GraMap: QoS-Aware Indoor Mapping Through Crowd-Sensing Point Clouds with Grammar Support[*]

Mohamed Abdelaal, Frank Dürr, Kurt Rothermel
Institute of Parallel and Distributed Systems,
University of Stuttgart, Stuttgart, Germany
first.last@ipvs.uni-stuttgart.de

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

ABSTRACT

Recently, several approaches have been proposed to automatically model indoor environments. Most of such efforts principally rely on the crowd to sense data such as motion traces, images, and WiFi footprints. However, large datasets are usually required to derive precise indoor models which can negatively affect the energy efficiency of the mobile devices participating in the crowd-sensing system. Furthermore, the aforementioned data types are hardly suitable for deriving 3D indoor models. To overcome these challenges, we propose GraMap, a QoS-aware automatic indoor modeling approach through crowd-sensing 3D point clouds. GraMap exploits a recently-developed sensors fusion mechanism, namely Tango technology, to cooperatively collect point clouds from the crowd. Afterward, a set of backend servers extracts the required geometrical information to derive indoor models.

For the sake of improving the energy efficiency of the mobile devices, GraMap performs data quality assurance along with 3D data compression. Specifically, we propose a probabilistic quality model—implemented on the mobile devices—to ensure high-quality of the captured point clouds. In this manner, we conserve energy via sidestepping the repetition of sensing queries due to uploading low-quality point clouds. Nevertheless, the resultant indoor models may still suffer from incompleteness and inaccuracies. Therefore, GraMap leverages formal grammars which encode design-time knowledge, i.e. structural information about the building, to enhance the quality of the derived models. To demonstrate the effectiveness of GraMap, we implemented a crowd-sensing Android App to collect point clouds from volunteers. We show that GraMap derives highly-accurate models while reducing the energy costs of pre-processing and reporting the point clouds.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile devices**;

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

KEYWORDS

indoor mapping, crowd-sensing, energy efficiency, formal grammars

ACM Reference format:
Mohamed Abdelaal, Frank Dürr, Kurt Rothermel and Susanne Becker, Dieter Fritsch. 2017. GraMap: QoS-Aware Indoor Mapping Through Crowd-Sensing Point Clouds with Grammar Support. In *Proceedings of 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Melbourne, VIC, Australia, November 7–10, 2017 (MobiQuitous 2017),* 11 pages.
https://doi.org/10.1145/3144457.3144493

1 INTRODUCTION

Location-based services (LBS) play nowadays a significant role in our daily life. They broadly acquire real-time geo-data from mobile devices to provide information, entertainment or security. However, the provision of such services in indoor environments, e.g. emergency response, indoor navigation/routing in airports, universities, hotels, and other public buildings, is still limited due to the lack of precise and up-to-date indoor maps. The primary reason behind this scarcity emerges since manual mapping of indoor environments is often cumbersome and costly. Therefore, it is drastically favorable to automate the process of generating indoor maps.

Lately, several approaches for automatic indoor mapping have been proposed in the literature [1, 10, 16, 18]. The common feature among these efforts is the acquisition of mapping data, such as motion traces, images, WiFi footprints from a crowd of users. Although these approaches showed promising results, they primarily suffer from various shortcomings: (1) Large datasets are mostly required to construct precise indoor maps. For instance, Jigsaw [10] demands collecting between 100 and 150 images for each landmark to achieve precise mapping. (2) Important QoS metrics, such as energy overhead on the participating mobile devices as well as the sensing latency, have been broadly overlooked. (3) The collected data types are limited to the derivation of 2D maps. To tackle such challenges, we introduce GraMap as an indoor modeling method through crowd-sensing 3D point clouds in a participatory fashion. GraMap leverages a recently-developed sensor fusion mechanism, namely Tango technology, to cooperatively collect point clouds from mobile devices of normal individuals.

GraMap represents a two-step process of deriving indoor models: (1) deriving an *initial* model through applying several stages of clutter filtration on the collected point clouds and (2) augmenting accuracy of the initial model via integrating them with *indoor grammars*. In general, indoor grammars are powerful tools to encode structural information for different kind of architectural domains. Floor plans of public buildings usually follows certain architectural principles. For instance, the buildings are typically traversed by a set of hallways to ensure convenient access to the rooms. Such hallways divides each floor into a set of hallway areas and non-hallway areas. The latter can be further divided into a set of rooms or room units. In these cases, grammars can be simply adopted along with point clouds to derive a highly-accurate indoor model. To the best of our knowledge, GraMap is the first mapping method which employs formal grammars along with point clouds to generate highly-accurate indoor models.

Aside from modeling accuracy, GraMap also considers the energy costs of collecting point clouds from "resources-constrained" mobile devices. To this end, we employ Octree compression to reduce the transmission overhead. Nevertheless, energy can still be wasted if a sensing area is repeatedly queried due to receiving low-quality data collected by non-experts. In these cases, the sensing task typically takes longer time and the mobile devices waste energy of pre-processing and uploading several point clouds of the same area. To overcome such a problem, We first analysis the *quality* of point clouds in terms of several criteria including completeness and skewness. Based on this analysis, we introduce a *probabilistic quality model* (PQM) — implemented on the mobile devices — to assess the quality of the captured point clouds before further processing and radio transmission. Hence, point clouds can be entirely rejected if their quality measure is smaller than a quality margin. In this realm, it is important to mention that we are the first to consider several QoS metrics (i.e. accuracy, latency, and energy consumption) while deriving indoor models.

In detail, the paper provides the following contributions: (1) We define an architectural framework for modeling indoor maps using crowd-sensed 3D point clouds. (2) We present an approach that exploits structural knowledge encoded in formal grammars to compensate for inaccuracies due to collecting point clouds from the crowd. (3) We introduce a probabilistic quality model to avoid wasting energy due to uploading several low-quality point clouds of the same geographical region. (4) We provide an implementation of indoor grammars describing structural information about room layouts. In this regard, we employ a *Hidden Markov Model* (HMM) [19] along with the Viterbi algorithm to properly estimate the probable room layout. (5) We present a proof-of-concept implementation and evaluation of the proposed approach in a real-world scenario. To this end, we implemented an Android App to collect more than 107 point clouds, i.e. circa 2.5 GBytes of collected data. Our App improves accuracy of the collected data through integrating the main three features of Tango technology, namely *motion tracking*, *depth perception*, and *area learning* [2]. The results show that using our grammar-based approach, we obtain a significant improvement of the modeling accuracy relative to the initial models with a reasonable reduction in the consumed energy.

The remainder of this paper is organized as follows: Section 2 introduces the system model and briefly explains the various components of our proposed method. Section 3 explains the acquisition of point clouds from personal mobile devices along with presenting the probabilistic quality model. Section 4 describes the processing of point clouds to derive initial indoor models. In Section 5, we explain the exploitation of formal grammars to enhance accuracy of the initial models derived using point clouds. Section 6 presents our real-world performance evaluations in terms of modeling accuracy and energy consumption of the mobile devices. Finally, Section 7 discusses recent work in the realm of indoor mapping and crowd-sensing before Section 8 concludes the paper with an outlook on future work.

2 SYSTEM OVERVIEW

In this section, we introduce the system architecture along with our assumptions. The system consists of a set of mobile devices $\mathcal{M} = M_1, \cdots, M_n$ which upload their point clouds $\mathcal{P} = P_1, \cdots, P_s$ to a back-end server. A sensing query q in indoor modeling applications typically comprises: (1) the location of the area to be scanned l_q and (2) the minimum quality requirements T_q. Once a participating mobile device receives a sensing query, the user scans the queried space l_q. Figure 1 depicts the main processing steps required to extract precise indoor models. The *point cloud acquisition* component, implemented on the mobile devices, receives sensing queries from a crowd-sensing server to scan geographical areas. In this regard, we assume that the mobile devices $\mathcal{M}$ are aware of their location in the building. To reduce the energy overhead of repeatedly scanning an area, the point cloud acquisition component implements a probabilistic quality model to sidestep uploading low-quality data.

