2018 IEEE 38th International Conference on Distributed Computing Systems

SnapTask: Towards Efficient Visual Crowdsourcing for Indoor Mapping

Marius Noreikis
Aalto University
Espoo, Finland
marius.noreikis@aalto.fi

Yu Xiao
Aalto University
Espoo, Finland
yu.xiao@aalto.fi

Jiyao Hu
Fudan University
Shanghai, China
jyhu1995@gmail.com

Yang Chen
Fudan University
Shanghai, China
chenyang@fudan.edu.cn

***Abstract*—Visual crowdsourcing (VCS) offers an inexpensive method to collect visual data for implementing tasks, such as 3D mapping and place detection, thanks to the prevalence of smartphone cameras. However, without proper guidance, participants may not always collect data from desired locations with a required Quality-of-Information (QoI). This often causes either a lack of data in certain areas, or extra overheads for processing unnecessary redundancy. In this work, we propose SnapTask, a participatory VCS system that aims at creating complete indoor maps by guiding participants to efficiently collect visual data of high QoI. It applies Structure-from-Motion (SfM) techniques to reconstruct 3D models of indoor environments, which are then converted into indoor maps. To increase coverage with minimal redundancy, SnapTask determines locations for the next data collection tasks by analyzing the coverage of the generated 3D model and the camera views of the collected images. In addition, it overcomes the limitations of SfM techniques by utilizing crowdsourced annotations to reconstruct featureless surfaces (e.g. glass walls) in the 3D model. According to a field test in a library, the indoor map generated by SnapTask successfully reconstructs 100% of the library walls and 98.12% of objects and traversal areas within the library. With the same amount of input data our design of guided data collection increases the map coverage by 20.72% and 34.45%, respectively, compared with unguided participatory and opportunistic VCS.**

I. Introduction

Visual Crowdsourcing (VCS) has become a popular paradigm of collecting photos and videos of interesting objects and views from widely available smartphone cameras [1]. It has been applied to obtain visual data for detecting cherry trees by the roads [2], reconstructing floor plans [3], [4], and providing information in emergency situations [5], just to name a few. However, the crowdsourced data do not always provide useful and accurate information. Therefore, proper guidance on data collection would be needed, in order to achieve high Quality-of-Information (QoI).

In this work we develop SnapTask, a VCS system that guides participants to collect visual data with high QoI in order to build complete indoor maps in an efficient manner. Our system reconstructs 3D models from crowdsourced images using Structure-from-Motion (SfM) [4], [6] techniques, and compiles indoor maps from the 3D models. In our case, QoI refers to the amount of useful information the data can contribute to the reconstruction of the indoor maps. In practice, it can be quantified by measuring the increase in the completeness of the maps.

In order to collect visual data with high QoI and to create complete indoor maps with minimal amount of data, there are three challenges to be solved. The first is the lack of data in some areas, as participants tend to move around *public hotspots* instead of performing a purely random movement [7], [8]. On the other hand, redundant data collected from public hotspots imposes extra processing costs, which becomes the second challenge. The third one comes from the limitations of the current computer vision techniques. For example, Structure-from-Motion (SfM) [4], [6] has been widely used for creating maps. An SfM algorithm can generate a 3D representation of a space from unordered 2D images. However, SfM algorithms fail in reconstructing featureless surfaces, such as glass walls and mirrors.

SnapTask solves the above-mentioned challenges in three steps. Firstly, it provides a novel task generation approach that identifies the areas of lacking data based on the analysis of model coverage. Secondly, it determines what kind of data to collect, either images for 3D reconstruction or the ones for reconstructing featureless surfaces. It then guides participants to collect the required data. Thirdly, it overcomes the issue of reconstructing featureless surfaces by utilizing the annotations provided by participants. In practice, it provides an online tool for participants to annotate the bounds of featureless surfaces, and imprints the surfaces with artificial and distinctive textures.

Through an evaluation in a library in Aalto University we show that SnapTask achieves 100% reconstruction of library walls and 98.12% reconstruction of obstacles and traversable areas inside the library. Furthermore, with same amount of input data, our approach of data collection outperforms unguided participatory VCS and opportunistic VCS by 20.72% and 34.45%, respectively, in terms of model coverage. We also show that the system achieves 98.14% precision and 90.23% F-score in reconstructing featureless surfaces.

