# ComNSense: Grammar-Driven Crowd-Sourcing of Point Clouds for Automatic Indoor Mapping

MOHAMED ABDELAAL, DANIEL REICHELT, FRANK DÜRR, KURT ROTHERMEL, Institute of Parallel and Distributed Systems, University of Stuttgart. Stuttgart, Germany
LAVINIA RUNCEANU, SUSANNE BECKER, DIETER FRITSCH, Institute of Photogrammetry, University of Stuttgart. Stuttgart, Germany

Recently, point clouds have been efficiently utilized for medical imaging, modeling urban environments, and indoor modeling. In this realm, several mobile platforms, such as Google Tango and Apple ARKit, have been released leveraging 3D mapping, augmented reality, etc. In modeling applications, these modern mobile devices opened the door for crowd-sourcing point clouds to distribute the overhead of data collection. However, uploading these large points clouds from resources-constrained mobile devices to the back-end servers consumes excessive energy. Accordingly, participation rates in such crowd-sensing systems can be negatively influenced. To tackle this challenge, this paper introduces our ComNSense approach that dramatically reduces the energy consumption of processing and uploading point clouds. To this end, ComNSense reports only a set of extracted geometrical data to the servers. To optimize the geometry extraction, ComNSense leverages formal grammars which encode design-time knowledge, i.e. structural information. To demonstrate the effectiveness of ComNSense, we performed several experiments of collecting point clouds from two different buildings to extract the walls location, as a case study. We also assess the performance of ComNSense relative to a grammar-free method. The results showed a significant reduction of the energy consumption while achieving a comparable detection accuracy.

CCS Concepts: • **Human-centered computing** → **Mobile devices**; *Ubiquitous and mobile computing systems and tools*; • **Networks** → *Cyber-physical networks*; • **Theory of computation** → Data compression;

Additional Key Words and Phrases: Data Acquisition, Crowd-Sensing, Point Clouds, Formal Grammars, Energy Efficiency

**ACM Reference Format:**
Mohamed Abdelaal, Daniel Reichelt, Frank Dürr, Kurt Rothermel and Lavinia Runceanu, Susanne Becker, Dieter Fritsch. 2018. ComNSense: Grammar-Driven Crowd-Sourcing of Point Clouds for Automatic Indoor Mapping. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 1 (March 2018), 26 pages. https://doi.org/10.1145/3191733

## 1 INTRODUCTION

In the recent decade, several researchers investigate the derivation of 2D and 3D indoor maps via sampling 3D data structures representing points in space, referred to as *point clouds* [2, 21, 24]. In a 3D point cloud, the points usually represent the X, Y, and Z geometric coordinates of an underlying sampled surface. In general, point clouds can be directly acquired from hardware sensors such as stereo cameras, 3D scanners, or time-of-flight cameras; or can be synthetically generated from fusing manifold 2D images [28]. In the aforementioned endeavors of deriving indoor maps from point clouds, the data was collected using laser scanners, which certainly increases the

Authors' addresses: Mohamed Abdelaal, Daniel Reichelt, Frank Dürr, Kurt Rothermel, Institute of Parallel and Distributed Systems, University of Stuttgart. Stuttgart, Germany, first.last@ipvs.uni-stuttgart.de; Lavinia Runceanu, Susanne Becker, Dieter Fritsch, Institute of Photogrammetry, University of Stuttgart. Stuttgart, Germany, first.last@ifp.uni-stuttgart.de.


000000-00000/2018/0-ART1 $15.00
https://doi.org/10.0000000/0000000

overhead and time of data collection. An innovative solution for this issue, namely *crowd-sensing*, is to harness the data that is already available, coming from our smart mobile devices. These mobile devices become a meaningful source of information supported by powerful sensors and increasing computational capabilities. Crowd-sensing provides an opportunity to involve many users ("the crowd") in collecting sensor data. For instance, Gao et al. introduce Jigsaw, an indoor mapping system in which a point cloud is generated from a bundle of 2D images [10]. A similar approach for crowd-sensing images to generate point clouds is introduced in [8]. However, it is worth to mention here that generating 3D point clouds from 2D images is broadly a cumbersome task owing to the imperative acquisition of large datasets. For instance, Jigsaw collects, on average, 200 images for each landmark so that the generated point clouds are reliable enough. Hence, the overhead in terms of energy and time of data collection cannot be ignored.

Recently, several modern mobile devices, such as Google Tango and Apple ARKit have been released enabling the direct acquisition of 3D point clouds. Throughout this paper, we focus on Tango devices while developing the proposed concepts. However, it is important to mention that these concepts are platform-agnostic, and can be easily applied to other platforms. In general, Tango is a sensor fusion technology which enables the mobile devices to track their 3D motion and to create a 3D representation of the surroundings [4]. Thanks to these capabilities, Tango devices serve as a promising candidate for acquiring point clouds in indoor mapping applications. However, running such applications demands a significant amount of resources of the mobile devices, e.g. energy budget and CPU power, to provide reliable indoor maps. Furthermore, the energy overhead of acquiring point clouds may dramatically increase given that the point clouds scanned by the crowd are typically incomplete. Hence several scans of a scene may be necessary. Accordingly, the challenge of crowd-sensing point clouds in an energy-efficient manner emerges.

In this paper, we tackle this challenge through introducing a novel data acquisition approach, referred to as ComNSense. The core idea behind ComNSense is to directly process the point clouds on the mobile devices to extract the required geometrical information. Hence, the mobile devices solely report these *intermediate* data to a crowd-sensing server which aggregates the required data from different users and then derives the indoor model. To the best of our knowledge, there exist no prior efforts for optimizing the point cloud acquisition from mobile devices. Our approach significantly reduces the uploading energy consumption due to shrinking the size of transmitted data. However, another challenge of efficiently processing the point clouds on the mobile devices without negatively influencing the energy budget arises. For instance, we ran the region-growing algorithm [29] to divide a point cloud—generated through scanning a medium-sized room—into a bundle of planar surfaces on our mobile device. The mobile device took circa seven hours to cluster the point cloud. Consequently, solving this problem entails sidestepping the adoption of traditional point clouds processing algorithms, such as region-growing.

To this end, ComNSense leverages the design-time knowledge to reduce the search space while estimating the required geometrical information, e.g. wall locations. Specifically, we exploit so-called *indoor grammars* to search solely a subset of the collected point clouds, in which the walls exist with high probability. Indoor grammars are powerful tools for encoding structural information for different kinds of architectural domains. Considering public buildings, we found that rooms usually have specific dimensions and there exist semantic relationships between rooms, e.g. an assistant's office mostly lies next to an executive's office. If the floor plan of a building has these features, then the roles of indoor grammar in ComNSense is twofold: (1) It can be implemented on the mobile devices to efficiently extract the required geometrical information from point clouds. To this end, we match between a set of grammar-generated indoor models, referred to as *grammar hypotheses*, and the collected point clouds. These hypotheses are generated through randomly walking on a Markov chain. (2) Indoor grammars can be used at the server to fit the collected geometrical information while deriving the indoor map. It is worth to mention that we are the first to employ indoor grammars to efficiently process point clouds on the resources-constrained mobile devices.

In detail, the paper provides the following contributions: (1) We propose a novel grammar-based point cloud acquisition method. The proposed method considers reducing the energy overhead of processing and reporting point clouds. (2) For the sake of proper matching between a hypothesis and a point cloud, we adapt the hypothesis generation to the sensed data. To this end, we present a simple yet powerful prediction-based segmentation algorithm that classifies the motion traces into room and hallway segments. In this regard, we make the clustering decision in accordance with the occurrence of adequate changes in the motion direction as well as the motion speed. (3) We introduce a Bayesian inference method through which the probable locations of the scanned geometries can be estimated. The prior probabilities are computed from the relative frequency of rooms and their transition probabilities, whereas the motion traces are exploited to derive the likelihood probabilities. (4) We introduce an algorithm to derive grammar hypotheses while considering the motion traces and the Bayesian posterior probabilities. The intuition is to control the hypothesis generation within those areas scanned by the users. (5) We present a wall segments extraction method on the mobile device as a case study of geometrical information extraction. In this realm, we employ the *principal component analysis* (PCA) [14] to precisely detect walls at the locations specified by the grammar hypothesis. (6) We present a proof-of-concept implementation and evaluation in real-world scenarios. To this end, we implemented an Android App to simultaneously collect point clouds and motion traces from two different buildings. This App improves the collected data accuracy via integrating the main three features of Tango technology, namely *motion tracking*, *depth perception*, and *area learning* [4]. We provide a comparative study between our ComNSense approach and a compression-based data acquisition method using 3 GBytes of collected data. In terms of energy consumption, ComNSense is found to consume at least 56% less energy than the baseline method. Moreover, we can identify a comparable detection accuracy of the geometrical semantics in both methods.

The remainder of this paper is organized as follows: Section 2 defines some preliminaries, including the traditional processing steps of point clouds and the formal grammars. Afterward, we introduce the system architecture and our assumptions. Then, Section 4 analyzes the motion traces and introduces the prediction-based segmentation and classification method. Subsequently, Section 5 discusses the Bayesian-based rules inference method and the grammar hypotheses generation. In Section 6, we explain the registration of point clouds and the PCA-based walls extraction method. In Section 7, we present our real-world evaluation, including data collection and comparison between the ComNSense approach (grammar-based) and the grammar-free approach. Finally, Section 8 presents the related work, before Section 9 concludes the paper with an outlook on future work.

## 2 PRELIMINARIES

In this section, we provide an overview of a grammar-free processing pipeline for sampling and processing point clouds to extract the geometrical information. Such a processing pipeline forms a baseline method while evaluating the performance of ComNSense. Since it represents the backbone of our ComNSense approach, we elaborate on indoor grammars and their roles in data acquisition as well as map construction.

### 2.1 Grammar-Free Processing Pipeline

There exist several methods for extracting geometrical objects from point clouds [32]. The method presented in this section is inspired by the work done in [24]. Figure 1 depicts the processing steps for extracting walls. To obtain other 3D objects, such as windows, doors, and furniture, the following steps have to be refined. The core idea is to upload from the mobile devices to the server a compressed version of the point clouds. Subsequently, we extract the required geometrical information on the server side. To this end, we utilize a well-known compression method, called *Octree-based compression* [31]. An Octree is a tree-based data structure for managing 3D data where each internal node has exactly eight children. This compression method has three primary steps, including: (1) Spatial decomposition of the point cloud into an Octree data structure. (2) Quantization of the point cloud

by replacing the points by the cell centers of the Octree's leaves. (3) Arithmetic coding of the remaining points through considering only cells whose child cells are nonempty.