The *Initial modeling* component receives n point clouds from the participating mobile devices. Subsequently, it processes the point clouds to generate an initial indoor model. Specifically, it starts with downsampling the point clouds to reduce the noise level. Furthermore, performing downsampling significantly reduces the execution time of the subsequent steps. Afterward, GraMap employs *principal component analysis* (PCA) to filter out the points in accordance with their normals' angle (cf. Section 4). To infer the geometrical semantics, we partition the point clouds into a set of planar segments. Subsequently, an *initial indoor model* is obtained through projecting the 3D segments onto 2D segments, converting the segments into lines, and finally refining the line segments.

In fact, the collected point clouds $\mathcal{P}$ typically comprise duplicated objects, i.e. common walls of neighboring rooms. Furthermore, not all queried areas are perpetually accessible by the participating users. Besides, the inexact localization often results in generating less-accurate point clouds. These shortcomings hinder the direct derivation of highly-accurate indoor models from crowd-sensed data. To amend the initial model, the *Grammar-based modeling* component improves the modeling accuracy through exploiting indoor grammars. Indoor grammars typically comprise a set of terminal and non-terminal symbols which represent the rooms and hallways. A typical grammar comprises also a set of derivation rules for splitting the non-hallway areas into room sequence. Particularly, GraMap extracts geometrical information, e.g. room size, from the initial model and then it feeds this knowledge to the grammar for generating the room layout. In this paper, we assume that the grammar is generated from the floor plan of other buildings which have similar architectural style. Since grammar generation is beyond the scope of this paper, the implementation details of the grammar generator can be found in [5].

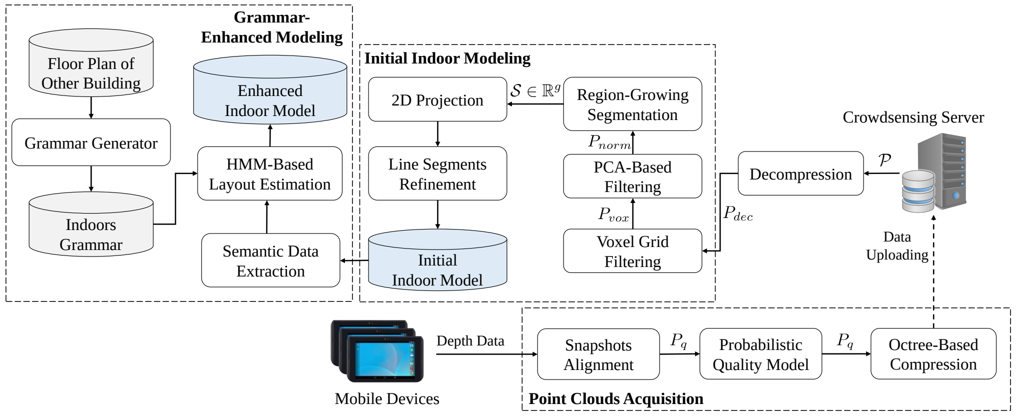


Figure 1: Various processing steps in GraMap

3 POINT CLOUDS ACQUISITION

In this section, we explain our approach for sampling point clouds from the depth sensors of Tango devices. Afterward, we discuss collecting point clouds from the crowd in an energy-efficient manner. In this regard, we apply a well-known 3D data compression method, namely *Octree compression*. We also introduce our probabilistic model for assessing the quality of the sampled point clouds before compressing and uploading them to the crowd-sensing servers.

3.1 Point Clouds Pre-Processing

To sample the point clouds, we exploit the main features of Tango devices, namely *motion tracking*, *area learning*, and *depth perception*. Tango devices are generally capable of tracking their own position and orientation in space, i.e. their pose, through fusing visual and inertial odometry [2]. Specifically, Tango devices estimate changes in their position via tracking features in a set of image frames collected using the motion tracking camera. Subsequently, Tango integrates these position changes with the rotation and acceleration changes coming from the inertial motion sensors. However, the motion traces — obtained by motion tracking — often suffer from accumulative drifts. To overcome such a technological obstacle, the collected frames are to be matched to unique reference frames. Through area learning, mobile users can store the key visual landmarks of a physical space in highly-compressed files, namely area description files (ADFs). These landmarks are then exploited as reference frames.

Generally, depth sensors rely on viewing infrared (IR) light to collect the depth information. Practically speaking, the sensors generate several snapshots while scanning a certain space, e.g. room or room unit. 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. In GraMap, we perform two steps to sample 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 so that all sampled point clouds are referenced 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. In the second step, we align the sampled snapshots to their correct orientation and position. 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 properly align the snapshots, we again exploit the motion traces — recorded while scanning the snapshots — to rotate and translate the snapshots into the same coordinate system [22].

To reduce the communication costs, the mobile devices upload a compressed version of the point clouds to the back-end server. Afterward, the required geometrical information are precisely extracted on the server-side. To this end, we utilize the octree-based compression which was mainly designed for 3D point clouds [23]. In octree compression, a sampled point cloud is spatially decomposed into an octree data structure. Literally, an octree is a tree-based data structure for managing 3D data where each internal node has exactly eight children. Subsequently, the decomposed data is quantized through replacing the points, present in each cell, by the cell centers of the octree's leaves. An additional compression step involves arithmetically encoding the remaining points through considering only cells whose child cells are nonempty. After receiving the compressed point clouds, the servers perform decompression before extracting the geometrical information. To further improve the energy efficiency, we below describe a probabilistic method to judge quality of the sampled point clouds.

3.2 Probabilistic Quality Model

Since point clouds are to be collected by non-experts, the collected data may suffer from low quality, i.e. being incomplete, inaccurate, and/or oblique. In this case, the crowd-sensing servers may need to repeatedly query for additional point clouds of the same area l_q. Consequently, the energy overhead for scanning a certain area can be significantly increased. As a way out of this limitation, we adopt a quality check point on the mobile devices while answering a given sensing query q. Once a mobile device m receives a sensing query q, the depth sensor generates a point cloud P_q of the targeted area l_q. Ahead of pre-processing and uploading the point cloud P_q to a corresponding server, the mobile device m checks various quality metrics. In this regard, we define an *acceptance probability* P_{ac} as a conditional probability of the depth sensor to generate a high-quality point cloud given a set of quality metrics $\mathcal{X} = \{x_1, x_2, \cdots\}$. If the probability P_{ac} is found to be low compared to the quality requirements T_q encoded in the sensing query, then the crowd-sensing App asks the user to rescan the area l_q. Thus, GraMap sidesteps the repetition of sensing queries which can be instantiated due to uploading low-quality point clouds.

In fact, there exist several factors which inevitably affect the quality of point clouds. First, the acquisition system and the measurement principle, i.e. time-of-flight, stereo matching, or structured light, strongly affect accuracy of the depth data. For instance, we demonstrated in [20] that point clouds generated from laser scanners are relatively more accurate than those acquired by Tango mobile devices. Nevertheless, we selected to utilize the mobile devices to share the load of collecting point clouds among several users. Simultaneously, GraMap compensates for this accuracy difference through imposing the Tango-based point clouds to additional filtering steps (cf. Section 4).

Second, the environmental conditions such as variations in ambient light, pressure, temperature or humidity have sometimes a non-negligible influence during the sampling process [26]. Since they rely on IR light, Tango devices are not mostly capable of collecting depth information in areas illuminated by light sources high in IR such as sunlight or incandescent bulbs. Furthermore, lighting may negatively affect the localization accuracy. According to [2], the sampled rooms may shift from their actual locations whenever the lighting conditions are dissimilar to that conditions while recording the area learning model. To avoid such drifts, GraMap utilizes several ADF models recorded at different time of the day. While answering a sensing query, a participating mobile device selects an ADF model whose recording time is relatively close to the moment of scanning the queried area l_q.