In summary we make the following contributions:

1) We develop a guided participatory crowdsourcing system capable of collecting visual data with high QoI for building complete indoor maps.
2) We design task generation algorithms that issue tasks only in locations, from where the collected photos would contribute the most to the completeness of an indoor map.

2575-8411/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDCS.2018.00063

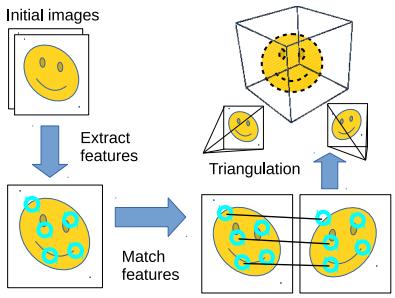


Fig. 1: Overview of a typical SfM pipeline

3) We invent a way to reconstruct featureless surfaces in SfM-based 3D models by employing crowdsourced annotations.
4) We evaluate the system with a field test in a library, and prove the effectiveness of our design.

The rest of the paper is organized as follows: Section II introduces SfM and different approaches of VCS. Section III and IV explain how we design and implement our system. Section V defines evaluation criteria and discusses the experimental results. Section VI reviews the related work before we conclude the paper in Section VII.

II. Background and Motivation

In this work we aim at developing an efficient VCS system for creating complete indoor maps. This section will introduce the SfM techniques for implementing 3D modeling, as well as the different ways to conduct VCS. Moreover, it will discuss the key challenges of building complete indoor maps from crowdsourced visual data.

A. Structure from Motion

SfM is a way of recovering a 3D world from 2D images [9]. There are several freely available SfM solutions. Examples include Bundler [10], VisualSfM [11] and OpenMVG [12]. A typical pipeline of SfM starts with visual feature extraction from images, and continues with matching them within image pairs and triangulating the matched 2D positions into 3D points using a multi view geometry (see Figure 1). Having enough photos of a particular place or an object, one can reconstruct a 3D representation of that object in terms of a point cloud or a 3D mesh. The output of the SfM pipeline includes a 3D point cloud and camera poses of the images used to build the 3D point cloud. The camera pose refers to a position and facing direction of a camera that took the photo. Throughout the paper we will refer to 3D point cloud and its corresponding camera poses as an *SfM-based 3D model* or a *3D model* in short. To calculate camera's field-of-view and its visibility coverage, a camera pose information is typically combined with a focal length from the photo EXIF[1] metadata. SfM models can be updated by adding additional photos as an input to an SfM pipeline. The new photos get registered alongside the already existing ones and typically increase the number of 3D points in the model. SfM-based 3D models have also been used for implementing image-based localization [13] and augmented reality based navigation [14].

Currently, SfM still faces a few important challenges. Firstly, it relies on an initial feature extraction stage, when visual features are found and extracted from 2D photos. Usually a feature is represented as a gradient change in an image, therefore, most features are extracted from texture-full surfaces, alongside corners and edges of visible objects and from other regions of the 2D image that has notable changes in color and texture. Afterwards, the extracted features are used as a basis for calculation of 3D points and structures. Therefore, objects made of glass, highly reflective or other featureless materials cannot be reconstructed using traditional SfM techniques, since state-of-the-art feature extractors detect no useful features there. Such objects include windows, glass doors, office walls, mirrors and any see-through items.

Secondly, SfM algorithms are highly compute intensive with an exponentially increasing processing time. Dong et al. [7] pointed out a high demand of computing resources as one of the main challenges for building crowdsouced SfM models. Moreover, Guo et al. argues that depending on an application, visual crowdsourced data can have a high degree of redundancy that leads to computing overhead [1]. Regarding SfM, the whole reconstruction pipeline highly relies on the number of extracted visual features. In order to reconstruct a single 3D point, there needs to be 2 or more images containing exactly the same visual features, thus covering the same parts of the view. Having even more photos of the same view yields more reconstructed 3D points. Considering our indoor mapping scenario, we only need to have enough 3D points to detect obstacles and enough extracted features to enable image based positioning [13].