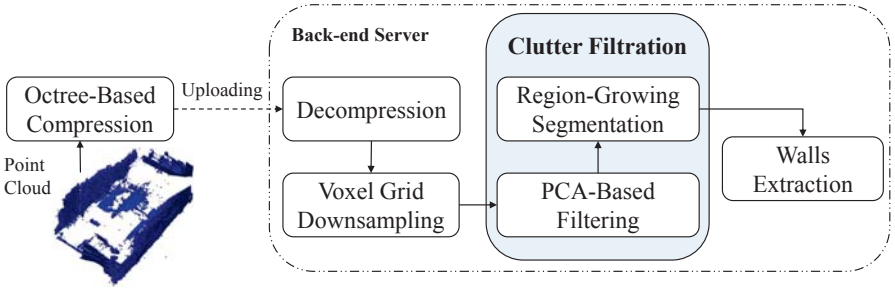



Fig. 1. A grammar-free method for extracting walls from point clouds

After uploading the compressed point cloud to the server and decompressing it again on the server-side, the server performs several filtration steps, leaving only the possible wall segments. We start with a voxel grid to downsample the point clouds, before we filter the clutter through normals-based filtration and region-growing segmentation [29]. Literally, clutter denotes the points which do not belong to any existing planar surface. The *principal components analysis* method [14] is utilized to estimate the surface *normals*, i.e. eigenvectors describing the surface orientation. More details about normals estimation are given in Section 6.2. Region-growing is used to identify the planar segments so that we can remove clutter present in the corners between these segments. Each segment is a set of points that are considered to be a part of the same smooth surface, i.e. curvature-based clustering. Afterward, we perform walls extraction where we differentiate between the wall segments and the reference line segments. The latter segments represent those lines orthogonal to the wall segments connecting several rooms together. Our criteria, in this regard, are the height of the segments as well as the angle of the surface normal. Below, we introduce the formal grammars and their application to model the building's structural information.

## 2.2 Indoor Grammars

In most cases, the structure of a building's floor can be modeled as a composition of two areas, including *hallway* spaces and *non-hallway* spaces. The former are typically areas traversed by individuals to reach various rooms, whereas the non-hallway spaces comprise numerous cascaded rooms which have different sizes. It is important to realize that the size and relative ordering of these rooms follows architectural principles and semantic relationships. For example, public buildings often feature a very limited set of room sizes. Moreover, individual rooms can be categorized according to their semantic relationship. This grammar knowledge simplifies modeling the non-hallway areas where detecting an executive office implicitly means detecting the neighboring assistant's office. Practically speaking, interior grammars cover several hierarchical inference levels. For instance, the L-System grammar is well-suited for detecting hallway areas. The split grammar can be applied to identify the layout of rooms, the room types, and the furniture [34]. A split grammar generally comprises split rules to divide an initially simple shape into more complex ones.

In [5], we presented an iterative search method to instantiate grammar rules from observation data forming a generic grammar template. This grammar template can be flexibly adopted to individual buildings based on raw point clouds to generate several hypotheses about possible indoor geometries. This implies that generating a grammar of a certain building solely requires a small set of motion traces to instantiate the grammar rules for identifying the hallway and non-hallway areas. More details about the grammar instantiation from the

template can be found in [5]. Formally, the split grammar can be defined as a quadruple $G = (V_N, V_T, S, R_i)$, where $V_N = Space$ is the set of non-terminal symbols, $V_T = \{\epsilon, r_a, r_b, r_c, \cdots\}$ is the set of terminal symbols, $R_i$ is the set of production rules and $S = Space$ is the axiom. The axiom represents an empty non-hallway space which is to be further divided into rooms. The terminals $V_T$ describe solids that are not divisible any further. Both non-terminals and terminals have attributes such as the space's geometric extent and probability. The production rules $R_i$ are defined as replacement rules that perform a split to divide a *Space* into two *Space* elements along a partitioned plane. It is sometimes beneficial to specify that a symbol can be replaced by nothing at all. To describe this case, the *null* symbol $\epsilon$ is utilized. Hence, the production rules $R_i$ can be formulated as follows.

(1) $R_i^{room}$: $Space \rightarrow r_i\ Space$
(2) $R_n^{unit}$: $Space \rightarrow r_j \cdots r_k\ Space$ where $n \in$ room units in the building
(3) $R^{\epsilon}$: $Space \rightarrow \epsilon$

## 3 SYSTEM MODEL

In this section, we explicitly define the research problem along with explaining the system model and the processing architecture. Specifically, the system consists of $N$ personal mobile devices which are capable of generating point clouds, such as Tango devices. These participating devices acquire a set of point clouds $\mathcal{P} = P_1, \cdots, P_i, \cdots, P_N$, given that $P_i = p_1, \cdots, p_j, \cdots, p_M$ where $M$ is a variable denoting the number of points in a point cloud. Each point $p_j^i \in P_i$ is characterized by a 3D position as $p_j^i = [x_j^i, y_j^i, z_j^i]^T$, ignoring the color components. Point clouds are collected in a participatory crowd-sensing fashion, i.e., users are involved in the sensing task to scan the surroundings. In fact, reporting the point clouds $\mathcal{P}$ to the crowd-sensing servers represents a major source of the energy overhead on the participating mobile devices. To reduce such an overhead, the sampled points $p_j^i$ are *locally* processed on their originating devices to find the geometrical information. Notwithstanding, executing search and filtering tasks for point clouds on the mobile devices is also energy-expensive. Let $\Omega(\mathcal{P})$ denote the accuracy of mapping rooms given a set of point clouds $\mathcal{P}$. Similarly, assume $\Omega(\mathcal{M})$ represents the mapping accuracy in case the geometrical information $\mathcal{M}$ is extracted on the mobile devices and then is uploaded. Equation 1 expresses our objective in minimizing the energy overhead of online processing and uploading point clouds to the corresponding servers where $\mathcal{Q} = q_1, q_2, \cdots$ is the set of sensing queries distributed by the crowd-sensing servers. This objective of uploading less data is governed by achieving a comparable modeling accuracy where $|\mathcal{M}_{\mathcal{P}}|$ is the size of the extracted geometrical information and $|\mathcal{P}|$ denotes the size of the corresponding point clouds.

$$\begin{aligned} &\text{minimize} \quad E^{(q)}_{processing} + E^{(q)}_{upload}, \quad q \in \mathcal{Q} \\ &\text{subject to} \quad \Omega(\mathcal{P}) = \Omega(\mathcal{M}_{\mathcal{P}}), \quad |\mathcal{M}_{\mathcal{P}}| << |\mathcal{P}|. \end{aligned} \tag{1}$$

To achieve these objectives, we propose to harness indoor grammars while *locally* processing the point clouds. Our approach relies on generating a grammar hypothesis for each mobile user to optimize the point cloud processing at his/her mobile device. The intuition is to match the collected point cloud with a grammar hypothesis to identify the wall locations. This requirement means that properly inserting specific rules at any location in the floor has to be feasible. Therefore, our grammar derivation utilizes random as well as constrained walks on a Markov chain to derive the grammar hypotheses. To this end, we detect the reference line segment at which the point clouds have been scanned. Subsequently, we adjust the transition probabilities to ensure the insertion of specific rules. More details about the proposed algorithm is given in Section 5. Additionally, we utilize the grammar at the server side for fitting the geometrical information collected from the participating mobile devices. To this end, we turn the extracted wall locations into constraints which have to be fulfilled while generating a grammar hypothesis. In this case, the hypothesis reflects the actual indoor model of the building.

Figure 2 depicts the proposed processing pipeline for automatically generating indoor maps while optimizing point cloud processing on the mobile devices. The dashed numbered lines denote a sequence of data exchange between the mobile devices and the crowd-sensing server. Specifically, the optimized processing occurs through matching the point clouds with grammar hypotheses. Hence, the search space for finding locations of the geometrical information can be dramatically reduced. To this end, the mobile devices upload their motion traces—recorded while capturing the point clouds—to the servers (labeled 2). Afterward, the server customizes the hypotheses generation in accordance with the motion traces such that inappropriate rules are entirely averted at the detected user location. Below, we briefly introduce the various components of our system architecture.

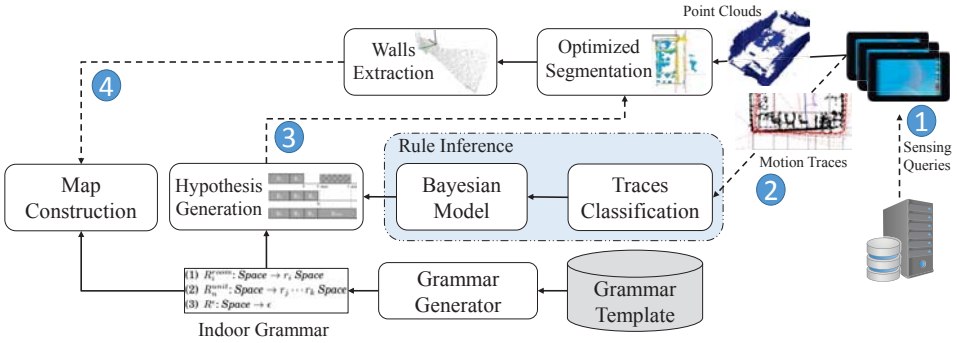



Fig. 2. System architecture

The *rule inference* component receives a set of motion traces $\mathcal{T} = T_1, \cdots, T_i, \cdots, T_N$. Each motion trace $T_i$ comprises the mobile user's position and orientation over time. To derive a reliable grammar hypothesis, these motion traces are utilized for: (1) updating the prior probabilities of a Bayesian method and differentiating between grammar rules whose attributes are similar, (2) finding the reference line segments on which the scanned area lies, and finally (3) determining the index at which the Bayesian-inferred rules have to be inserted on the reference line segment. Before collecting the motion traces, we collect—in an offline session—a set of *area description files* (ADFs) to rectify the localization drifts inherent in the motion traces and to avoid the case in which the scanned point clouds overlap each other. In our system settings, mobile users move freely without any constraints. Since we are merely interested in the room segments, we process the collected motion traces to identify the room and hallway segments. The *hypothesis generation* component is responsible for generating a grammar hypothesis such that the inferred rule is inserted at the actual user location.

The *optimized segmentation* component aims at reducing the processing overhead of point clouds on the mobile devices. It receives a grammar hypothesis from the corresponding server to match with the collected point clouds. In particular, the output of this component is a subset $S_{P_i}$ of the input point cloud $P_i$, where the walls lie in this subset with high probability. Subsequently, the *walls extraction* component employs a PCA-based method to detect the wall inside the subset $S_{P_i}$. Finally, the *map construction* component is performed at the server via performing constraint-restricted random walk on a Markov chain whose nodes represent the grammar rules. In this regard, the extracted wall segments define constraints of the transitions between the Markov nodes. In the sequel, we explain each of the aforementioned steps required to optimize the energy costs of processing and uploading point clouds in more detail.