Third, characteristics of the observed scene in terms of object materials, surface reflectivity, and surface roughness significantly influence light reflection on the object surfaces. This implies that Tango devices can hardly collect depth information from objects that do not reflect IR light, e.g. glass walls and windows [2]. Fourth, scanning geometry — defined as the distance and orientation of scanned surfaces with respect to the involved depth sensor — highly impacts the density and accuracy of the collected points. Current Tango devices have typically an operating range between half a meter to four meters. Scanning an object at a distance outside this range leads either to collecting an insufficient number of points or to crashing the Tango core. Similarly, swift movements of the mobile devices can negatively contribute to the collected data quality.

Based on this analysis of factors impacting the quality of point clouds, we selected to monitor two distinct metrics, namely *incompleteness* and *skewness*. As aforementioned, point clouds are captured through aligning several snapshots together. Through our experiments, we found that the number of points in each snapshot is inversely proportional to the distance between the mobile device and the targeted surface. Additionally, the number of points per snapshot highly reduces when scanning a window, or a glass wall. To collect as many points in these scenarios, a user has to slowly scan these surfaces several times to cover as much details as possible. To express the incompleteness metric, we define a points threshold α_p where snapshots whose number of points n_s is smaller than the threshold α_p can be annotated as *incomplete* snapshots S_{in}. Throughout our experiments, we found that setting α_p to five thousand 3D points enables proper annotation of snapshots. Accordingly, we define the probability of a point cloud to be incomplete $P_{in} = \frac{|S_{in}|}{|S|}, \forall\, S_{in} : n_s \leq \alpha_p$ where $|S|$ is the total number of snapshots in the captured point cloud and $|S_{in}|$ is the number of incomplete snapshots.

Aside from incompleteness, skewness denotes the incorrect orientation of scanned objects as a result of tilting the mobile devices while scanning an area. In fact, it is hard to entirely sidestep skews originated from improper scanning. However, the quality model can reject severe cases in which tilting of the scanned objects is broadly intolerable. In this regard, GraMap exploits the pose information to estimate the Euler's rotation angle, namely the roll angle ψ (cf. Equation 1). Specifically, Tango devices provides the orientation data as a *quaternion* of the target frame with reference to the base frame [2]. These quaternions are typically defined in the form: $a + b \cdot \overrightarrow{i} + c \cdot \overrightarrow{j} + d \cdot \overrightarrow{k}$ where a, b, c, and d are real numbers while $\overrightarrow{i}$, $\overrightarrow{j}$, and $\overrightarrow{k}$ represent the standard orthogonal basis of the 3D space [6].

$$\psi = \text{atan2}(2(ab + cd), 1 - 2(b^2 + c^2)) \tag{1}$$

To infer skewness of captured objects due to improper handling of the mobile device, we map the roll angle ψ onto a random variable X_ψ which is normally-distributed, i.e. $X_\psi \sim \mathcal{N}(\mu, \sigma^2)$. It is reasonable to set the mean μ to 90° which reflects the correct handling of the mobile devices while scanning point clouds. A variance $\sigma^2 = 30°$ has been found convenient for detecting intolerable skews. Accordingly, skewness probability P_{sk} can be estimated as follows.

$$P_{sk} = 1 - P(X_\psi = \psi) \tag{2}$$

To denote the acceptance probability P_{ac}, we employ the inclusion–exclusion principle to sum up the various probabilities. In this regard, we introduce a random variable $X_a \in \{0, 1\}$, which reflects the case that a sampled point cloud P can be further pre-processed and uploaded ($X_a = 1$), or not ($X_a = 0$). Since our quality metrics are independent, we obtain

$$P_{sum} = P_{in}(P_q) + P_{sk}(P_q) - P_{in}(P_q) \times P_{sk}(P_q). \tag{3}$$

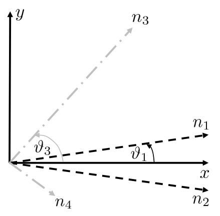
Consequently, the acceptance probability of P_q to be annotated as a high-quality point cloud can be computed according to Equation 4. In this regard, we consider the quality requirement T_q encoded in the sensing query q.

$$P_{ac}(X_a = 1 \mid \mathcal{X}) = \begin{cases} 1 & \text{if } P_{sum} \leq T_q \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

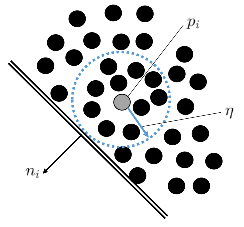
4 POINT CLOUDS-BASED MODELING

In this section, we elaborate on the initial modeling of indoor environments through crowd-sensed point clouds. As depicted in Figure 1, the aggregated point clouds are imposed to multi-stage filtering to compensate for possible inaccuracies. Let the octree decompression generates a point cloud $P_{dec} \in \mathbb{R}^m$ where m is the number of 3D points, i.e. $p_i = \{x_i, y_i, z_i\}, p_i \in P_{dec}$. We first perform downsampling and removal of clutter, i.e. any undesirable object present in the scanned point cloud P_{dec}. In general, the desired objects, e.g. walls, ceiling, floors, have usually superfluous points describing them. Hence, the *voxel grid filter* [21] is employed before further processing the point cloud P_{dec}. A voxel grid is principally a 3D grid map with fixed resolution, i.e. a set of tiny identical 3D boxes in space. The core idea behind voxel grid is to convert the continuous geometric representation into a set of voxels, i.e. $V = \{v_1, \cdots, v_i, \cdots, v_r\}$ where $r \ll m$, that best approximates the continuous representation. Downsampling is performed through approximating the points present in each 3D box v_i with their centroid $C := \frac{\sum_{i=1}^{\rho} p_i}{\rho}$ where ρ is the number of points in each voxel v_i. Hence, the resultant point cloud P_{vox} has a resolution of r, i.e. $P_{vox} \in \mathbb{R}^r$.

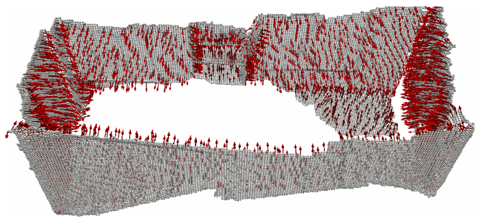
To derive an indoor map, we have to identify the walls location in the point cloud P_{vox}. To this end, we utilize the *normal* of each point to filter out the points describing other semantics, e.g. furniture. In geometry, a normal is a vector that is orthogonal to a given object. For points belong to a wall, their normals are ideally parallel to the ground plane $\vartheta = 0$ (cf. Figure 2a). Whereas, other objects may have normals pointing to any other direction. In fact, estimating the normal to a point on a surface is approximated by the problem of estimating the normal of a plane tangent to that surface. As depicted in Figure 2b, the 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 η comprises k neighboring points). In this realm, we employ the *principal component analysis* (PCA) method [11] for computing the surface *normal*.



