.

References

[1] A. Moses. (2012, Nov.) 'Indoor GPS': Every step you take, every move you make, Google's got maps for you. Accessed: 07.01.2014. [Online]. Available: http://www.theage.com.au/digital-life/smartphone-apps/indoor-gps-every-step-you-take-every-move-you-make-googles-got-\maps-for-you-20121115-29e1b.html

[2] A. Moustafa and Y. Moustafa, "Crowdinside: Automatic construction of indoor floorplans," in *Proc. Conf. Advances in Geographic Information Systems (SIGSPATIAL)*, 2012.

[3] M. Angermann and P. Robertson, "Footslam: Pedestrian simultaneous localization and mapping without exteroceptive sensors – hitchhiking on human perception and cognition," *Proc. of the IEEE*, vol. 100, pp. 1840–1848, 2012.

[4] A. R. Jiménez, F. Seco, F. Zampella, J. C. Prieto, and J. Guevara, "Improved heuristic drift elimination with magnetically-aided dominant directions (MiHDE) for pedestrian navigation in complex buildings," *J. Location Based Services*, vol. 6, pp. 186–210, 2012.

[5] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool, "Procedural modeling of buildings," *ACM Trans. Graph.*, vol. 25, pp. 614–623, 2006.

[6] Y. I. H. Parish and P. Müller, "Procedural modeling of cities," in *Proc. 28th Annu. Conf. Comp. Graph. and Interactive Techniques*, ser. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 301–308.

[7] S. Becker, M. Peter, D. Fritsch, D. Philipp, P. Baier, and C. Dibak, "Combined grammar for the modeling of building interiors," in *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, 2013.

[8] S. Becker and N. Haala, "Grammar supported facade reconstruction from mobile LiDAR mapping," in *Proc. WS Object Extraction for 3D City Models, Road Databases and Traffic Monitoring*, 2009.

[9] E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *IEEE Comp. Graph. and Applicat.*, vol. 25, pp. 38–46, 2005.

[10] M. B. Kjærgaard, H. Blunck, T. Godsk, T. Toftkjær, D. L. Christensen, and K. Grønbæk, "Indoor positioning using gps revisited," in *Proc. Conf. on Pervasive Computing (Pervasive)*, 2010.

[11] O. Woodman, "An introduction to inertial navigation," University of Cambridge, Technical Report 696, 2007.

[12] J. O. Talton, Y. Lou, S. Lesser, J. Duke, R. Mech, and V. Koltun, "Metropolis procedural modeling," *ACM Trans. Graph.*, vol. 30, pp. 11:1–11:14, 2011.

[13] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping," *IEEE Trans. Robotics*, vol. 28, pp. 1104–1119, 2012.

[14] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: real-time dense surface mapping and tracking," in *Proc. Symp. Mixed and Augmented Reality (ISMAR)*, 2011.

[15] H. Shin, Y. Chon, and H. Cha, "Unsupervised construction of an indoor floor plan using a smartphone," *IEEE Trans. Syst., Man, and Cybernetics, C: Applicat. and Reviews*, vol. 42, pp. 889–898, 2012.

[16] Y. Jiang, Y. Xiang, X. Pan, K. Li, Q. Lv, R. P. Dick, L. Shang, and M. Hannigan, "Hallway based automatic indoor floorplan construction using room fingerprints," in *Proc. Joint Conf. Pervasive and Ubiquitous Computing (UbiComp)*, 2013.

[17] J. M. Coughlan and A. L. Yuille, "Manhattan world: Compass direction from a single image by bayesian inference," in *Proc. 7th IEEE Int. Conf. Comp. Vision*, vol. 2, 1999, pp. 941–947.

[18] P. Prusinkiewicz and A. Lindenmayer, *The algorithmic beauty of plants*. Springer New York, 1990.

[19] G. Gröger and L. Plümer, "Derivation of 3D indoor models by grammars for route planning," *Photogrammetrie-Fernerkundung-Geoinformation*, vol. 2010, pp. 193–210, 2010.