## 4 MOTION ANALYSIS

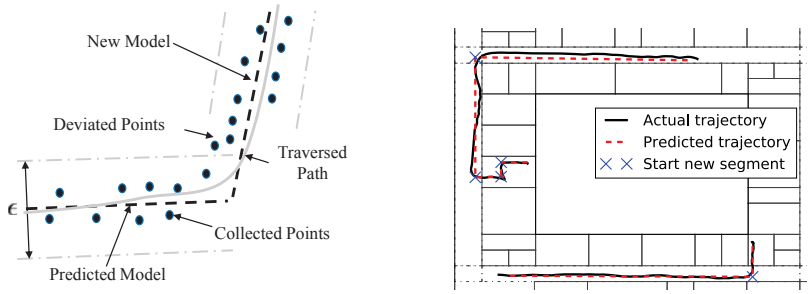
In ComNSense, the motion traces are not utilized to accurately identify the geometrical features of the hallways and the rooms. Instead, we exploit the motion traces to provide an insight about the width of the rooms while generating the grammar rules from the template (cf. Section 2.2). To this end, we first have to identify the room segments and filter out the hallway segments. Afterward, we extract the footprint of the user inside the room from the motion traces. To cope up with the noisy nature of the motion traces,we integrate several features of the Tango devices, such as motion tracking and area learning. Technically speaking, area learning permits the Tango device to remember the visual features of the area it has visited and uses them to correct errors in its understanding of its position, orientation, and movement [7]. Moreover, the extracted footprints are then used as input to a Bayesian inference method which provides a probabilistic estimation of the room type. In this section, we perform two processing steps, namely *prediction-based segmentation* and *traces classification*. The former step partitions a collected trajectory $T_i = t_1, \cdots, t_u \mid t =< x, y >$ into several segments $S = S_1, \cdots, S_v$, where $u$ is the length of the trajectory and $v$ denotes the total number of segments. The partitioning decision is made in accordance with the occurrence of significant changes in the motion direction. Whereas, the latter step classifies the segments into room $S^{room} \in S$ and hallway $S^{hallway} \in S$ segments. To differentiate between these two classes, we adopt two criteria, including the segment length and the motion speed.

### 4.1 Traces Segmentation

In this section, we are interested in identifying the room segments $S^{room}$ of each trace $T_i$. The intuition behind this step is to find the user's footprint while scanning a certain room. This detected footprint is exploited to identify the most probable grammar rules (cf. Section 5.1). In fact, we found it is reasonable to partition a motion trace $T_t$ at the turns, i.e. when significant changes in the motion direction occur. To this end, we carry out a simple yet powerful prediction-based traces segmentation method. Several advanced approaches have been proposed in the literature [11, 35]. However, we tend to develop a simple approach which can be easily integrated with our ComNSense approach. Algorithm 1 lists the steps to perform prediction-based segmentation as well as segments classification into room and hallway segments.

The core idea behind our prediction-based segmentation method is to exploit *linear regression* to generate a model and then estimate its deviation from the actual trajectory. Figure 3a) depicts an explanatory example of the proposed method. The gray curve denotes the actual path traversed by a user, while the black dots represent the collected motion points. The proposed method uses a subset of the collected points to learn a linear model in the learning phase (line 2). We use a subset of the trace $T_i$ between two edge points (i.e. `StartPoint` and `EndPoint`) to generate the linear model. Subsequently, the learned model is used to predict the next points in the prediction phase (line 4). Whenever the difference between the predicted model and the actual points exceeds a predefined threshold $\varepsilon$, a new linear model is generated to describe the new motion direction. In general, linear regression fails to fit vertical lines. To overcome this limitation, we check the line "verticality" by comparing the standard deviations of $X$ and $Y$ values. In the vertical case (i.e. $\frac{\sigma(Y)}{\sigma(X)} > 1$), we regress a line through inverting the $Y$ and $X$ points, before we restore the regressed line.

To make the decision of generating a new model, we have to properly set the value threshold $\varepsilon$. We found that it is more robust to determine the threshold as the maximum between a relative error and an absolute acceptable error, i.e. $\varepsilon = max(\varepsilon^{abs}, \frac{y_i}{100}\varepsilon^{rel})$. The intuition behind this estimation is to adapt the threshold relative to the segment length. During our experiments, we observed that long segments encounter higher deviations. If we set an absolute value $\varepsilon^{abs}$, several models will be generated to describe the same segment. Similarly, setting only a relative threshold $\varepsilon^{rel}$ will override short traces. Figure 3b shows an example of applying the proposed method to a real trajectory of traversing the hallway and rooms to scan point clouds. The red-dotted lines represent the




(a) Generating a new linear model when deviation occurs

(b) An example of clustering a real trajectory into four segments

Fig. 3. Pose traces segmentation using linear regression

generated models while the blue crosses indicate the locations at which segments have been identified. Below, we explain the criteria upon which we classify a motion segment as room $S^{room}$ or hallway $S^{hallway}$ segment.

### 4.2 Traces Classification

To distinguish between hallway segments $S^{hallway}$ and room segments $S^{room}$, we adopt two criteria, namely the *segment length* and the *motion speed*. The former criterion has the intuition that hallway segments $S^{hallway}$ are typically larger than the room segments $S^{room}$. Additionally, the room segments $S^{room}$ are also constrained by the room dimensions. Similarly, users normally move faster in hallways than inside rooms. This hypothesis is supported by the fact that users should move slowly while scanning point clouds. Line 12 checks the length of the segments in comparison with the largest room width $\Psi$. To identify $\Psi$, we obtain the grammar rule which has the maximum room width. If the segment is smaller than $\Psi$, we check the user speed during this segment. If the speed is low, we annotate this segment as a room segment. Simultaneously, we estimate the most distant points in the room segments as the two edge points, i.e. `firstEdgePoint`, `secondEdgePoint`. These points represent the minimum and maximum coordinates parallel to the room width. Accordingly, the user's footprint is estimated as the Euclidean distance between these two edge points in the room segments $S^{room}$. Below, we harness the resultant footprint to infer the grammar rule corresponding to the scanned area.

## 5 HYPOTHESIS GENERATION

In this section, we discuss the Bayesian model and the grammar hypotheses generation. We start with inferring the most probable rules for the areas scanned by the mobile devices. Afterward, we present an algorithm to deliberately insert specific rules for the space scanned by the users.

### 5.1 Bayesian Rules Inference

Recalling that ComNSense harnesses the user footprint to infer the grammar rule $r_i$ corresponding to the scanned area. To this end, we propose a Bayesian approach to adjust the degree of belief in a certain grammar rule given a set of observed pose traces. In general, probability assessment quantifies the information gap between what is known, and what needs to be known for an optimal decision. *Bayesian inference* (BI) leverages fusing our prior knowledge of a given problem with the information extracted from the collected measurements. Recently, BI has been exploited in the realm of crowd-sensing for data analysis and prediction [12, 33]. Formally speaking, BI estimates a posterior probability as a consequence of two antecedents, a prior probability and a likelihood

function derived from a statistical model for the collected measurements [9]. Bayesian inference computes the posterior probability according to Bayes' theorem:

$$P(\text{Hypothesis } H_i \mid \text{Measurement } M) = \frac{P(M \mid H_i) \cdot P(H_i)}{P(M)}. \tag{2}$$

**ALGORITHM 1:** Traces segmentation and classification algorithm

**Input:** A sequence of motion data $T_i$, the grammar rules, the speed threshold $\Phi$
**Output:** hallway segments $S^{hallway}$, room segments $S^{room}$, the user footprint

```
while index < Length (T_i) do
    slope, intercept ← LearnModel (startPoint, endPoint)        // Starting a new learning phase
    while |predictedPoint − T_i[index]| ≤ ε do
        predictedPoint ← Predict (slope, intercept, index)      // linear regression-based prediction
        index ← index + 1
    end
    segmentList ← T_i[startPoint : index]
    startPoint ← index                                  // setting a new learning period
    endPoint ← index + α                                // α denotes the size of the learning window
end
foreach segment ∈ segmentList do                                // classifying the motion segments
    if Length (segment) ≤ Ψ then
        if segment.speed ≥ Φ then
            S^hallway ← segment
        else
            S^room ← segment
            firstEdgePoint ← Min(firstEdgePoint, Min(segment))
            secondEdgePoint ← Max(secondEdgePoint, Max(segment))
    else
        S^hallway ← segment
end
footprint ← |firstEdgePoint − secondEdgePoint|                  // estimating the user's footprint
return S^hall, S^room, footprint
```

Often, there exist several hypotheses $\mathcal{H} = H_1, \cdots, H_i$, and the BI's task is to determine which is the most probable hypothesis. The term $P(H_i)$ denotes the prior probability of a given hypothesis $H_i$ before incorporating the collected measurements $M$. The posterior probability $P(H_i|M)$ specifies the probability of a hypothesis $H_i$ given the measurements $M$ are already collected. Similarly, the likelihood probability $P(M|H_i)$ indicates the compatibility of the measurement $M$ with a given hypothesis $H$. Finally, the marginal likelihood $P(M) = \sum_{\mathcal{H}} (P(M \mid H_i) \cdot P(H_i))$—which is identical for all the available hypotheses—is solely used as a normalization factor. To select the most probable hypothesis given the measurements and prior knowledge, the maximum a posterior $H_{MAP}$ is obtained as follows.

$$H_{MAP} = \arg \max_{H \in \mathcal{H}} P(M \mid H) \cdot P(H) \tag{3}$$

In our problem, the unnormalized posterior probability distribution $P_{ap}(R = r_i|F) \propto P(F|R) \cdot P(R)$ estimates the probability of the current rule $R = r_i$ to represent the collected point cloud, provided that the mobile device's

user has a room footprint $F$. To compute the posterior probability, we have to estimate the prior probability $P(R)$ and the likelihood probability $P(F|R)$. Generally, the prior probability distribution can either be directly provided by the application or we can assume an appropriate probability distribution (e.g., uniform, Gaussian, or Poisson distribution). In our scenario, we exploit the available grammar rules to estimate an informative prior distribution. As aforementioned in Section 2.2, each terminal symbol $r_i \in V_T$ denotes a class of rooms $i \in \{a, b, c, \cdots\}$, identified by their geometric extent. Table 1 provides an example of two grammar rules $r_1^{room}$ and $r_2^{unit}$. Both rules describe an object $obj \in \{\text{room}, \text{room unit}\}$ in terms of the width, the type, and the a priori probability $P_a(r_i)$. The latter principally stands for the relative frequency of occurrence of a given object. The width $W_i^{obj}$ represents the distance between consecutive walls. As the frequency of occurrence provides a reasonable guess of the probable rule, we adopt the a priori probability as the prior probability distribution, i.e. $P(R) = P_a(r_i)$. In this way, each version of the Bayesian model is not starting from scratch, i.e. based only on the present data, but the cumulative effects of all data, past and present, can be taken into account.