(a) Normals angle relative to the ground plane



(b) Normal estimation at point p_i



(c) An example of normals estimation for a small room

Figure 2: PCA-based normals estimation

For each point $p_i \in P_{vox}$, the PCA method determines three eigenvector/eigenvalue pairs. An eigenvector $\overrightarrow{v}$ serves as a direction, while the corresponding eigenvalue λ denotes the variance in this direction. Formally, for each point $p_i \in P_{vox}$, a covariance matrix Σ is estimated for a set of neighboring points within the circle, as expressed in Equation 5 [21] where $\overline{p}$ is the 3D centroid of the neighbors. Generally, the covariance between two points p_i and p_{i+1} is a measure of the joint variability of these two points, i.e. quantifying their cross-correlation. Specifically, the eigenvalues are determined by solving $\det(\Sigma - \lambda \cdot I) = 0$ where det is the determinant operator and I is the identity matrix. Subsequently, the eigenvectors are determined through solving Equation 6 given the covariance matrix Σ and the eigenvalues λ.

$$\Sigma = \frac{1}{k} \sum_{i=1}^{k} (p_i - \overline{p}) \cdot (p_i - \overline{p})^T \tag{5}$$

$$\Sigma \cdot \overrightarrow{v_j} = \lambda_j \cdot \overrightarrow{v_j}, \quad j \in \{1, 2, 3\} \tag{6}$$

To illustrate the normals-based filtering method, Figure 2a shows four different normals $n_1, \cdots, n_4$. To compensate for possible drifts due to approximating the normals estimation method, we consider a small safety margin θ_{th} while classifying a point to be part of a wall. Accordingly, the normals n_1, n_2 are to be classified as wall points since their angles are smaller than the margin θ_{th}. Whereas, the other normals n_3 and n_4 are annotated as outliers. Figure 2c demonstrates an example of uniformly spatial-distributed normals for a point cloud P_{vox}. As it can be seen in the figure, for each point p_i there exists a normal vector, i.e. red arrow which declares the point's orientation. The number of points having normals in a direction parallel to the ground plane is a convenient criterion for filtering out the clutter. After filtering out the points whose normal angle is larger than the safety margin, i.e. $\vartheta > \theta_{th}$, the point cloud P_{vox} turns into a smaller cloud P_{norm}. Nevertheless, there may exist clutter whose surface is parallel to the walls, e.g. chairs body, screens, etc. Moreover, clutter may present in the corners between the walls.

Therefore, the next filtering step involves partitioning the point cloud P_{norm} into a set of planar segments $\mathcal{S} = \{S_1, \cdots, S_g\}$ where g is the number of segments. Afterward, we filter out segments according to their orientation, their point density, and their surface curvature. Literally, segmentation stands for the process of partitioning a point cloud into smaller and distinct segments, with the points in each segment S_i being in a semantic relation to each other. For our purpose of detecting the wall segments, the semantic relations are defined in terms of the points' locations in space, i.e. points are part of the same smooth surface, and the difference between normals angle of the neighboring points. Points which cannot be allocated to any segment are referred to as outliers and hence they are filtered out. GraMap employs the *region-growing segmentation* [15] to cluster the point clouds into planar segments.

As its name implies, the region-growing method selects a seed point p_s with normal angle θ_s and then it extends the region around p_s till meeting a stopping condition. To select an appropriate seeding point p_i, the method sorts the point cloud P_{norm} to find the point having the minimum curvature value. The intuition behind this selection is that growth from the flattest area typically decreases the number of segments [14]. The algorithm starts with finding the k-nearest neighbors of the seed point p_s. These neighboring points are added to the region whenever the difference between their normals and the angle θ_s is less than a certain threshold. The points whose curvature values are less than a threshold C_{th}, are to be added to the list of potential seed points. If the list of potential seeds is empty, the region is annotated as a complete segments and the algorithm is repeated for the rest of the point cloud.

So far, the point cloud P_{norm} is converted into a set of segments $\mathcal{S} \in \mathbb{R}^g$. Recalling that the point clouds are collected from the crowd, the uploaded point clouds sometimes suffer from skews which cause the segments to be tilted and rotated, instead of — in case of ideal scanning — being aligned in a flat vertical plane. To overcome these problems, we project the segments onto the x/y-plane. Afterward, we construct a bounding box for each segment. To this end, we make use of the eigenvectors e_1 and e_2 pointing in the direction of the segment's width and height, i.e. parallel to the segment's surface (cf. Figure 3). Specifically, the eigenvectors are utilized to rotate the segment around its centroid such that the eigenvectors align with the axes of the coordinate system. The four corner points of the bounding box then correspond to possible combinations of the minimum and maximum x- and y-coordinates of the segment's points. Finally, we inscribe a line segment (long dashed line in Figure 3) in the middle of the bounding box such that its end points are located in the middle of the segment's lateral edges.

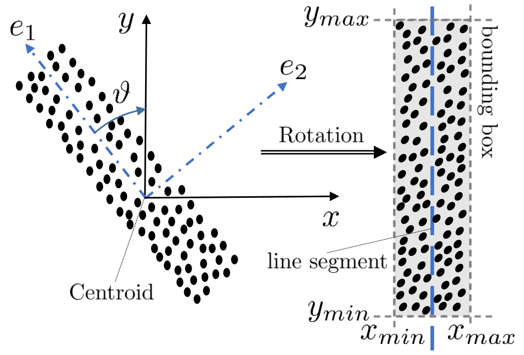


Figure 3: Transforming a segment of points into a line

The resultant model from the previous step, for some wall segments, may contain multiple disjointed lines describing the same wall. An example could be a line segment with a large window, where the wall segments above and below the window might have been split into separate segments by region growing. It is then likely, that their ventral positions differ and thus they also yield two individual wall sections. Moreover, some walls which separate neighboring rooms, are to be mostly scanned twice from both sides. To refine the line segments, we merge segments with similar features. Merging occurs, if (1) two segments s_i and s_j are oriented in the same direction (i.e. both horizontally or both vertically) and (2) the Euclidean distance d between the two line segments is less than a specified threshold. If these two conditions are met, an artificial line segment s_{merged} is constructed replacing the original segments s_i and s_j. After refining the line segments, the output model represents the indoor model derived from crowd-sensing point clouds. Despite the successive filtration and refining steps, the resultant model still suffers from repetitive line segments and several inaccurately localized wall segments. To further refine the indoor model, we exploit the structural knowledge embedded in formal indoor grammars. Below, we explain the formal grammars in more detail, before we explain their utilization for enhancing the indoor modeling process.

5 GRAMMAR-ENHANCED MODELING

In this section, we introduce indoor grammars as a tool for fitting the initial model derived by processing the crowd-sensed point clouds. We start with defining the indoor grammar and its components. Subsequently, we explain a *hidden Markov model* (HMM) method to derive the indoor model using the grammar rules and the initial model, obtained in the previous section.

5.1 Indoor Grammar

Before delving into the details of indoor grammars, we have first to introduce the idea behind formal grammars [17]. A grammar mainly consists of a set of *production rules* of the form $A \longrightarrow \alpha$ where A is a *non-terminal* symbol and α is a sequence of symbols. The production rules can be viewed as definitions of the non-terminal symbols at the left-hand side of the rules. There exists also a set of *terminal* symbols which do not appear on the left-hand side of any rule. The process of replacing non-terminal symbols by their definitions is referred to as *derivation* where a distinguished non-terminal symbol, called *Start S* is used to trigger the derivation process. Derivation is successively applied till obtaining a sequence which contains solely terminal symbols.

Grammars have been utilized in manifold applications for inferring higher-level semantics, data compression, and pattern matching [12, 13]. In this paper, we discuss the invocation of formal grammars to model the structural information of buildings. In general, buildings are usually traversed by a system of hallways to ensure convenient access to the rooms. Such hallways partitions each floor into hallway spaces and non-hallway spaces. Along each non-hallway space predominantly lies a linear sequence of rooms which are parallel to the adjacent hallway. The design of such buildings typically follows architectural principles and semantics relationships. For example, public buildings often comprise a limited set of room sizes. Such architectures can be precisely modeled by formal grammars.