B. Different approaches of Visual Crowdsourcing

Crowdsourcing in general can be classified into opportunistic and participatory crowdsourcing. In the first category, the data is sensed, collected and shared automatically without user intervention. For example, CrowdSense@Place [15] utilized opportunistically captured images and audio clips from smartphones to link place visits with place categories (e.g., store, restaurant). The data was collected automatically by a smartphone app whenever the device was exposed to the environment. In the second category, participants willingly collect visual data by accomplishing assigned data collection tasks. For example Chen et al. [16] developed a crowdsourcing system where users were asked to collect photos while walking along corridors and inside rooms.

Ma et al. [8] pointed out that opportunistic crowdsourcing needs many more participants than participatory crowdsourcing, as people tend to move around particular places and do not mimic arbitrary movement, which greatly reduces the probability of visually covering all aspects of a venue. In this work we propose to apply participatory VCS to collect data for image-based indoor mapping, and place our focus on the design of task generation.

[1]What is EXIF data? https://photographylife.com/what-is-exif-data

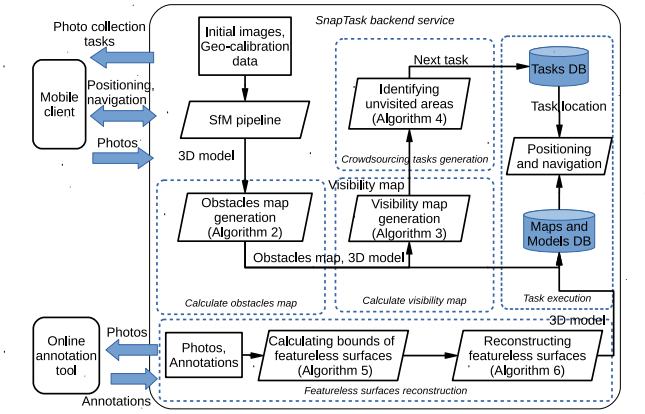
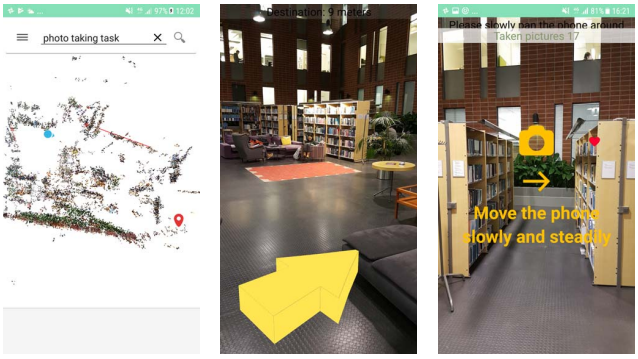


Fig. 2: SnapTask system overview.



(a) User's position (blue) and task location (red) on a map

(b) Guidance to the task location

(c) Guided collection of crowdsourced images

Fig. 3: Mobile VCS data collection application.

III. System Overview

As illustrated in Figure 2, our system consists of a mobile client, an online annotation tool and a backend server. In general, the mobile client is used for collecting images for building SfM models and photos for annotations, the online tool is used to annotate featureless surfaces, while the backend server generates instructions for data collection and processes the received data into indoor maps. The user scenario is described below.

Firstly, given an unknown space, an initial position (e.g. an entrance) is chosen and a set of images is collected at that position, including a set of images for geo-calibration. Geo-calibration is a process to automatically align SfM model to a real world coordinate system and is out of scope of this paper. Based on the collected images, an initial 3D model is created at the backend server.

Secondly, the backend server starts to generate tasks that guide users to collect images or annotations. The 3D model is expected to expand from the initial position, and gradually cover the whole space. Concerning the challenges of crowdsourced indoor mapping described in Section II, We identify 2 different tasks: to collect images and to annotate featureless surfaces.