Table 1. Example for room grammar rules and room unit grammar rules

| | $r_1^{room}$ | $r_2^{unit}$ |
|---|---|---|
| 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 |

To complete the definition of our Bayesian model, the likelihood distribution has to be specified. The likelihood function typically contains the available information provided by the trajectories. Specifically, we utilize the estimated footprint to infer the most probable rule. To this end, we map each rule $r_i$ onto a random variable $X_{r_i}$ which is normally-distributed, i.e. $X_{r_i} \sim \mathcal{N}(W_i, \sigma^2)$. The mean is given by the width $W_i$ associated with the rule $r_i$. During our experiments, we found a variance of one meter, i.e. $\sigma = 1$ is able to differentiate between all rules. Accordingly, the likelihood function $P(X_F = F_j|X_{r_i})$ can be simply estimated from the probability density function (PDF) of the rule random variable $X_{r_i}$ as follows

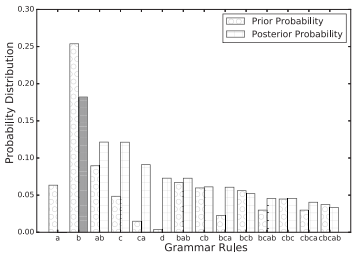
$$P(X_F = F_j|X_{r_i}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp[\frac{-1}{2\sigma^2} \cdot (F_j - W_i)^2]. \tag{4}$$

Considering the case in which a user scans several rooms, i.e. a room unit. The grammar is still robust to deal with these cases through the relationship probabilities between rules, provided by the indoor grammar. The relationship probability $P_r(r_i|r_j)$ is a conditional probability which models the neighborhood relationships between rooms or room units. As an example, assume a relationship probability $P_r(r_i|r_j) = 0.8$. This denotes that with a probability of 80%, room $r_i$ is adjacent to room $r_j$. In this case, we refine the prior probability $P(R)$ definition via incorporating the relationship probabilities $P_r(r_i|r_j)$. Accordingly, the prior probability is estimated as the transition probability between adjacent rooms, given by

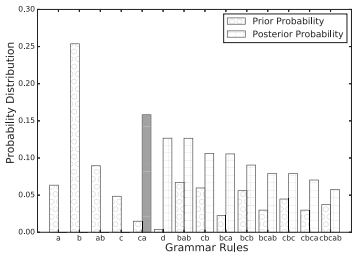
$$P(R) = \frac{P_a(r_j)}{P_a(r_i)} \cdot \frac{P_r(r_i|r_j)}{P_r(r_j|r_i)}. \tag{5}$$

In such cases, there exist hallway motion segments in between the room segments. Accordingly, the overall footprint is estimated as the sum of footprints in each room, i.e. $\sum_{l=1}^{\phi} F_l$ given that $\phi$ is the number of scanned rooms. In this manner, we can differentiate between rooms and room units which have the same width. Therefore, the likelihood distribution is estimated in a similar manner as in the single room scenario. Figure 4 shows two

examples of updating the prior probability distribution. The figures compares the prior and posterior distribution for all grammar rules. For instance, Figure 4a depicts the case of inferring the rule single rooms. As it can be seen, the shaded bar at rule "b" has the maximum posterior probability. Similarly, Figure 4b shows the case of inferring room units. In this figure, the Bayesian method selected the grammar rule "ca" where this rule has the maximum posterior probability. Below, we utilize the estimated rule sequence to derive the grammar hypothesis.




(a) An example of inferring rooms




(b) An example of inferring room units

Fig. 4. Bayesian-based inference of rooms and room units

## 5.2 Hypothesis Derivation

As explained earlier, the indoor grammar can generate plenty of possible hypotheses of the floor plan through randomly walking on the Markov chain. To match a hypothesis to a collected point cloud, we have to filter the hypotheses out to consider only the ones which conform to the user's motion. In this section, we explain how to insert the grammar rule $R_{MAP}$, with maximum a posterior probability, to derive a grammar hypothesis. Algorithm 2 summarizes the required steps to derive the grammar hypothesis. We start with identifying the Bayesian reference line segment (RLS)—along which the scanned room lies—exploiting the footprint $F$ (line 2). We iterate over all RLS segments to find the segment which has the minimum distance with the user footprint. If the current RLS segment is not Bayesian, then we follow the random walk on the Markov chain (lines 6-8). Otherwise, we consider the footprint and the rule $R_{MAP}$ while assigning the rules. As an illustration, Figure 5a depicts the process of cascading the rules. At the outset, we select a rule, i.e. `RuleCandidate` based on the prior probability. Then, we check the distance between the current location, assuming this rule is adopted, and the far edge of the footprint, i.e. $F$.`end` (line 12). If the distance is larger than the width of $R_{MAP}$, we adopt the rule candidate. If the condition does not hold, we reject the candidate rule and insert the rule $R_{MAP}$ instead.

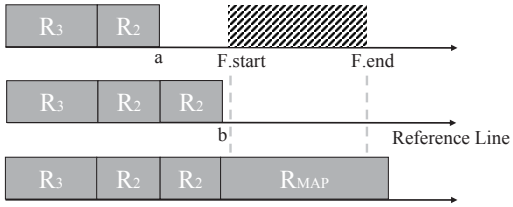
As shown in Figure 5a, the distance between the location $a$ and the footprint edge ($F$.end) is larger than the width of $R_{MAP}$, hence the rule $R_2$ is inserted. Conversely, $|b - F.\text{end}|$ is smaller than the rule width. In this case, we insert the rule $R_{MAP}$. It is important to realize that after each rule insertion we have to update the prior and relationship probabilities. After inserting the Bayesian rule $R_{MAP}$, we force the random walk on the Markov chain via setting `isRandomWalk` to true (line 20).

After selecting all rules, the proposed algorithm scales the selected rules for each RLS provided that the summation of all rules' width is equal to the RLS's length. i.e. $\text{RLS}.width = \sum_{i=1}^{Q} R_i.width/\chi$, where $Q$ denotes the number of rules assigned to the RLS and $\chi$ represents the scaling factor estimated for that RLS. Accordingly, the rules are either compressed or stretched before being converted into rooms or room units. It is worth to mention that our algorithm avoids hangovers at the Bayesian RLS segments. In fact, randomly walking on a RLS segment may lead to unavoidable overlapping with the next RLS segment. If the user footprint lies within the overlap region, it becomes infeasible to insert the Bayesian rule $R_{MAP}$. To overcome this limitation, we track the RLS, which precedes the Bayesian RLS, to adjust their rules' probabilities for the sake of removing possible hangovers.

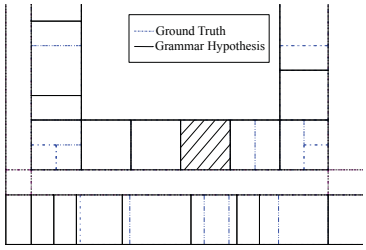
**ALGORITHM 2:** Grammar hypothesis generation

**Input:** the grammar rules, the room footprint $F$, the Bayesian rule $R_{MAP}$
**Output:** A grammar hypothesis

```
isRandomWalk ← false                                          // initialization
foreach currentReferenceLine ∈ referenceLinesList do
    bayesianRLS ← ISBayesianRLS (currentReferenceLine, F)
    while currentLength ≤ currentReferenceLine.width do
        if bayesianRLS is false or isRandomWalk is true then
            currentRule ← R_i | P_a(R_i), P_r(R_i|R_j)        // performing a random walk on the Markov chain
            Adjust the probabilities P_a, P_r
            currentLength ← currentLength + currentRule.width
        else                                                  // assigning rules to a Bayesian RLS
            ruleCandidate ← R_i | P_a(R_i), P_r(R_i|R_j)
            currentLocation ← currentLength + currentReferenceLine.start
            if |currentLocation − F.end| ≥ R_MAP.width then    // checking for space violation
                currentRule ← ruleCandidate
                Adjust the probabilities P_a, P_r
                currentLength ← currentLength + currentRule.width
            else
                currentRule ← R_MAP                           // inserting the Bayesian Rule
                Adjust the probabilities P_a, P_r
                currentLength ← currentLength + currentRule.width
                isRandomWalk ← true    // adopting the random walk after inserting the Bayesian rule
    end
end
return hypothesis  derived in the light of the motion traces and the a posterior probabilities
```




(a) Assigning the rooms while considering the motion traces and the inferred rule




(b) An example of hypoethesis generation with guiding the Markov chain to insert the rule $R_{MAP} = b$

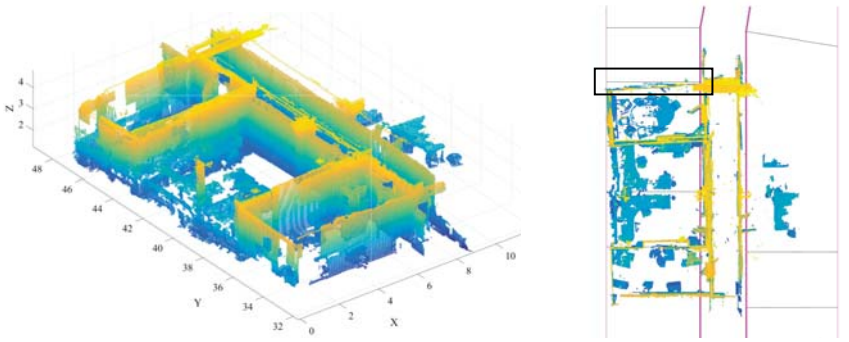
Fig. 5. Hypothesis generation

Figure 5b depicts an example of deriving a grammar hypothesis with considering the footprint and the rule $R_{MAP} = b$. In this figure, the dotted-dashed lines represents original walls existing in the ground truth, whereas the solid lines denotes the grammar-generated walls. As it can be seen, the grammar hypothesis does not assign the correct walls in several cases. However, the shaded room has been successfully inserted into its location within the RLS segment. It is important to mention that the generation of this hypothesis has been repeated

several times. In each iteration, the randomly assigned rooms changes while the rule $R_{MAP}$ is inserted in the same location. To sum up, we generate a grammar hypothesis while ensuring the correct allocation of the Bayesian rule. In the next section, we clarify how to match the collected point clouds to these generated hypotheses.

## 6 WALLS EXTRACTION

In this section, we describe how grammar hypotheses are to be matched with the collected point clouds. To clarify the proposed method, Figure 6a depicts a point cloud of three rooms and the hallway. Beside the walls, the shown point cloud also comprises furniture with a high level of noise. To reduce the search space during inferring the wall locations, we exploit the grammar rules to indicate the possible locations of walls, as shown in Figure 6b. The figure demonstrates the alignment of the point cloud to a grammar hypothesis (in magenta). If the grammar rules have been properly selected, they guide us to identify the wall locations in the collected point cloud. It is important to mention that the point clouds have to be registered, i.e. referenced to a unique origin point. Practically, we construct a bounding box around each grammar wall (cf. Figure 6b). This bounding box defines a subset of the point cloud. Accordingly, we limit our search for walls to this subset. However, we may encounter drifts which bring the wall outside this subset. In this case, we extend the bounding box to search for walls in the neighborhood (cf. Section 7.3). To detect a wall in this subset, we estimate the fraction of *surface normals*, orthogonal to the x-y plane. Literally, a normal to a certain surface at a point $p$ is a vector that is perpendicular to the tangent plane to that surface at $p$. If the fraction of similar normals exceeds a normals threshold, we annotate this plane as a wall segment. The problem of determining the normal to a point on the surface is approximated by the problem of estimating the normal of a plane tangent to the surface. Hence, normals estimation can be viewed as analyzing the principal component of points in the bounding box.




(a) Point cloud

(b) Aligning a point cloud to a grammar hypothesis

Fig. 6. Grammar-supported walls detection

### 6.1 Point Cloud Sampling

In general, the depth sensors, embedded in Tango devices, generate several snapshots while scanning a given area. Each snapshot comprises the 3D coordinates for as many points in the scene as are possible to calculate. These coordinates use the coordinate frame of the depth-sensing camera as the origin. We performed two steps to collect a proper point cloud, including (1) point cloud registration and (2) snapshots alignment. The former step denotes integrating area learning and depth perception features of the Tango technology. The intuition behind

integrating these features is to reference all collected point clouds to a unique coordinate system. In other words, the collected points are not anymore estimated relative to the focal center of the depth camera, instead they are computed relative to a stored ADF file.