Specifically, the structural information of non-hallway spaces can be described by a formal grammar of the form $G = (N, T, R, S)$, where N is the non-terminal symbols, $T = \{\epsilon, r_1, r_2, \cdots\}$ is the set of terminal symbols, R is a set of production rules, and $S = Space$ is the axiom. Each terminal symbol $r_i \in T$ constitutes a class i of rooms. Table 1 provides an example of a room grammar rule R_1^{room} which describes a room in terms of its width and its type. Although the indoor grammar encodes knowledge about the geometrical semantics, it lacks knowledge of the neighborhood of these semantics. Hence, we define two probability models for the grammar: (1) the *a priori* probability $P_a(r_i)$ principally stands for the relative frequency of occurrence of a room or room unit and (2) the *relationship* probability $P_r(r_j \mid r_i)$ is a conditional probability which models the relationship between rooms or room units. The room units are defined in a similar manner as sequence of rooms. Below, we explain our proposed method for enhancing the initial indoor model using the described indoor grammar.

5.2 Grammar-Based Model Fitting

As explained earlier, the collected point clouds are usually not complete and sometimes they do not include important details. The reason lies in collecting the point clouds on-the-fly while employing

Table 1: Examples of grammar rules representing rooms and room units

	R_1^{room}	R_5^{unit}
Rule	*Space* $\longrightarrow r_1$*Space*	*Space* $\longrightarrow r_3r_2r_3$*Space*
Width	2.4 m	19.2 m
A-priori	0.06	0.04
Type	small office	two executives with assistant's office

untrained individuals who lack the knowledge of the processing pipeline. As demonstrated in Figure 1, the initial indoor model is used as input to a subsequent processing stage for improving the modeling accuracy. To this end, we utilize the indoor grammar to fit the model so that the rooms are properly allocated. In particular, we perform two main steps: (1) extracting the room sizes from the initial model, and (2) employing an HMM method to derive a highly-accurate floor plan.

5.2.1 Walls Extraction. To explain how the split grammars can be used as a model fitting tool, we have to understand the methodology by which we apply the split grammar to derive the room layouts. Specifically, the production rules have to be applied along a given line, referred to as the reference line segment (RLS). These RLS lines $\mathcal{L} = L_1, \cdots, L_u$ are used as a reference while cascading the successive rooms $r_1, r_2, \cdots$. We exploit a set of reference lines, which are provided by the indoors grammar (cf. [5]). For each RLS line $L_k \in \mathcal{L}$, we iterate over the set of line segments — existing in the initial indoor model — and then search for branches. A branch represents a room's wall where the distance between two consecutive branches is used as the room's size. In order for a line segment to be recognized as a branch of the current RLS line, it has to satisfy two conditions: (1) being orthogonally oriented to the RLS line, and (2) intersecting with the RLS line or having an end point located in the vicinity of the RLS line. The branches which overshoot or intersect with an RLS line are recognized without problems. However, some potential branches may not be connected to an RLS line due to incomplete sampling data. Hence, we employ a distance threshold as a safety margin to avoid overlooking possible branches.

5.2.2 HMM-Based Model Derivation. Initially, deriving indoor models from the grammar G can be formulated as Markov chain Monte Carlo (MCMC) problem [24]. Specifically, the MCMC method generates several floor plans with random room sequences from some probability distributions. However, the MCMC method discards the fact that we have real observations that can be used while deriving the indoor model. In [16], we sequentially generate several hypotheses of the floor plan using a *constrained* random walk on a MCMC model. Afterward, a hypothesis — which minimizes the modeling error — is selected as the indoor map. Alternatively, GraMap formulates the problem of finding the probable room sequence for each RLS line as a *hidden Markov model* (HHM). The reason is that HMM models enable us to directly derive the indoor map rather than selecting one from a set of hypotheses. To this end, the HHM model exploits the extracted geometrical information while deriving the indoor map to adjust the probability distributions. Despite being relatively not accurate, the initial model, derived from point clouds is more rich of geometrical information than indoor models generated from motion traces. Hence, these information, e.g. room sizes, can be exploited to generate precise indoor models.

Generally, an HMM [19] is a statistical model in which the system being modeled is assumed to be a Markov process with unknown states, and the challenge is to determine the hidden states $\Psi = \psi_i \mid i = 1, 2 \cdots, n$ from a set of observables $O = o_j \mid j = 1, 2 \cdots, n$. Each *hidden state* ψ_i emits an observable, whose likelihood is given by a conditional emission probability $P_e(O_t \mid q_t = \psi_i)$ where q_t denotes the current hidden state. An HMM model $\Gamma := \langle \Psi, O, \pi, P_t, P_e \rangle$ also permits transitions among its hidden states, where π is the vector of initial state probabilities. These transitions are governed by a different set of likelihoods called transition probabilities $P_t(q_{t+1} = \psi_j \mid q_t = \psi_i)$. In our scenario, the rooms $r_1, r_2, r_3, \cdots, r_k$ represent the hidden states, while the extracted geometrical information from the initial model denote the observables (cf. Figure 4). The number of hidden states n in each RLS line L_k depends on the number of extracted observations. The initial probability vector π at $t = 0$ is set to the a priori probabilities, $\pi(r_i) = P_a(r_i)$. According to [24], the transition probability from rule r_i to rule r_j is given by

$$P_t(r_j \mid r_i) = \frac{P_a(r_j)}{P_a(r_i)} \cdot \frac{P_r(r_i \mid r_j)}{P_r(r_j \mid r_i)}. \tag{7}$$

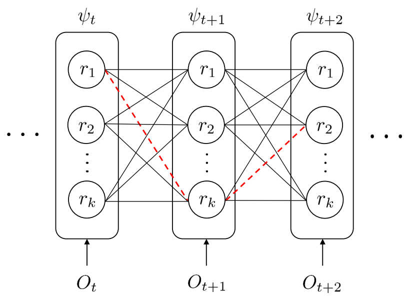


Figure 4: HMM trellis of modeling the grammar rules

The emission probability P_e captures the likelihood of observing a certain room size given the hidden states Ψ. To estimate the emission probabilities, we map each rule r_i onto a random variable X_{r_i} which is normally-distributed, i.e. $X_{r_i} \sim \mathcal{N}(\mu_i, \sigma^2)$. The mean μ_i is set to the width of the grammar rule, i.e. $\mu_i = W_i$, where a variance σ of one meter is found to be sufficient for properly inferring the rules. Consequently, the emission function $P_e(O_t | q_t = r_i)$ can be simply estimated from the probability density function of the rule random variable X_{r_i} as follows

$$P_e(O_t \mid q_t = \psi_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp[\frac{-1}{2\sigma^2} \cdot (O_t - W_i)^2]. \tag{8}$$

Figure 5 depicts the HMM model-based rule sequence inference. The sequence of observations is defined as the extracted room size from the initial indoor model. These semantics have to be filtered out before being applied to the HMM model (line 2). In particular, there may exist two neighboring line segments s_i, s_{i+1} representing the same wall, but they have been scanned from different sides. Accordingly, the Euclidean distance $\delta(s_i, s_{i+1})$ between these segments is deliberately discarded given that the distance δ is less than

```
Require: grammar rules $r_i \in T$, line segments from the initial model $\mathcal{S}$, RLS lines $\mathcal{L}$, threshold $\tau$
 1: for all RLS line $L_k \in \mathcal{L}$ do
 2:     $\mathbb{B} \leftarrow$ orthogonal lines $s_i \in \mathcal{S}$ with minimum distance to the RLS line $L_k$
 3:     for all line $b_i \in \mathbb{B}$ do
 4:         if $\delta(b_i, b_{i+1}) > \tau$ then
 5:             $O_t \leftarrow \delta(b_i, b_{i+1})$, where $O_t \in O$
 6:         else
 7:             skip the line $b_i$                               ▷ lines filtering
 8:         end if
 9:     end for
10:     compute the probabilities $P_e(O_t \mid r_i)$,   $r_i \in T$
11:     $\Gamma \leftarrow \Psi, O, \pi, P_t, P_e$                   ▷ HMM modeling
12:     for all $\psi_t \in \Psi$ do                           ▷ modified Viterbi
13:         for all $r_i \in \psi_t$ do
14:             for all $r_j \in \psi_{t-1}$ do
15:                 if $W_i^t + W_j^{t-1} \leq \overline{L}_k$ then
16:                     $\alpha_i^j = \eta_{t-1}(r_j) \cdot P_t(r_i \mid r_j) \cdot P_e(O_t \mid r_i)$
17:                 else
18:                     skip the path $r_j \rightarrow r_i$
19:                 end if
20:             end for
21:             $\eta_t(r_i) \leftarrow \max_j \left(\alpha_i^j\right)$
22:             $W_i^t \leftarrow W_i^t + W_j^{t-1}$
23:         end for
24:     end for
25:     estimate $\phi$ to get the optimal path                ▷ backtracking
26:     update the probabilities $\pi$ and $P_t$
27: end for
```

Figure 5: HMM-based rule sequence inference

a threshold τ (lines 3-9). After defining the HMM model, we can now find the best path (i.e. sequence of rules for each RLS line) which has the maximum joint probability (plotted as a red dashed line in Figure 4). To this end, we employ the Viterbi algorithm [9] to compute the best path through the HMM trellis.