Thirdly, when a participant requests a task, the location of the task will be sent to the mobile client and displayed on a 2D floor plan (Figure 3a). The floor plan is obtained by projecting a currently available 3D point cloud (e.g. the initial 3D model) onto a ground plane. If the participant confirms the task, the mobile client will receive navigation instructions from the backend server, and will guide the participant to the destination in an Augmented Reality (AR) mode (Figure 3b). We utilize our previous work for indoor positioning and navigation [14].

Fourthly, after arriving at the task location, the mobile client will guide users to conduct the task (Figure 3c). When the task starts, the phone starts taking pictures, as soon as it starts facing the premises at a perpendicular angle. The user is asked to slowly move around 360 degrees. Every 8 degrees the phone automatically captures an image. The phone simultaneously sends the captured images to a cloud server. Once the last image is received, the backend server starts data processing on the uploaded batch of photos. The annotation task is meant to aid reconstruction of featureless surfaces and consists of two parts. First, a user is asked to take photos that include the featureless surface. The photos are sent to an online annotation tool, where participants are asked to mark 4 points of the featureless surfaces on each of the photos. The photos and annotations are then sent to the backend server for processing. The second part of an annotation task does not necessarily need to be accomplished on-site. It can as well be outsourced through online crowdsourcing platforms, such as Amazon Mechanical Turk[2].

Finally, based on the processing results, SnapTask generates new data collection tasks. The loop continues until the system determines that the area is fully covered and no more tasks are sent to mobile clients. To implement the above-described user scenario, the backend server implements three major parts of functionality. 1) 3D model reconstruction, 2) task generation, 3) indoor positioning and AR navigation.

3D model reconstruction An indoor map is not only the required output of our system but is also utilized as an input for intermediate steps inside our system. We start construction of the map by building an SfM model from input photos and appending it to an already existing SfM model. For 3D reconstructions, we utilize a Multi View Geometry framework [12] which builds SfM models from arbitrary input images. We then utilize the SfM model to generate obstacles and visibility maps, and to compute a model coverage. We will present the aforementioned terms and algorithms to calculate them in detail in Section IV.

Task generation The goal of the system is to build maximum coverage indoor floor plans using VCS while minimizing data redundancy and preserving a high level of Quality-of-Information (QoI). Following this principle, the task gener-

[2]https://www.mturk.com/mturk/welcome

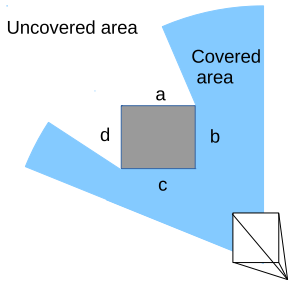


Fig. 4: Aspect coverage in VCS; Covered sides of an aspect *b* and *c*, uncovered sides - *a* and *d*.

ation algorithm must be able to identify the areas where more data is needed in order to increase the coverage of 3D reconstruction and the generated indoor map. The details of the task generation algorithm will be explained in Section IV.

Positioning and navigation With SfM-based 3D models, the system can identify user's current position based on an image taken from where the user is. The localization is implemented based on image feature matching. We reuse the AR navigation functionality developed in our previous work [14]. The module is responsible for providing 2D maps for a client application, serving localization queries and making sure that a client can quickly and easily reach an assigned task location.

IV. Algorithm Design and Implementation

In order to efficiently collect sufficient data for creating a complete indoor map, our system implements two sets of algorithms. The first algorithm set (Algorithm 1-4) is for identifying locations where more photos or annotations of featureless surfaces are needed. The other set (Algorithms 5-6) calculates the 3D coordinates of featureless surfaces from annotated photos. Before going into the details of these two algorithm sets, we first introduce important key terms.

Obstacles map is a 2D representation of non traversable areas (i.e. obstacles) in a venue. Algorithm 2 describes how we build an obstacles map from a 3D point cloud. The obstacles map is represented as a matrix where each cell of the matrix that contain a real world obstacle has a value greater than 0. Cells with values of 0 represent traversable areas.