The second step involves the alignment of the collected snapshots. The coordinate system of the range data from a tango device is relative to the depth sensor itself. This means that if we capture consecutive snapshots while moving, the snapshots will not display correctly relative to each other. To amend this, we utilize the motion traces to transform the collected snapshots, i.e. rotation and translation in space, into the same coordinate system. Specifically, the pose provided by Tango comprises two kinds of data, including: (1) a 4D quaternion $\overline{z} = a + b\text{fiŁ}\,\mathbf{i} + c\,\mathbf{j} + d\,\text{fiŁ}\mathbf{k}$ that defines the rotation of the target frame with respect to the base frame, where a, b, c, d are real numbers, and $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the fundamental quaternion units; (2) a 3D vector $< t_x, t_y, t_z >$ that defines the translation of the target frame with respect to the base frame. In order to obtain the point cloud in a local coordinate system, a $4 \times 4$ transformation matrix $\zeta$ has to be computed by augmenting a $3 \times 3$ rotation matrix $\Theta$ with the translation vector. The rotation matrix $\Theta$ serves as a clockwise/left-handed rotation by the unit quaternion, as expressed in Equation 6 [30]. Afterward, the rotated snapshots are aligned to their proper location via concatenating the translation vector (cf. Equation 7).

$$\Theta = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2(bc - ad) & 2(bd + ac) \\ 2(bc + ad) & a^2 - b^2 + c^2 - d^2 & 2(cd - ab) \\ 2(bd - ac) & 2(cd + ab) & a^2 - b^2 - c^2 + d^2 \end{bmatrix} \tag{6}$$

$$\zeta = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2(bc - ad) & 2(bd + ac) & t_x \\ 2(bc + ad) & a^2 - b^2 + c^2 - d^2 & 2(cd - ab) & t_y \\ 2(bd - ac) & 2(cd + ab) & a^2 - b^2 - c^2 + d^2 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

## 6.2 Normals Estimation

In this section, we discuss the normals estimation method through which we detect wall segments. To this end, the principal component analysis method is utilized to estimate the surface normals inside the bounding boxes. For a 3D point cloud, there exist three eigenvector/eigenvalue pairs. Given a geometric surface, the direction of the normal can be inferred at a certain point on the surface as the vector perpendicular to the surface in that point. Accordingly, the eigenvector $\overrightarrow{v_j}$, which has the least eigenvalue $\lambda_j$, represents the surface normal $n_i$. The surface normal $n_i$ at a point $p_i$ is determined from the surrounding point neighborhood support of the point (i.e. dotted circle of radius $\eta$). Formally, for each point $p_i \in P$, a covariance matrix $C$ is estimated for a set of neighboring points within the circle, as expressed in Equation 8 [29]. The variable $k$ denotes the number of neighboring points in the circle, while $\overline{p}$ stands for the 3D centroid of those neighbors.

$$C = \frac{1}{k} \sum_{i=1}^{k} (p_i - \overline{p}) \cdot (p_i - \overline{p})^T, \quad C \cdot \overrightarrow{v_j} = \lambda_j \cdot \overrightarrow{v_j}, \quad j \in \{1, 2, 3\} \tag{8}$$

The circle radius $\eta$ is an important design parameters where setting it to a large value results in a higher number of neighboring points. Hence, the computational overhead to estimate the normals increases. During our experiments, we select a moderate value which balances the detection accuracy while reducing the required computations.

### 6.3 Map Construction

To derive the final indoor model, the crowd-sensing server aggregates the extracted geometrical information from the mobile users. Collecting the mapping data from different sources may suffer from the presence of white spaces, which might occur due to: (1) the existence of no overlap between the collected data, and (2) the inaccurate detection of the wall segments owing to localization errors or imprecise sensor values from the commodity sensors. To overcome these problems, ComNSense utilizes the indoor grammar at the server side to provide a complete coverage. Specifically, the server checks whether the width of these white spaces are less than the minimum room width (i.e. obtained from the grammar rules). If this condition holds, the server immediately ignores these spaces and it then combines the neighboring segments, otherwise new sensing queries are distributed for these spaces. Consequently, the knowledge provided by the grammar significantly helps while crowd-sensing the geometrical information.

After filling all white spaces in each RLS segment, the server proceeds with constructing the indoor map. To this end, we harness a probabilistic grammar-based approach which was developed in a previous work by our group [23]. The method relies on performing a random walk on a Markov chain in which the transition probabilities are estimated based on prior and relationship probabilities. The grammar assigns rooms and room units on a set of predefined RLS segments. In [23], some constraints—extracted from motion traces—are used to direct the transition between the Markov states. ComNSense generates the final indoor model via converting the extracted geometrical information into constraints for deriving the Markov process. This implies that the grammar checks for the suitability of each rule in the light of the detected wall locations. This process resembles the hypothesis derivation step (cf. Section 5.2), but with applying the rule insertion procedure on all spaces along the various RLS segments. Thus, the output of our method is a floor plan of the scanned area including the rooms and their associated hallways. A more detailed overview of such an approach can be found in [23].

## 7 PERFORMANCE EVALUATION

In this section, we assess the effectiveness of ComNSense in efficiently acquiring the required geometrical information from two different buildings. We first describe the setup of our evaluations. Afterward, we evaluate the traces segmentation and classification method, before we compare the detected Bayesian rules with the actual rules. To clearly interpret the performance of our ComNSense approach, we compare it with a grammar-free method for acquiring and processing point clouds (cf. Section 2.1). In the grammar-free approach, compressed versions of the collected point clouds were uploaded. For ComNSense, the motion traces as well as the files containing the walls location were uploaded. The comparative study is performed in terms of: (1) the total energy consumption defined as the sum of processing energy and uploading energy, (2) the execution time elapsed while processing the point clouds on the mobile devices, (3) the fraction of detected walls, and (4) the accuracy of the detected walls location.

### 7.1 Experimental Setup

For sampling point clouds, we utilized Tango mobile devices which are equipped with a 7-inch HD display, 4 GB of RAM, 128 GB of internal flash storage, a 120° front-facing camera, a 4 MP RGB-IR rear-facing camera, and a 170° motion tracking camera [4]. We started with designing an Android App for collecting motion traces and point clouds. Through an offline phase, area learning is used to store the visual landmarks of the entire floor. In the online phase, the ADF files were primarily distributed to all participating mobile users. The intuition is to register all collected point clouds relative to the same ADF file. Now, when a user is queried to scan a certain area, the App generates two output files, representing the point cloud (e.g. in xyz format) and the accompanied pose traces. Once scanning is finished, the App combines several scene snapshots and performs several transformations to properly align the snapshots, i.e. registration.

Table 2. parameters used in the evaluations

| System Parameter | Value | System Parameter | Value |
|---|---|---|---|
| speed threshold $\Phi$ | 0.3 m/sec | normals filtering angle | 60° |
| learning window $\alpha$ | 3 | voxel grid size | 2 cm |
| maximum room size $\Psi$ | 9.7 m | normals percentage threshold | 70% |

To process the point clouds, we utilized the *point cloud library* (PCL), a standalone open-source project of C++ functions for manipulating point cloud data [29]. We integrated some of these functions in our implementation. For instance, Octree compression and PCA-based normals estimation have been implemented on the Tango devices. Whereas, the back-end servers utilizes region growing segmentation, Octree decompression, convex hull detection, and geometric projection. Table 2 summarizes the parameters and their values which have been used throughout the evaluations.

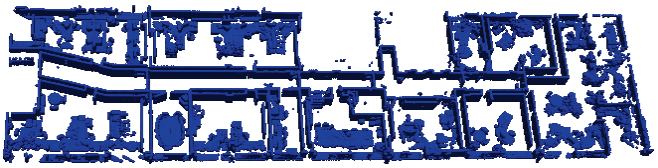
## 7.2 Data Sampling

To evaluate the performance of ComNSense in different environments, we performed our experiments in two different buildings: (1) the second floor of the IPVS building (cf. Figure 7b), University of Stuttgart, and (2) the fourth floor of the ifp building (cf. Figure 7c), University of Stuttgart. The IPVS building has four identical quadrants, hence the results received from one quadrant can be simply applied to other quadrants. Therefore, we limit our evaluations to the quadrant which is easily accessible by our volunteers. To avoid the accumulating drifts of the motion traces, we integrate motion tracking and area learning features of the Tango technology. Area learning stores where the device has recorded the area's landmarks in an area description file (ADF). When a Tango device has learned an area, it estimates the instantaneous position relative to a base frame, which is fetched from the ADF model. Additionally, we combine area learning with depth perception to move the scanned point clouds to their actual locations in lieu of being coincided in the same location.

Figure 7a demonstrates a comparison between three kinds of motion traces. The red dot-dashed lines, shown in the figure, represent the motion traces when area learning is disabled. As it can be seen, the red trajectories suffer from cumulative drifts and their orientation is highly dependent on the starting point. Alternatively, the black dashed trajectories stand for the traces collected with a "fresh" ADF file loaded. Freshness means that lighting during the point cloud acquisition is close to that during ADF recording. In this case, the trajectories follow the actual path traversed by the volunteers. Finally, the blue dotted trajectories are those traces collected with an "outdated" ADF. The term outdated implies that the environment changes over time. Specifically, area learning are sensitive to changes in lighting. The blue trajectories have been drawn here to demonstrate this shortcoming of area learning. As a solution, we switch between different ADF files recorded at different times of the day. In our evaluations, we collected 135 point clouds, i.e. 100 from the IPVS building and 35 from the ifp building, plus the associated motion traces from both buildings. To enhance the collected dataset, we have collected several point clouds for each area within two days. Four volunteers were asked to move freely between the hallways and the different rooms before reporting their data to the crowd-sensing server. Nevertheless, they had to follow two constraints: (1) slowly scanning the non-hallway areas and (2) keeping a distance of 0.5m between the mobile device and the surroundings [3]. If these conditions are violated, the Tango core, which provides core functionality for Tango applications, swiftly crashes. Figures 7b,7c depict samples of the collected point clouds where the point clouds have been not coincided in a certain location due to capturing them relative to a unique ADF model. Being relative to a unique ADF model implies that each point in the collected point clouds are referenced to the origin of the ADF model in lieu of being referenced to the focal point of the fisheye


(a) Samples of the collected motion traces from IPVS building

(b) Samples of the collected Point clouds from IPVS building



(c) Samples of the collected Point clouds from ifp building

Fig. 7. Combined point clouds and motion traces from two buildings

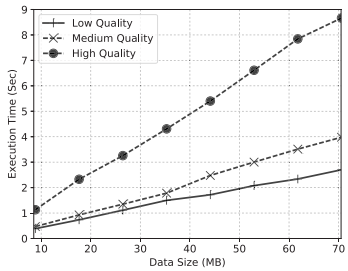
camera of the Tango device. Below, we first start with evaluating the energy performance of ComNSense and the base line method, before examining the accuracy of the final indoor model in both cases.