The Viterbi algorithm utilizes dynamic programming to quickly find the path through the trellis that maximizes the product of the emission probabilities and transition probabilities. For each intermediate and terminating state $r_i \in \psi_t$, there exists a most probable path to that state, referred to as the *partial* best path. Accordingly, the probability of the partial best path to a state r_i is recursively obtained by

$$\eta_t(r_i) = \max_j \left(\eta_{t-1}(r_j) \cdot P_t(r_i \mid r_j) \cdot P_e(O_t \mid r_i)\right). \tag{9}$$

To find the overall best path, the Viterbi algorithm utilizes a back pointer ϕ to the predecessor that optimally provokes the current state, where $\phi_t(i) = \arg\max_j \left(\eta_{t-1}(j) \cdot P_t(r_i \mid r_j)\right)$. To simplify the process of searching the best sequence, we filter out paths whose accumulative length is greater than the length of the corresponding RLS line (lines 15-19).

6 PERFORMANCE EVALUATION

Through a real-world scenario, we examine the effectiveness of GraMap in deriving precise indoor models while reducing the energy overhead on the mobile devices due to participating in the crowd-sensing system. We first describe the setup of our evaluations. Subsequently, we discuss the obtained results.

6.1 System Setup

To collect point clouds, we designed an Android App to collect point clouds and to check their quality. We utilized Tango mobile devices which are equipped with 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 [2]. Data acquisition was mainly performed in two phases. In the *offline* phase, area learning is employed to record a number of ADF models, i.e. the visual landmarks of the entire floor. In the *online* phase, the ADF models were distributed to the mobile devices to register all collected point clouds relative to the same coordinate system. If a user is queried to scan a certain area l_q, the crowd-sensing App continuously captures several snapshots. Once scanning is finished, the App generates the point cloud P_q in xyz format via combining several scene snapshots and performing several transformations to properly align the snapshots.

At the server side, we integrated some C++ functions from the *point cloud library* (PCL) [21] in our implementation, such as region growing segmentation, Octree decompression, and geometric projection. Table 6 summarizes the parameters and their values which have been used throughout the evaluations. In our evaluations, we collected 107 point clouds, i.e. 2.5 GByte of depth data, from the second floor of the Computer Science building in Stuttgart. To profile the energy consumption of Tango devices, we logged the battery's current and voltage[1]. These current and voltage values associated with timestamps were recorded every 100 ms 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 were repeated ten times and the resultant values are then averaged.

System Parameter	Value	System Parameter	Value
incompleteness threshold α_p	5000 point	normals filtering angle θ_{th}	10°
curvature threshold C_{th}	1 m	voxel grid size	2 cm
neighbor count k	50	distance threshold τ	1 m

Figure 6: parameters used in the evaluations

6.2 Initial Model Generation

As earlier explained, an initial indoor model is extracted through projecting the collected point clouds to several filters (cf. Figure 1). Figure 7a demonstrates a merged point cloud resultant from scanning a quadrant of the floor plan. In fact, the floor has four identical quadrants, hence the results received from one quadrant can be simply applied to other quadrants. Accordingly, we selected to limit

[1]We used a software for logging sine we could not use hardware logging due to the voltage limits of the measuring equipment.

our evaluations to the quadrant which is easily accessible by the volunteers. Figure 7b depicts the merged point cloud after downsampling and normals-based filtering. Voxel grid downsampling is indeed a lightweight operation where it took circa 1.7 seconds to significantly reduce size of the merged point cloud (at least 59% reduction) at a grid size of 2 cm.

To remove clutter and keep only the wall segments, normals are exploited to filter out all objects whose normals angle is outside the range defined by the threshold $\theta_{th} = \pm 10^{\circ}$. Figure 7c depicts a cleaner version of the merged point cloud after region-growing segmentation. As the figure shows, different colors are used to identify the various wall segments. As explained earlier, segmenting the point clouds enables us to remove the points which do not belong to a valid segment. As observed during our experiments, smaller values of the neighborhood count k create more boundaries and thus yield smaller segments. In contrast, larger k values lead to more widespread and consecutive segments with reducing the algorithm's execution time. Nevertheless, larger k values may result in so-called *overgrowing* in which two perpendicular walls can be detected as a single segment. As a compromise, we empirically selected a moderate value, i.e. $k = 50$, which entirely avoids both small segments and overgrowing.

When using a grid size of 2 cm, adjusting the curvature did not yield notable improvements. Thus, the maximum curvature was set to 1. This implies that, during region growing, as soon as a neighbor gets added to the current region based on normals disparity, it will also be considered as an additional seed point. The final parameter of region growing is the minimum cluster size, which can be used to set a lower boundary for the minimum number of points a segment must contain for it to be considered a valid segment. Such a boundary is very useful to dismiss segments stemming from noise or clutter, i.e. small and/or undesired objects. Note that this value highly depends on the model's point density, which in turn depends foremost on the voxel grid size and on the amount of points filtered out by normals angle. Specifically, a minimum cluster size of 100 points was considered while generating the initial indoor model.

Figure 7d depicts the merged point cloud after projecting it onto the x-y plane. Afterward, the wall segments are converted into a set of line segments, as shown in Figure 7e. Finally, close and parallel lines segments are turned into a single line segment which identifies the wall location. Figure 7f demonstrates the initial model after applying the processing pipeline. As it can be seen in the figure, the initial model still suffers from incomplete parts and inaccurate room sizes. Accordingly, we use this initial model to derive another processing step which incorporates the indoor grammar as a model fitting tool.