Visibility map is a 2D representation of visible areas that have been covered by camera views of the photos used for reconstructing the 3D point cloud. Zhang et al. [17] distinguished two types of coverage in VCS: point coverage and aspect coverage. In order to fully cover a particular aspect, one has to take photos or videos that would cover all sides of that aspect (see Figure 4). Similar with the obstacles map, the visibility map is a matrix where cells with values greater than 0 represent areas covered by camera views. The value of a cell is equal to a number of cameras which fields-of-view cover that particular cell. In our work a matrix cell size is 15 cm, which maps the cell into a physical area of 15cm x 15cm. The size can be adjusted depending on a venue size and a required granularity - typically between 10cm and 50cm.

Algorithm 1 Generating Tasks

Input: Set of photos P, existing model M, current model coverage C, task location L
Output: New model M_f, obstacles map O, visibility map CV, set of tasks T

build an SfM model M_1 from P and M
$M_f \Leftarrow sorFilter(M_1)$
$O \Leftarrow$ **calculateObstaclesMap**(M_f)
$CV \Leftarrow$ **calculateVisibilityMap**(M_f, O)
$coverage \Leftarrow O \cup CV$
if $P \in M_f$ **and** $coverage > C$ **then**
  $areas \Leftarrow$ **findUnvisited**(O, CV, MAX_TASKS)
  if $size(areas) = 0$ **then**
    $T \Leftarrow \emptyset$
  else
    $T \Leftarrow setLocationNextTasks(areas)$
  end if
else
  $quality \Leftarrow checkPhotoQuality(P)$
  if $quality \leq LOW\_QUALITY$ **then**
    $T \Leftarrow generateTask(L)$
  else if $triedAtLocation(L) > TT$ **then**
    $T \Leftarrow generateAnnotationTask(L)$
  end if
end if
return M_f, O, CV, T

The coverage of the 3D point cloud, also called the **model coverage**, is the union of the coverage of the obstacles and the visibility maps (see Figure 10). Any particular place in a venue is considered as an **unvisited area**, if it is not included in neither the obstacles map nor the visibility map.

A. Task Generation Algorithms

Algorithm 2 **calculateObstaclesMap**

Input: 3D point cloud M
Output: Obstacles map O

$O \Leftarrow \emptyset$
compute OctoMap [18] Om from M
$Om' \Leftarrow$ merge Om cells along up-pointing axis
for $cell[i, j] \in Om'$ **do**
  if $cell \geq OBSTACLE\_THRESHOLD$ **then**
    $O[i, j] \Leftarrow cell$
  else
    $O[i, j] \Leftarrow 0$
  end if
end for
return O

Algorithm 1 describes the workflow of task generation. It consists of 4 steps.

Firstly, when a new set of photos are uploaded to the backend server, the server tries to register the new photos into the existing 3D point cloud. Each photo is expected to

Algorithm 3 calculateVisibilityMap

Input: 3D point cloud M, Obstacles map O
Output: visibility map all_fields

1: extract camera positions P and facing directions D from M
2: $all\_fields \Leftarrow empty\ matrix$
3: **for** $p \in P$, $d \in D$ **do**
4: $f \Leftarrow fov(p, d)$ //compute single camera coverage
5: $visible\_field \Leftarrow intersect(f, O)$ //update coverage regarding obstacles (see Figure 4)
6: $all\_fields \Leftarrow all\_fields + visible\_field$
7: **end for**
8: **return** all_fields

Algorithm 4 findUnvisited

Input: obstacles map O, camera views CV, number of areas to find N
Output: unvisited areas $found$

1: $found \Leftarrow 0, queue \Leftarrow set(), checked \Leftarrow Matrix$
2: $queue \Leftarrow$ initial position
3: **while** $queue \neq \emptyset$ **and** $size(found) < N$ **do**
4: $q \Leftarrow poll(queue)$
5: **if** q is not checked **then**
6: **if** $CV(q) < COVERED\_VIEW\_TOLERANCE$ **then**
7: $area \Leftarrow expand(node)$
8: $updateCheckedCells(area, checked)$
9: **if** $size(area) \geq MIN\_AREA\_SIZE$ **then**
10: $found \Leftarrow area$
11: **end if**
12: **end if**
13: add any empty adjacent cell positions to q
14: **end if**
15: **end while**
16: **return** $found$

contain regular EXIF metadata as well as a venue identifier. After building a model, we filter the SfM model with Statistical Outlier Filter [19] (line 2) to remove any outlier 3D points.