### 7.3 Energy Consumption

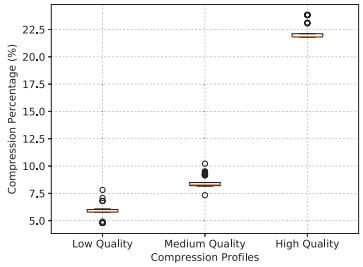
Energy is an extremely important resource for battery-powered mobile devices. Therefore, we evaluated the energy consumption of our ComNSense approach as well as of the baseline method in different environments. Specifically, we quantify the energy burden of processing and uploading the point clouds as well as the motion traces to the crowd-sensing server. For the power measurements, we found that our Monsoon power monitor is unable to power the Tango devices. The latter requires a 7.5 Volts power supply whereas the Monsoon power monitor can only supply a maximum of 4.5 Volts. Hence, the energy measurements were achieved through logging the battery's current and voltage. During the execution of each method, these current and voltage values associated with timestamps were recorded every 100 ms. The energy measurements were performed while disabling WiFi and no other background activity existed. The screen energy (at 50% brightness level) has been subtracted from the total energy consumption to consider only the energy overhead of running the algorithms. Each run of these measurements was repeated ten times and the resultant values are then averaged.

At the outset, we examine different compression profiles of the baseline method. For the Octree compression used in the grammar-free approach, we started with identifying the compression profile, adopted while performing the comparative study. Figure 8a depicts the time elapsed while compressing a set of point clouds of different sizes. We compare the incurred delay in three compression profiles, namely *low quality*, *medium quality*, and *high quality*. As shown in the figure, the elapsed time gradually increases with the size of the collected point clouds. Moreover, adopting the high quality compression profile—which is sometimes required in order to preserve as much details as possible—induces higher execution time as well as uploading energy consumption. Figure 8b

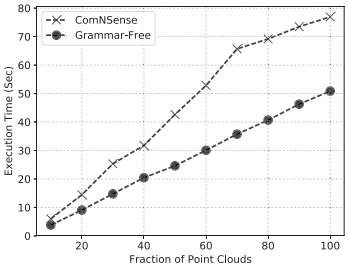
compares the compression percentage—defined as the percentage of output data relative to the original point cloud—of the three profiles while compressing the point clouds from the IPVS building. Apparently, the low-quality profile highly reduces the amount of transmitted data (at least by 31% compared to the medium-quality profile and 73.5% relative to the high-quality profile). We selected to compare our ComNSense approach with the low-quality profile since it requires shorter execution time compared to the two other profiles.



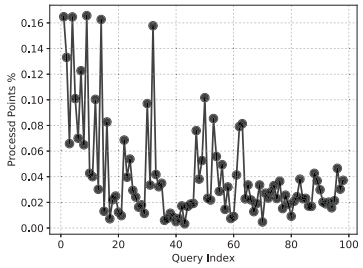

(a) Execution time of Octree compression on the mobile device



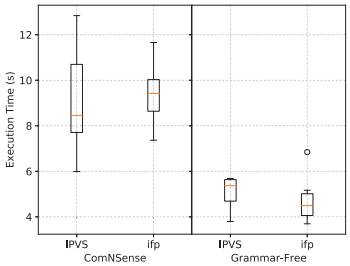

(b) Compression percentage of the various compression profiles



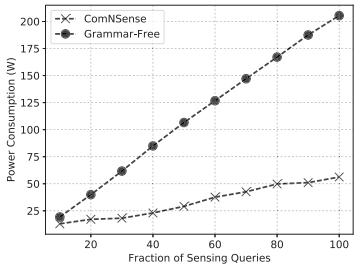

(c) Execution time on the mobile device



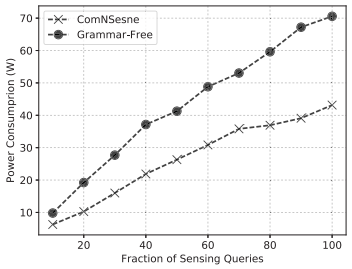

(d) Fraction of the processed points inside the bounding box



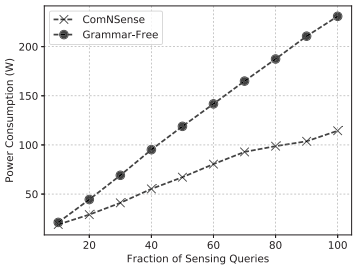

(e) Execution time in both use cases



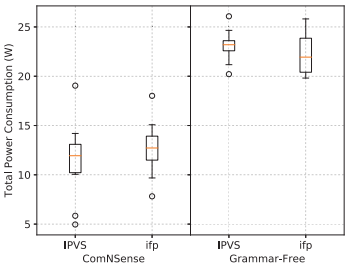

(f) Uploading energy consumption




(g) Processing energy consumption




(h) Total energy consumption




(i) Total power consumption in both uses cases

Fig. 8. Evaluation results

Since the execution time of running the algorithms highly affects the energy consumption of the mobile devices, we quantify the execution time in both use cases. Figure 8c depicts the cumulative execution time of both compared method versus the number of processed point clouds. The figure implies that the low-quality Octree compression requires less time (up to 42%) to process the point clouds. The reason is that ComNSense searches for walls in the region, i.e. bounding box, specified by the grammar hypothesis. However, the collected point clouds are sometimes slightly shifted due to the inexact localization (cf. Figure 7b). Hence, our proposed approach compensates for these shifts through extending the bounding box to search the neighborhood for walls. If the percentage of normals—whose angle is larger than 60°—does not exceeds a percentage threshold, the algorithm continues searching the neighboring points. As a result, the execution time of the wall extraction algorithm increases in the presence of such drifts. Despite these possible drifts, ComNSense tremendously reduces the fraction of the processed points.

Figure 8d shows percentages of the processed points, in each sensing query, relative to the total number of points in the collected point cloud. In most cases, the bounding box, specified by the grammar hypothesis, directly finds the walls location. In these cases, the percentage of the processed points was less than 0.08%. In case of drift occurrence, the algorithm repeats the search process. Nevertheless, the percentage is still less than 0.16% of the total number of points. Figure 8e compares the execution time of the algorithms while processing different data bundles—each having ten point clouds—in both buildings. Similar performance is obtained in both uses cases where ComNSense spends longer execution time than the grammar-free approach (at most by 43% in the IPVS building and 48% in the ifp building).

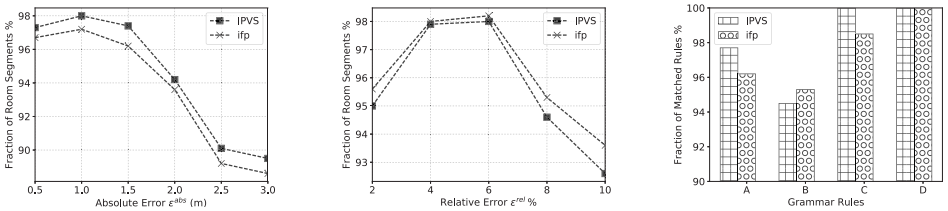
Figure 8f compares the cumulative uploading power consumption required for both methods in the IPVS building. The raw point clouds have different sizes, ranging from five MBytes to 50 MBytes. It is obvious that ComNSense tremendously reduces the energy of uploading the data (at least by 85%) to the corresponding crowd-sensing server. This saving emerges from solely reporting the walls location which are then used as an input to the grammar-based map construction algorithm at the back-end server (cf. Section 6.3). Nevertheless, this saving in the uploading energy consumption comes at the expense of increasing the processing energy consumption. As expected from Figure 8c, the ComNSense approach consumes more energy than the grammar-free approach (at most by 37.5%) while searching for the walls location. Figure 8g depicts the processing energy consumption required by ComNSense and the grammar-free approach.

However, we can clearly judge the compared methods through combining the total energy consumed in both methods. Figure 8h shows a comparison in terms of the overall energy consumption. In both methods, the overall energy consumption can be defined as the summation of processing and uploading energy. Additionally, sending the motion traces as well as the reception of the grammar hypotheses have been also considered in the ComNSense method. As the figure shows, ComNSense outweighs the grammar-free approach consuming, on average, 56% less energy. Finally, Figure 8i demonstrates a comparison between the total energy consumption of executing the adopted approaches on several data bundles collected from both buildings. Nearly, the adopted methods consume a comparable amount of energy in both buildings where ComNSense consumes less energy (at least by 49.5% for the IPVS building and 43% for the ifp building). To sum up, the evaluation results confirmed that ComNSense outweighs the grammar-free approach in terms of the consumed energy. Below, we examine the modeling accuracy of ComNSense and the baseline method.

## 7.4 Hypothesis Derivation

As explained earlier, ComNSense requires the derivation of a set of grammar hypotheses to process the collected point clouds on the mobile devices. To ensure the correctness of the generated hypotheses, we evaluate here the traces segmentation and classification, before assessing the ability of the proposed Bayesian method to infer the correct rules. Most of the traces were collected between rooms and hallways. Moreover, some traces contain only

room segments to examine the cases in which the users are already inside the queried rooms. Several ADFs were recorded to select the one whose recording time is close to the time of collecting the point clouds. Figures 9a, 9b demonstrate the accuracy of detecting room segments versus different values of the absolute error $\varepsilon^{abs}$ and the relative error $\varepsilon^{rel}$. The figures show that the proposed method has a nearly similar performance when being adopted in different buildings, i.e. IPVS and ifp. As depicted in Figure 9a, the highest performance, i.e. detection accuracy of 98%, occurs at low values of $\varepsilon^{abs}$, ranging from 0.5 to 1.5$m$. Beyond this range, the proposed method tends to overlook small motion segments. Similarly, Figure 9b shows that the highest performance occurs when $\varepsilon^{rel}$ varies between 4% and 6%. It is important to mention here that the missing room segments were recorded while the volunteers were quickly scanning some rooms. Accordingly, their motion speed inside these rooms violated the speed condition, expressed in Algorithm 1. Furthermore, Tango, in these cases, loses the ability to collect the depth data. For the rest of our experiments, we set the relative error to 5% and the absolute error to 1$m$. Moreover, we decided to skip these inaccurate traces while deriving the grammar hypotheses.




(a) Detected room segments at $\varepsilon^{rel} = 5\%$

(b) Detected room segments at $\varepsilon^{abs} = 1m$

(c) Matched grammar rules at $\varepsilon^{rel} = 5\%$, $\varepsilon^{abs} = 1m$

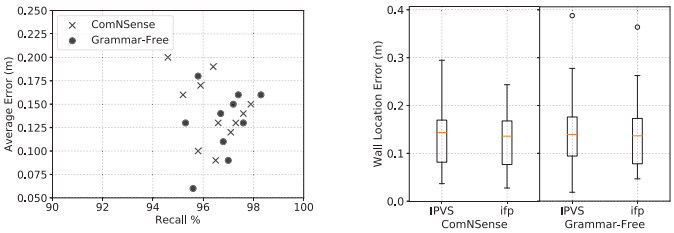
Fig. 9. Evaluating the grammar derivation steps in both buildings

Next, we evaluate the performance of our Bayesian method in properly inferring the grammar rules, which are consistent with the user footprints. Figure 9c demonstrates a comparison between the matched rules in both buildings. The horizontal axis denotes the detected rules, while the vertical axis represents the percentage of correct detections. For IPVS, rule "a" (i.e. room width is equal to 2.4m) was successfully detected in 97.7%, i.e. 44 motion trace, out of 45, result in detecting this rule. For both of rules "c" and "d", the Bayesian method entirely detected the correct rules. Similar performance was obtained in the ifp building. It is worth to mention that these undetected rules correspond to two extremely crowded rooms. Hence, it was not trivial to move freely along the room width.