6.3 Enhanced Model Generation

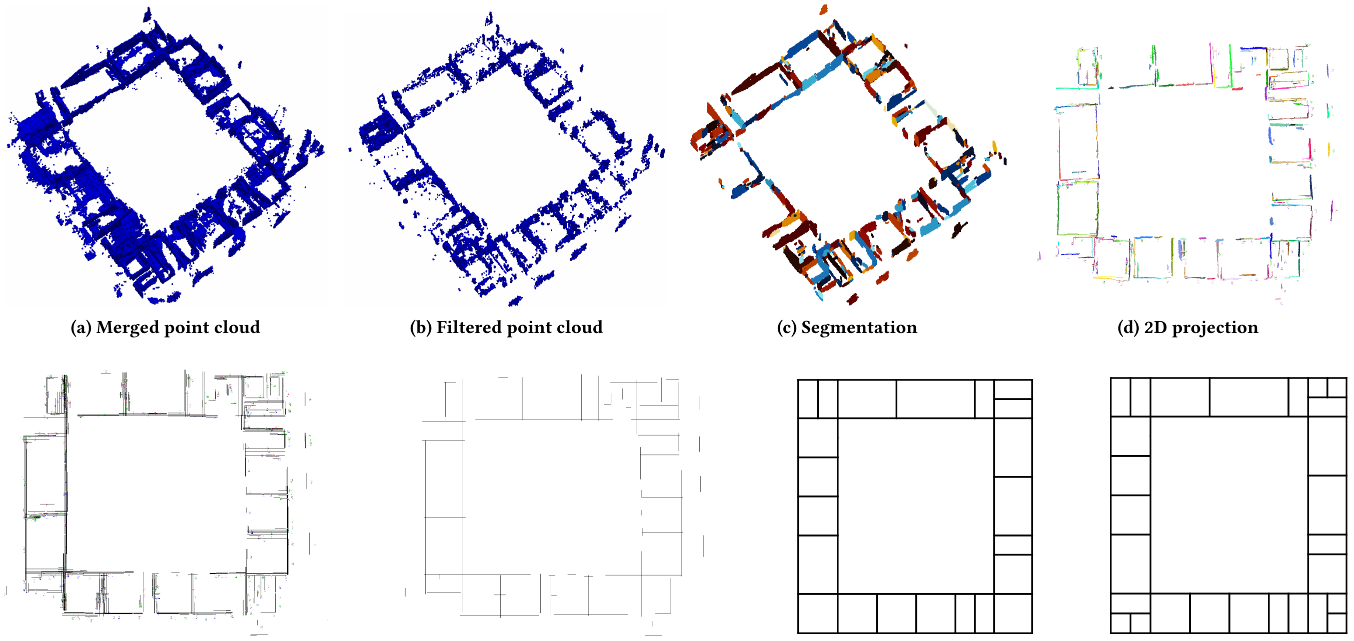
Figure 7g depicts the final indoor model after constructing a HMM model for each RLS line. By a quick comparison with the ground truth shown in Figure 7h, we found that the final model is highly accurate except at the corner areas. The reason is that our current version of the grammar does not support mapping of overlapping rooms, as is the case in the corners. In other words, the grammar considers a fixed depth for all rooms. Accordingly, a reasonable extension of this work involves the modification of the indoor grammar to support such overlaps. Comparing the initial and the final indoor model, we find that the indoor grammar filled in some missing spaces (cf. top-left corner of the quadrant) due to the lack of input data (inaccessible rooms).

6.4 QoS Metrics

In this section, we quantify the accuracy and energy overhead of GraMap. We employ two grammars for the room layout generation: (1) an *accurate* grammar derived from the floor plan of the 2nd floor, and (2) a *semi-accurate* grammar derived from the 1st floor of the same building. Although both floors are quite similar, there exist different room types, room units, and neighborhood relationships. The intuition is to demonstrate the performance of GraMap when using data obtained from a similar building. Figure 8a demonstrates the fraction of detected rooms for different numbers of sensing queries. In this set of experiments, we excluded the inaccessible maintenance rooms — which account for 19 % of rooms in the quadrant — since they cannot scanned by the volunteers. When considering only the collected point clouds ("No Grammar") the percent of detected rooms, i.e. identified via matching the wall segments to the ground truth, sharply increases with adding more sensing queries. For instance, 95% of rooms can be detected using only 20 point clouds. Nevertheless, we found the percentage goes beyond the maximum value (i.e. 100%) due to several false positives. The problem emerges from the displacement of neighboring walls due to localization errors. Accordingly, the in-between gaps between these walls are erroneously annotated as rooms (cf. bottom side of Figure 7f). On the contrary, both of the semi-accurate grammar and the accurate one detect less rooms (at least 86.3% for 22 sensing queries) due to the aforementioned overlapping problem at the corners. It is important to realize that both grammars sometimes fill the gaps due to missing input data. Whenever the RLS line contains a set of scanned rooms and an inaccessible room, the grammar can simply anticipates the missing room.

Figure 8b depicts the average error in the size of rooms versus the fraction of detected rooms for all indoor models built from 22 sensing queries. Although the "No Grammar" scenario detects more rooms (i.e. after suppressing the false positives) than the two other scenarios, the average error of rooms size is significantly increased while considering only point clouds (on average 0.64 m). As the figure shows, the "Accurate" and "Semi-Accurate" scenarios nearly achieve the same performance while avoiding false positives through filtering out all rooms whose size is less than one meter (cf. Figure 5). The "Accurate" grammar has slightly smaller size error (on average 0.16 m) than the "Semi-Accurate" scenario (on average 0.22 m). To sum up, adopting grammars highly improves detection accuracy of the room layout.

To quantify the energy overhead, we compared the energy of pre-processing and uploading point clouds to the crowd-sensing server while enabling and disabling the probabilistic quality model (PQM). We examined the amount of consumed power consumption of the participating mobile devices to upload 22 point clouds using different values of the quality margin T_q. For high values of the quality margin, the consumed power is found relatively close


(a) Merged point cloud (b) Filtered point cloud (c) Segmentation (d) 2D projection

(e) Line segments inscription (f) Initial indoor model (g) Grammar-based Model (h) Ground truth

Figure 7: Various steps toward deriving a highly-precise indoor model

to the naïve case in which the PQM model is disabled. When selecting small values of T_q, several point clouds are immediately rejected due to exceeding the quality margin. In the scanned quadrant, the probability P_{sum} ranged from 20% up to 60% (two-sided glass walls). For our next experiment, we adopted a quality margin of 0.3 to minimize the energy overhead. Figure 8c compares the power consumption with and without adopting the PQM model for different number of sensing queries. As the figure shows, enabling the PQM model reduces the power consumption (at least by 21%) compared to the naïve case. It is important to mention here that the energy gain can be significantly increased whenever the scanned regions have a large number of glass walls or windows.

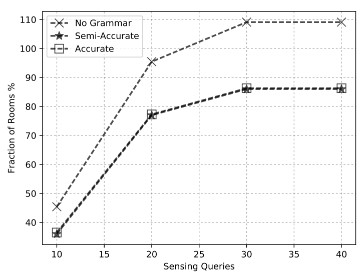
7 RELATED WORK

In this section, we discuss related work in the realm of indoor mapping, focusing on the automatic derivation of indoor models using crowd-sensing. Afterward, we present some examples of quality models in crowd-sensing applications and explain the various definitions of the quality metrics.

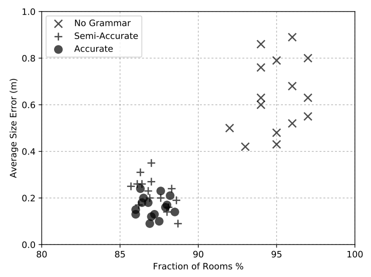
Indoor Mapping. The problem of automatically deriving indoor models has been tackled in several research works, such as MapGENIE [16], CrowdInside [1], JigSaw [10], and iFrame [18]. CrowdInside exploits the set of sensors in modern smartphones to collect motion traces from the crowd. Specifically, CrowdInside uses unique anchor points which are found in typical indoor spaces for motion error resetting. Similarly, iFrame uses personal devices to collect motion traces, Bluetooth, and WiFi footprints. To compensate for the shortcomings of each technology, iFrame adopts a matrix fusing mechanism. In this realm, we proposed in [16] MapGENIE, a framework for automatically deriving floor plans. MapGENIE relies on crowd-sensing 2D motion traces and it exploits the buildings exteriors to refine the collected traces. Furthermore, MapGENIE adopts formal grammars to increase the fraction of detected rooms. Although these approaches showed promising results, they often acquire large datasets which turns crowd-sensing into a cumbersome task. Moreover, energy efficiency of the mobile devices in such approaches were entirely overlooked. Finally, the collected data types, i.e. motion traces, WiFi footprints, pressure, are hardly sufficient for the derivation of 2.5D or 3D indoor models.