Secondly, the server updates the obstacles map and the visibility map based on Algorithm 2 and 3, respectively. $OBSTACLE\_THRESHOLD$ in Algorithm 2 refers to a minimum number of points inside of a cell of a merged OctoMap Om', for this cell to be considered as a part of an obstacle. In our work we set $OBSTACLE\_THRESHOLD = 4$.

Thirdly, if the model coverage increases, the backend server searches for the remaining unvisited areas based on Algorithm 4, and generates photo-taking tasks in selected unvisited areas. The maximum number of tasks to generate in each iteration is defined by the parameter MAX_TASKS. The maximum amount of possible tasks also depends on the size of the unvisited areas. Currently we generate 1 task at a time per participant. In Algorithm 4, we use a flood fill based algorithm. We start at a cell in a matrix and search for a closest

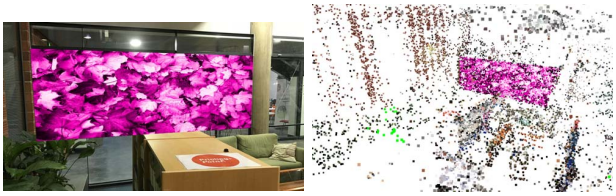


Fig. 5: a) an example of a featureless surface marked with an artificial texture. b) a view of a point cloud with an obstacle reconstructed using the artificial texture. Green dots indicate camera positions.

unvisited cell by recursively checking four neighbouring cells (up, down, left, right). We consider a cell unvisited if it does not contain any obstacles and is covered by less than $COVERED\_VIEW\_TOLERANCE$ camera views. In our system we set $COVERED\_VIEW\_TOLERANCE = 3$ since SfM pipeline that we use needs at least 3 observations of a same point to reconstruct it in 3D space. Once we find an unvisited cell, we recursively check unvisited neighbouring cells (line 7) until we find enough cells to cover an area defined by MIN_AREA_SIZE ($2.25m^2$ in our system). We take a center point of the discovered unvisited area and convert it to a 3D position. It is the position where a new task will take place.

Finally, if for some reason the new photos were not added to a model or the model coverage did not increase even after adding the photos, additional actions are taken. The backend server first examines the quality of the photos. It uses *variation of the Laplacian* [20] to calculate the blurriness of the photos, as blurry photos cannot be used for 3D reconstruction. High blurriness indicates poor quality input, when e.g. the camera was of a low quality or the worker did not manage to capture steady pictures. In this case the server simply assigns the same task to other participants. It may happen that the photos were of a high quality but still did not contribute in growing the 3D model. It commonly indicates that there are featureless surfaces which cannot be reconstructed by conventional SfM techniques. In this case the server generates an annotation task for the same location L, if the situation happened more than TT times (in our case $TT = 2$).

The output of Algorithm 1 includes an updated 3D point cloud, an obstacles map, a visibility map, and either a set of new crowdsourcing tasks or a notification that the venue is fully covered. The model and maps are stored in a database for further iterations.

B. Featureless surfaces reconstruction

As the current computer vision techniques cannot extract visual features from featureless surfaces, such as interior walls or glass panels, we propose to ask participants to manually label the bounds of featureless surfaces that appear in the photos. In practice, if Algorithm 1 generates an annotation task, the task will be completed in two steps. Firstly, photos that capture the featureless surfaces are taken from the given

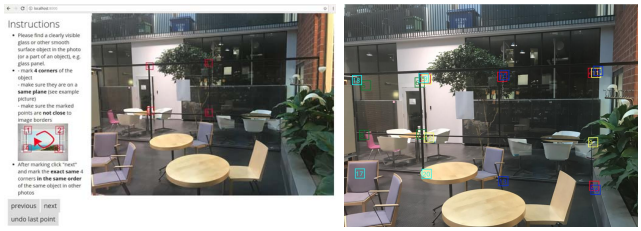


Fig. 6: a) SnapTask labelling tool. b) Different colors represent annotations of glass surfaces from multiple users. We observe that participants may not label the same objects in the same photo.

location. Secondly, the photos are passed to workers for manual annotation (see Figure 6a). Here are the instructions given by our online tool to the workers.