## 7.5 Modeling Accuracy

In this section, we examine the detection accuracy using both methods. To this end, we utilize the *recall* metric as a measure of the detection accuracy. Recall is defined as the fraction of relevant instances that are retrieved, i.e. $\frac{TP}{TP+FN}$ where $TP$ and $FN$ denote true positive and false negative, receptively. Figure 10a depicts the recall metric versus the average error, i.e. defined as the distance between the actual and detected wall locations. In this figure, we ran the two methods ten times where several point clouds were examined for each non-hallway area. The figure shows that both approaches have a comparable performance. For instance, the percentage of correct detection using the ComNSense approach ranges from 94.5% to 98%. Furthermore, the average error of both methods did not exceed a value of 0.2 m, even with the existence of clutter.

Figure 10b compares the performance of ComNSense in the two adopted use cases, i.e. IPVS and ifp buildings. For this comparison, we estimated the detection error for matching 30 point clouds from both buildings. Clearly,

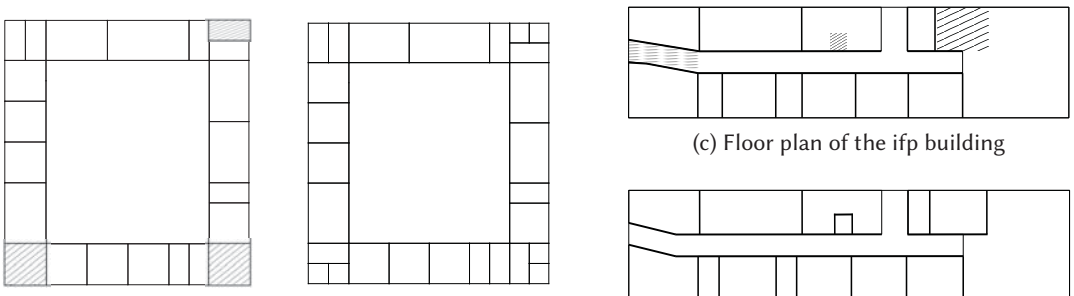



(a) Average error versus the fraction of matched rooms

(b) Wall detection error in both buildings

Fig. 10. Evaluating the accuracy of walls extraction and indoor mapping

both use cases achieve approximately similar performance. Finally, Figure 11 compares the obtained indoor model—when adopting ComNSense—with the ground truth in both uses cases. As it can be seen, the obtained models tremendously resemble the actual layout with few exceptions, including (1) overlapped rooms which can be added if multi-split grammar rules are adopted, and (2) some undetected walls due to clutter existence.



(a) Left-bottom quadrant of the IPVS building

(b) IPVS ground truth

(c) Floor plan of the ifp building

(d) ifp ground truth

Fig. 11. Indoor models obtained from constraint-driven walk on the Markov chain

## 7.6 Discussion

Apparently, our ComNSense approach managed to reduce the energy overhead of acquiring point clouds while maintaining a sufficient level of accuracy. As aforementioned, ComNSense takes the point clouds, the motion traces and the indoor grammar as inputs. For the point clouds, they are to be collected from different Tango devices. However, ComNSense ensures that they will not be coincided in a given location thanks to the area learning model shared among these mobile devices in the offline phase. For example, if two users receive a sensing query to scan a wall, then the extracted wall should present in the same location in both point clouds where this location is relative to the origin of the ADF model.

For the indoor grammar, we would like to mention that our grammar-based approach is targeting buildings with regular structures, which typically can be found in office buildings, hospitals, the buildings on a campus, etc. It is less suited for irregular building structures since this leads to complex grammars and consequently a high

effort to define the grammar. It is part of our future work to design concepts to learn grammars automatically, to support such scenarios, however this is beyond the scope of this paper. For now, we focus on regular building structures, more specifically, rectangular structures.

## 8 RELATED WORK

In this section, we discuss the recent efforts in the realm of crowd-sensing and indoor mapping. Moreover, we provide examples of formal grammar utilization for compressing bulky data and for deducing behaviors and shapes.

### 8.1 Localization & Mapping

Most indoor mapping methods are built on the basis of the Simultaneous Localization and Mapping (SLAM) principle, originating from robotics. In general, SLAM is concerned with the problem of building a map of an unknown environment by a mobile robot while at the same time navigating the environment using the map. In fact, the primarily challenge in SLAM-based systems is to recover the camera pose as well as the map structure while initially knowing neither. In ComNSense, we use the pose obtained from smart mobile devices to construct the indoor map. In this realm, several researchers derived 2D indoor maps through crowd-sensing mobile data such as acceleration, pressure, audio, images, and WiFi footprints [1, 10, 23, 26]. In [23], we introduce MapGENIE, an architectural framework for the automatic generation of indoor maps supported by structural information. The core idea is to derive an initial floor plan from a set of inaccurate traces. Afterward, the inaccurate traces are refined using information about the building exterior. Similarly, Alzantot and Youssef [1] present CrowdInside, a crowd-sensing-based system which constructs the general layout of buildings. To enhance the dead-reckoning accuracy, CrowdInside resets the accumulation of error through detecting unique anchor points which are found in typical indoor spaces. Based on crowd-sensed images and motion traces, Chen et al. [6] implement IndoorCrowd2D, a method which generates multiple panoramic images for visualizing the building interior view. Luo et al. [19] introduce iMap, a smartphone-based opportunistic sensing system that detects the floor plan as well as higher-level semantics such as stairs, escalators, elevators and doors. The authors collected several types of data including motion traces, atmospheric pressure, audio, and WiFi signal strength. Robertson et al. [26] introduce FootSLAM, a system that employs foot-mounted IMU sensors to reconstruct the building's interior as a map of walkable areas, not distinguishing between hallways or rooms. In contrast to the aforementioned approaches, we target, in this paper, the acquisition of 3D point clouds from a crowd of users in an energy-efficient manner. To this end, we harness recently developed mobile platforms, such as Google Tango and Apple ARKit, to directly capture point clouds. Table 3 compares between various crowd-sensing techniques for automatically generating indoor maps.

Several approaches have been devoted to develop efficient indoor localization systems [22, 27]. To collect the positioning data, the researchers often adopt either wireless infrastructure (i.e. WiFi, Bluetooth) or the sensors embedded in smart mobile devices. The former strategy is hardly accurate owing to the time-varying nature of the wireless signals. Moreover, they can be energetically expensive with continue updates. The second strategy has been usually employed along with machine learning algorithms to properly localize the users. For instance, Rottmann et al. [27] introduce a supervised learning method to classify the position of a robot on objects extracted from a set of 2D images and range scans. To this end, the authors adopt a two-stage classification approach, including the AdaBoost algorithm which boosts weak features extracted from the input data; and a Hidden Markov model (HMM) which filters possible outliers. Alternatively, our work mainly targets reducing the energy overhead of acquiring the mapping data using personal mobile devices while preserving the mapping accuracy. To this end, we exploit the design-time structural knowledge encoded in a formal grammar. Furthermore, we adopt Bayesian inference for classifying the scanned rooms according to their dimensions. Park and Teller [22] present

Table 3. Comparison of different crowd-sensing techniques for indoor mapping

| | Crowd-Sourcing | Input | Output | Methodology | QoS Metrics | Real-time |
|---|---|---|---|---|---|---|
| CrowdInside [1] | opportunistic | motion traces and WiFi fingerprints | indoor layout and PoIs (elevators, stairs, and escalator) | 2D geometric fitting | accuracy | no |
| iMap [19] | opportunistic | motion traces, atmospheric, pressure, audio, and WiFi fingerprints | indoor layout and PoIs (elevators, stairs, and escalator) | 2D geometric fitting | accuracy | no |
| iFrame [25] | opportunistic | motion traces, Bluetooth fingerprints, and WiFi fingerprints | indoor layout | 2D geometric fitting and curve fit fusion | accuracy | no |
| MapGENIE [23] | opportunistic | motion traces | indoor layout | formal grammars and 2D geometric fitting | accuracy | no |
| IndoorCrowd2D [6] | participatory | motion traces and 2D images | panoramic images with a navigable hallway skeleton | data fusion and normalization based on Reduced Manhattan World | accuracy | no |
| iMoon [8] | participatory | 2D images, motion traces, and WiFi fingerprint | navigation meshes | point clouds generation and mesh extraction | accuracy and latency | no |
| Jigsaw [10] | participatory | motion traces, WiFI fingerprints, and 2D images | indoor layout and PoIs (elevators, stairs, and escalator) | 2D geometric fitting | accuracy | no |
| FootSLAM [26] | opportunistic | motion traces | indoor layout | 2D geometric fitting | accuracy | no |
| CIMloc [36] | opportunistic | motion traces and WiFi fingerprints | indoor layout | 2D geometric fitting | accuracy | no |
| ALIMC [37] | opportunistic | motion traces and WiFi fingerprints | indoor layout and PoIs (elevators, stairs, and escalator) | activity detection and 2D geometric fitting | accuracy | no |
| Our approach: ComNSense | participatory | 3D point clouds | indoor layout[1] | formal grammars and 3D geometric fitting | accuracy and energy consumption | no |

an indoor localization method based on the sensors embedded in off-the-self mobile devices. They employ a HMM model to match the motion estimates to a prior map consisting of a set of 2D floor plans connected through horizontal and vertical adjacencies. Along a different line, ComNSense deliberately collect data to construct indoor maps from individuals traversing the building's hallways and rooms. We rely on the localization capabilities provided by modern smart devices, e.g. Google Tango and Apple ARKit.

## 8.2 Indoor Grammars

In this section, we discuss some recent endeavors for exploiting formal grammars for indoor mapping and compression. In fact, grammar-based modeling approaches typically outperform data-driven approaches based purely on geometric fitting [18]. These results occur thanks to harnessing the high-level structure information encoded in the indoor grammars. In this realm, there exist several articles which utilize different kind of grammars to efficiently generate indoor models [13, 16]. Recently, Ikehata et al. [13] devise a framework to model indoor environments as a structure graph whose nodes represent rooms, doors, and objects while their geometrical relationships form the edges. To adopt such a graph to a certain building, a structure grammar defines a set of possible transformations based on a set of collected 3D point clouds, i.e. segmentation, merging, and

[1]Extending the approach to also detect higher level semantics is planned in our future work

reconstruction. Alternatively, our formal grammar utilizes a probabilistic approach based on Markov chains to overcome uncertainties and noise present in the acquired data. Khoshelham et al. [16] adopt the shape grammars for modeling regular structures, i.e. cuboid spaces repeatedly appearing in indoor environments. A unit cube is used as the starting symbol of the shape grammar with providing three production rules for placing, connecting, and merging cuboids. To learn parameters of the grammar rules, they extracted the histogram peaks of projecting a raw point cloud on $x$, $y$, and $z$ coordinates. Nevertheless, this shape grammar-based approach is only valid for Manhattan-world scenarios. Moreover, existence of ceiling points is necessary for the points-on-ceiling test that excludes invalid sub-spaces made by the integrating cell decomposition step of the reconstruction process. Along a different line, ComNSense mainly tackles the trade-off between energy efficiency and modeling accuracy when crowd-sensing the mapping data, i.e. point clouds, using personal mobile devices. Furthermore, our formal grammar can readily be extended to describe irregular structures [5].