Quality Models. In the literature, several definitions of quality metrics were discussed. Wang et al. [25] define the quality of point clouds in terms of noise and outliers presence. Alternatively, Feng et al. [8] consider the spatial structure projection in coordinate planes and positioning accuracy as metrics for assessing the quality of point clouds. In fact, the quality models have been utilized for several purposes, such as improving accuracy of the collected point clouds and improving energy efficiency. For instance, Huang et al. [7] introduce a semi-supervised learning method to automatically assess the quality of indoor mapping data. Specifically, they employ feature extraction on a subset of the available data to obtain a set of optimal features. These features are later used to train the learning process while predicting the labels of unlabeled data. Weinmann et al. [26] present two approaches based on range reliability and local planarity for filtering the point clouds in the light of their quality measures.

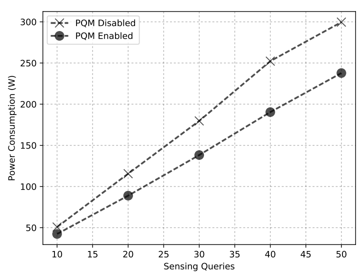
In terms of energy efficiency in crowd-sensing, we developed a quality model to probabilistically assess the sensing capabilities of mobile users while distributing the sensing queries [3]. The model



(a) Average fraction of detected rooms for various sensing queries



(b) Average error of the room size versus the fraction of detected rooms



(c) Power savings versus accumulated sensing queries

Figure 8: Evaluating the QoS metrics

defines the probability that a mobile device satisfies a given sensing query. In this case, the quality model assists in reducing the set of query recipients, which in turn conserves energy of other mobile devices. Later, we proposed in [4] a quality model to annotate the indoor environment as a set of accurately-modeled and inaccurately-modeled areas. Accordingly, energy consumption of the mobile devices can be reduced via deactivating their localization system when passing through an accurately mapped area. In GraMap, we alternatively implement quality models, on the mobile devices, to reduce the energy costs of processing and uploading several point clouds for each sensing query.

8 CONCLUSION & FUTURE WORK

In this paper, we introduced the GraMap approach that collects point clouds from a crowd of mobile devices for the sake of deriving precise indoor models. First, we presented the data acquisition phase including a probabilistic quality model to ensure the quality of the reported point clouds. Afterward, we introduced a processing pipeline for extracting an initial indoor model from the collected point clouds. Finally, these initial models are refined through adopting a formal grammar as a model fitting tool. The results show that GraMap significantly enhances the detection accuracy while reducing the overall energy overhead on the mobile devices. A logical extension of this work involves the refinement of our grammar to avoid the overlapping problem. Moreover, we seek to develop a 3D grammar which can be used to derive detailed 3D indoor models. To this end, we can use the current processing pipeline as the basis for our 3D indoor mapping system.

REFERENCES

[1] M. Alzantot and M. Youssef. 2012. CrowdInside: Automatic Construction of Indoor Floorplans. In *Proc. of the 20th Int. Conf. on Advances in Geographic Information Systems (SIGSPATIAL '12)*. ACM, 99–108.
[2] ATAP. 2016. Google Project Tango. (2016). https://developers.google.com/project-tango/ accessed on June 2016.
[3] P. Baier, F. Dürr, and K. Rothermel. 2013. Efficient Distribution of Sensing Queries in Public Sensing Systems. In *Proc. of the IEEE Inter. Conf. on Mobile Ad-Hoc and Sensor Systems*. 272–280.
[4] P. Baier, D. Philipp, F. Dürr, and K. Rothermel. 2014. *Quality-based Adaptive Positioning for Energy-Efficient Indoor Mapping*. Technical Report. IPVS Institute, Universität Stuttgart.
[5] S. Becker, M. Peter, D. Fritsch, D. Philipp, P. Baier, and C. Dibak. 2013. Combined Grammar for the Modeling of Building Interiors. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* II-4/W1 (2013), 1–6.
[6] J. Blanco. 2010. *A Tutorial on SE(3) Transformation Parameterizations and On-Manifold Optimization*. Technical Report 3. University of Malaga.
[7] H. Fangfang, W. Chenglu, L. Huan, C. Ming, W. Cheng, and L. Jonathan. 2016. Local Quality Assessment of Point Clouds for Indoor Mobile Mapping. *Neurocomputing* 196 (2016), 59–69.
[8] J. Feng, R. Zhong, Y. Yang, and Zhao W. 2008. Quality Evaluation of Spatial Point-Cloud Data Collected by Vehicle-Borne Laser Scanner. In *Proc. of the Int. Workshop on Geoscience and Remote Sensing*, Vol. 2. 320–323.
[9] G. Forney. 1973. The Viterbi Algorithm. *Proc. of the IEEE* 61, 3 (1973), 268–278.
[10] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li. 2014. Jigsaw: Indoor Floor Plan Reconstruction via Mobile Crowdsensing. In *Proc. of the Int. Conf. on Mobile Computing and Networking (MobiCom '14)*. ACM, 249–260.
[11] I. Jolliffe. 2002. *Principal Component Analysis*. Wiley Online Library.
[12] D. Lymberopoulos, A. Ogale, A. Savvides, and Y. Aloimonos. 2006. A Sensory Grammar for Inferring Behaviors in Sensor Networks. In *Proc. of the Int. Conf. on Information Processing in Sensor Networks (IPSN'06)*. ACM, 251–259.
[13] S. Maruyama, Y. Tanaka, H. Sakamoto, and M. Takeda. 2008. Context-Sensitive Grammar Transform: Compression and Pattern Matching. In *Proc. of the Int. Symp. on String Processing and Information Retrieval*. Springer, 27–38.
[14] N. Melkumyan. 2009. *Surface-Based Synthesis of 3D Maps for Outdoor Unstructured Environments*. Ph.D. Dissertation. Department of Aerospace, Mechanical and Mechatronics Engineering, The University of Sydney.
[15] A. Nguyen and B. Le. 2013. 3D Point Cloud Segmentation: A Survey. In *Proc. of the Conf.on Robotics, Automation and Mechatronics (RAM)*. IEEE, 225–230.
[16] 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 *Proc. of the IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*. 139–147.
[17] J. Power. 2002. Notes on Formal Language Theory and Parsing. *National University of Ireland, Maynooth, Kildare* (2002).
[18] C. Qiu and M. Mutka. 2016. iFrame: Dynamic Indoor Map Construction through Automatic Mobile Sensing. In *Proc. of the IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*. 1–9.
[19] L. Rabiner. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. of the IEEE* 77, 2 (Feb 1989), 257–286.
[20] L. Runceanu, S. Becker, N. Haala, and D. Fritsch. 2017. Indoor Point Cloud Segmentation for Automatic Object Interpretation. In *Proc. of the 37. Wissenschaftlich-Technische Jahrestagung der DGPF*.
[21] R. Rusu and S. Cousins. 2011. 3D Is Here: Point Cloud Library (PCL). In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 1–4.
[22] J. Schmidt and H. Niemann. 2001. Using Quaternions for Parametrizing 3-D Rotations in Unconstrained Nonlinear Optimization. In *Proc. of the Vision Modeling and Visualization Conference (VMV '01)*. Aka GmbH, 399–406.
[23] R. Schnabel and R. Klein. 2006. Octree-Based Point-Cloud Compression. In *Proc. of the Symp. on Point-Based Graphics*. Eurographics.
[24] J. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun. 2011. Metropolis Procedural Modeling. *ACM Trans. Graph.* 30, 2, Article 11 (April 2011), 14 pages.
[25] J. Wang, K. Xu, L. Liu, J. Cao, S. Liu, Z. Yu, and X. Gu. 2013. Consolidation of Low-quality Point Clouds from Outdoor Scenes. In *Proc. of the Computer Graphics Forum*, Vol. 32. Wiley Online Library, 207–216.
[26] M. Weinmann. 2016. *Preliminaries of 3D Point Cloud Processing*. Springer Int. Publishing, Cham, 17–38. https://doi.org/10.1007/978-3-319-29246-5_2