- Please find a closest glass or other smooth surface object in the photo.
- Mark 4 corners of the object, making sure they are on a same plane (see Figure 6a).
- Mark the exact same 4 corners of the object in other photos.

Algorithm 5 Get marked obstacle bounds

Input: photo sets P, a set of 2D annotations A
Output: obstacles bounds N

1: $N \Leftarrow \emptyset$
2: **for** $pSet \in P$ **do**
3: $C \Leftarrow$ find annotation $A[pSet[0]]$ center
4: $center\_clusters \Leftarrow cluster(C)$
5: **for** $photo \in pSet$ **do**
6: $obstacles \Leftarrow \emptyset$
7: **for** $center \in center\_clusters$ **do**
8: $\alpha \Leftarrow A[photo]$ corresponding to $center$
9: $obstacles[i] \Leftarrow \alpha$
10: **end for**
11: **for** $o \in obstacles$ **do**
12: $k\_sets = kmeans(o, 4)$ //using 4 clusters for 4 points
13: $corner\_points = cluster(k\_sets)$
14: $N[photo, o] \Leftarrow corner\_points$
15: **end for**
16: **end for**
17: **end for**
18: **return** N

After the photos are annotated, we obtain bounds of labelled obstacles N by using Algorithm 5. The participants may have labelled different obstacles and with variable precision (see Figure 6b), thus, we design our algorithm to robustly detect and combine annotations of objects inside images. In the algorithm we firstly obtain centres of annotations using DBSCAN [21] clustering to get distinct marked objects (lines 3-4). We then collect annotations of each distinct object from subsequent photos in the same set (lines 6-11). Finally, for each the object we have a set of marked 2D points. We use

Algorithm 6 Featureless surfaces reconstruction

Input: photos P, annotated obstacle bounds N, SfM model M, textures database DB
Output: updated SfM model M'

1: **for** $photo \in P, i \in [1..size(P)]$ **do**
2: **for** $obstacle \in N[photo]$ **do**
3: $T \Leftarrow DB[i]$
4: $b \Leftarrow N[photo, obstacle])$ //bounds of the obstacle
5: $photo \Leftarrow projectTextureToPhoto(T, photo, b)$
6: **end for**
7: **end for**
8: $M' \Leftarrow runSfMReconstruction(M, P)$
9: **return** M'

k-means algorithm [22] to find 4 sets of points and utilize DBSCAN to pinpoint actual locations of every point (line 14).

We then choose unique distinctive textures from an artificial texture database to imprint on annotated images and execute SfM reconstruction with a generated image set (Algorithm 6). We use imagemagick[3] to project a generated 2D image on each marked photo. Since now the glass area contains enough features, the annotated area gets reconstructed and any possible reflections of that photo are removed from the model (see Figure 5). Since we use distinctive colors, it is easy to locate the artificial points later on in a model, in case they need to be analyzed separately.

As an alternative approach, it is also possible to obtain 3D coordinates of the labels based on triangulation. However, it requires that the labelled photos have been registered into the SfM model beforehand. As the photos may contain reflective surfaces and the reflections are seen as blurry objects, the 3D reconstruction process would highly likely fail. Therefore, we decide to render an artificial distinctive texture on the annotated photos before passing it to the SfM pipeline.

V. Evaluation

We tested SnapTask with 10 participants in a library in Aalto University, and evaluated how efficiently our system can guide participants to collect enough data for creating a complete indoor map. In the experiments, participants received tasks generated automatically by the backend server, and used our mobile client for taking photos for reconstruction and annotations. For a comparison, we also collected data in opportunistic and unguided participatory manner, and evaluated the achieved coverage and the amount of photos used.

In this section we will first introduce our experimental setup and measurement metrics, and then describe the datasets and the experimental results.