Aside from indoor modeling, formal grammars have been exploited for data compression and analysis. For instance, Kensuke et al. [15] introduce a grammar-based map compression framework. Manhattan World assumptions are used to generate a set of rules for a *context-free grammar* (CFG). This framework assumes that the majority of surfaces (e.g. walls, floors, ceilings) in man-made environments are typically aligned with dominant orthogonal planes. The main idea is to find a compact representation of the data by replacing them with the corresponding rules. Afterward, the CFG grammar is converted into a bit string using an arithmetic encoding technique. Although the results showed that high compression ratios can be achieved, the approach is still limited to buildings which follow the Manhatten assumptions. Moreover, the authors assumed that the indoor map has already been derived. Alternatively, we seek in our work to exploit the grammar while generating the indoor map through efficiently extracting the geometrical information. Similarly, Kieffer et al. [17] investigate a grammar-based lossless source coding method. The core idea is to generate a grammar using the data to be compressed. Subsequently, an encoder transmits code bits to the decoder that allow reconstruction of the grammar, from which the decoder infers the original data. Lymberopoulos et al. [20] present a framework that utilizes a hierarchy of *probabilistic context free grammars* (PCFGs) to parse out observable activities into a set of distinguishable actions. Each grammar layer in the hierarchy produces an output that interprets and summarizes its inputs. The framework commences with extracting image features that are analogues to phonemes in natural language. Through subsequent layers of the grammar, these phonemes are interpreted into words (i.e. symbols), sentences (i.e. behaviors), and paragraphs (i.e. macroscale behaviors) that describe the happenings inside the sensed area. In contrast, ComNSense approach performs indirect compression via uploading only the extracted geometrical semantics.

## 9 CONCLUSION AND FUTURE WORK

In this paper, we presented the ComNSense approach, which exploits formal indoor grammars for efficiently crowd-sensing 3D point clouds. At the outset, we exploited motion traces and a Bayesian-based rules inference method to derive a set of grammar hypotheses. Subsequently, these hypotheses were matched with the collected point clouds to extract the geometrical semantics. To assess the performance of ComNSense, we conducted extensive experiments for collecting motion traces and processing point clouds on our mobile devices. The results showed that ComNSense detects the walls in the point clouds while adding negligible computational overhead on the mobile devices. Moreover, it significantly reduces the uploading energy consumption in comparison with the grammar-free approach.

So far, ComNSense enables the detection of walls in point clouds. However, deriving detailed indoor maps requires the extraction of other geometrical semantics, e.g. ceiling, floor, furniture. Therefore, a reasonable next step will be generalizing the proposed approach so that it can extract walls as well as other semantics. To this

end, we plan to develop a 3D indoor grammar which not only encodes wall segments, but also other higher level semantics, e.g. doors, windows.

## ACKNOWLEDGMENTS

The authors would like to thank Martin Graner for his help in the implementation of the Android App.

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

## REFERENCES

[1] M. Alzantot and M. Youssef. 2012. CrowdInside: Automatic Construction of Indoor Floorplans. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*. ACM, 99–108.

[2] I. Anagnostopoulos, V. Pătrăucean, I. Brilakis, and P. Vela. 2016. Detection of Walls, Floors, and Ceilings in Point Cloud Data. In *Proceedings of the Construction Research Congress 2016*. 2302–2311.

[3] ATAP. 2016. Depth Perception. (2016). https://developers.google.com/tango/overview/depth-perception accessed on July 2016.

[4] ATAP. 2016. Google Project Tango. (2016). https://developers.google.com/project-tango/ accessed on June 2016.

[5] S. Becker, M. Peter, and D. Fritsch. 2015. Grammar-supported 3D Indoor Reconstruction from Point Clouds for" as-built" BIM. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 2, 3 (2015), 17.

[6] S. Chen, M. Li, K. Ren, X. Fu, and C. Qiao. 2015. Rise of the Indoor Crowd: Reconstruction of Building Interior View via Mobile Crowdsourcing. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15)*. ACM, 59–71. https://doi.org/10.1145/2809695.2809702

[7] Google Tango Developers. 2016. Integrating Motion Tracking with Area Learning. (2016). https://developers.google.com/tango/overview/area-learning accessed on May 2016.

[8] J. Dong, Y. Xiao, M. Noreikis, Z. Ou, and A. Ylä-Jääski. 2015. iMoon: Using Smartphones for Image-based Indoor Navigation. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15)*. ACM, 85–97. https://doi.org/10.1145/2809695.2809722

[9] N. Fenton and M. Neil. 2012. *Risk Assessment and Decision Analysis with Bayesian Networks*. CRC Press.

[10] R. Gao, M. Zhao, T. Ye, F. Ye, G. Luo, Y. Wang, K. Bian, T. Wang, and X. Li. 2016. Multi-Story Indoor Floor Plan Reconstruction via Mobile Crowdsensing. *IEEE Transactions on Mobile Computing* 15, 6 (June 2016), 1427–1442. https://doi.org/10.1109/TMC.2016.2550040

[11] J. Gudmundsson, A. Thom, and J. Vahrenhold. 2012. Of Motifs and Goals: Mining Trajectory Data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*. ACM, 129–138.

[12] G. Hartl and B. Li. 2005. Infer: A Bayesian Inference Approach towards Energy Efficient Data Collection in Dense Sensor Networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*. 371–380.

[13] S. Ikehata, H. Yang, and Y. Furukawa. 2015. Structured Indoor Modeling. In *Proceedings of the IEEE International Conference on Computer Vision*. 1323–1331.

[14] Ian Jolliffe. 2002. *Principal Component Analysis*. Wiley Online Library.

[15] K. Kensuke, T. Kanji, and N. Tomomi. 2011. Grammar-Based Map Compression Using Manhattan World Priors. In *Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2612–2617.

[16] K. Khoshelham and L. Díaz-Vilariño. 2014. 3D Modelling of Interior Spaces: Learning the Language of Indoor Architecture. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 40, 5 (2014), 321.

[17] J. Kieffer and E. Yang. 2000. Grammar-Based Codes: a New Class of Universal Lossless Source Codes. *IEEE Transactions on Information Theory* 46, 3 (2000), 737–754.

[18] M. Li, P. Wonka, and L. Nan. 2016. Manhattan-World Urban Reconstruction from Point Clouds. In *European Conference on Computer Vision*. Springer, 54–69.

[19] C. Luo, H. Hong, L. Cheng, K. Sankaran, and M. C. Chan. 2015. iMap: Automatic Inference of Indoor Semantics Exploiting Opportunistic Smartphone Sensing. In *Proceedings of the 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 489–497.

[20] D. Lymberopoulos, A. Ogale, A. Savvides, and Y. Aloimonos. 2006. A Sensory Grammar for Inferring Behaviors in Sensor Networks. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN'06)*. ACM, 251–259.

[21] B. Okorn, X. Xiong, B. Akinci, and D. Huber. 2010. Toward Automated Modeling of Floor Plans. In *Proceedings of the Symposium on 3D Data Processing, Visualization and Transmission*, Vol. 2.

[22] J. Park and S. Teller. 2014. *Motion Compatibility for Indoor Localization*. Technical Report MIT-CSAIL-TR-2014-017. http://hdl.handle.net/1721.1/89075

[23] D. Philipp, P. Baier, C. Dibak, F. Dürr, K. Rothermel, S. Becker, M. Peter, and D. Fritsch. 2014. MapGENIE: Grammar-Enhanced Indoor Map Construction From Crowd-Sourced Data. In *Proceedings of the 2014 IEEE International Conference on Pervasive Computing and*

*Communications (PerCom)*. 139–147.
[24] S. Pu and G. Vosselman. 2009. Knowledge Based Reconstruction of Building Models from Terrestrial Laser Scanning Data. *ISPRS Journal of Photogrammetry and Remote Sensing* 64, 6 (2009), 575–584.
[25] C. Qiu and M. Mutka. 2016. iFrame: Dynamic Indoor Map Construction through Automatic Mobile Sensing. *Pervasive and Mobile Computing* (2016).
[26] P. Robertson, M. Angermann, and B. Krach. 2009. Simultaneous Localization and Mapping for Pedestrians Using Only Foot-mounted Inertial Sensors. In *Proceedings of the 11th International Conference on Ubiquitous Computing (UbiComp '09)*. ACM, New York, NY, USA, 93–96. https://doi.org/10.1145/1620545.1620560
[27] A. Rottmann, C. Mozos, Ó.and Stachniss, and W. Burgard. 2005. Semantic Place Classification of Indoor Environments with Mobile Robots Using Boosting. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3 (AAAI'05)*. AAAI Press, 1306–1311. http://dl.acm.org/citation.cfm?id=1619499.1619543
[28] S. Rusinkiewicz and M. Levoy. 2000. QSplat: A Multiresolution Point Rendering System for Large Meshes. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 343–352. https://doi.org/10.1145/344779.344940
[29] R. Rusu and S. Cousins. 2011. 3D Is Here: Point Cloud Library (PCL). In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1–4.
[30] J. Schmidt and H. Niemann. 2001. Using Quaternions for Parametrizing 3-D Rotations in Unconstrained Nonlinear Optimization. In *Proceedings of the Vision Modeling and Visualization Conference (VMV '01)*. Aka GmbH, 399–406. http://dl.acm.org/citation.cfm?id=647260.718651
[31] Ruwen Schnabel and Reinhard Klein. 2006. Octree-Based Point-Cloud Compression. In *Proceedings of the Symposium on Point-Based Graphics*, M. Botsch and B. Chen (Eds.). Eurographics.
[32] P. Tang, D. Huber, B. Akinci, R. Lipman, and A. Lytle. 2010. Automatic Reconstruction of As-Built Building Information Models from Laser-Scanned Point Clouds: A Review of Related Techniques. *Automation in Construction* 19, 7 (2010), 829–843.
[33] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi. 2014. Community-Based Bayesian Aggregation Models for Crowdsourcing. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 155–164.
[34] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky. 2003. *Instant Architecture.* ACM, New York, NY, USA. 669–677 pages. https://doi.org/10.1145/1201775.882324
[35] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. 2013. Semantic Trajectories: Mobility Data Computation and Annotation. *ACM Transactions on Intelligent Systems and Technology* 4, 3, Article 49 (July 2013), 38 pages. https://doi.org/10.1145/2483669.2483682
[36] X. Zhang, Y. Jin, H. Tan, and W. Soh. 2014. Cimloc: A Crowdsourcing Indoor Digital Map Construction System for Localization. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*. IEEE, 1–6.
[37] B. Zhou, Q. Li, Q. Mao, W. Tu, X. Zhang, and L. Chen. 2015. ALIMC: Activity Landmark-Based Indoor Mapping via Crowdsourcing. *IEEE Transactions on Intelligent Transportation Systems* 16, 5 (Oct 2015), 2774–2785.

Received August 2017; revised November 2017; accepted January 2018