A. Experimental Setup

The library where we tested our system is an arbitrarily shaped space that includes bookshelves, computer workstations, sofas, etc. (see Figure 7). The size of the area is around

[3]https://www.imagemagick.org/script/index.php

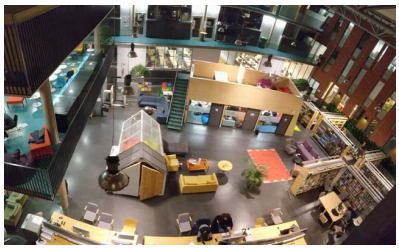


Fig. 7: Panoramic shot of a library in Aalto university where we evaluated SnapTask.

Fig. 8: Paths of the participants who have carried out opportunistic sensing tasks.

$350m^2$. Two outer walls of the library are made of bricks, while the other two are made of large transparent glass panels.

To bootstrap our system we shot a 2-minutes video near the entrance, and collected 39 photos for geo-calibration. From the video we extracted 46 frames, added 39 geo-calibration photos and created an initial model (see top left image in Figure 10). Afterwards, the system can generate crowdsourcing tasks with particular locations inside the library.

To update the initial model, we collected photos from the library in 3 different ways (described in Section V-B), and compare how the coverage of the generated maps increases with the amount of collected photos. More specifically, we measure the coverage of the reconstructed outer bounds and visible areas that include obstacles and traversable paths within the bounds. Here, obstacles refer not only to walls but also sofas, desks, tables, and other pieces of furniture that determine the traversal areas in the venue. Chairs and other movable objects are not considered, since their locations often change and do not necessarily block the traversable paths. Regarding a complete visibility of an area, it is required that all aspects of the area are covered by camera views (see Figure 4).

We used a laser range finder to obtain ground truth measurements inside the library. We firstly calculated the total length of the outer bounds which was 98.89 meters. Note that we have excluded the length of the entrance, since the photos for creating initial 3D model were collected at the entrance door and the entrance was already included in the initial model. Secondly, we obtained ground truth visibility and obstacles maps (see Figure 12d) by measuring obstacles' sizes and distances between them.

B. Datasets

We conducted 3 experiments in the library to compare our guided participatory crowdsourcing to opportunistic crowdsourcing and unguided participatory crowdsourcing.

We used the following smartphones for experiments and data collection inside the library: *Samsung Galaxy S7* and *Apple iPhone 7* for opportunistic and unguided participatory crowdsourcing, and *Samsung Galaxy S7* and *LG Nexus 5* for guided participatory crowdsourcing. Regarding the annotation tasks, we developed an online tool for labelling featureless surfaces following the instructions described in Section V-C3.

1) Opportunistic crowdsourcing: We envision that in a near future more people will use smart wearable devices that can easily capture images of what a person sees, e.g. smart glasses. Such devices would be perfect tools for opportunistic VCS. For our current experiment, however, we asked participants to carry a smartphone in front of them – mocking a smart wearable device – that was taking a video of the surroundings. We have asked 10 participants to carry out their daily activities in the library, e.g. going to a meeting room, finding a book, accessing a local workstation, and collected visual data while they were walking through the library.

We collected 20 videos along the participants' walking paths of a total length of 369 seconds. The paths are show in Figure 8. Blue icons indicate camera positions of extracted frames that were eventually used for model reconstruction. We used a sliding window frame extraction approach, where we select only a sharpest frame in that window, to prevent blurry samples from being added to the dataset. For this experiment we defined the window size of 30 and extracted 700 sharpest video frames for model reconstruction.

2) Unguided participatory sensing: For the second dataset, we asked each of the 10 participants to capture 100 photos inside a library. None of the participants were experts in computer vision and were taking arbitrary photos in the venue. After obtaining the photos, we filtered out blurry ones with *variation of the Laplacian* [20], since this task can be done automatically. We used 903 sharpest images the dataset.

3) Guided participatory sensing: We utilized the mobile application to collect data at the locations defined by the task generation algorithms.

Our system generated in total 17 crowdsourcing tasks at different positions. 11 tasks to collect photos and 6 tasks to annotate bounds of featureless surfaces. During 11 photo collection tasks we collected 633 photos that were used for SfM reconstructions. Figure 9 shows locations and sequence of all the generated tasks. A red circle indicates system generated task position, while a blue cross shows a mean center camera position of a corresponding collected photo set. The mean center position is calculated after the photos were added to the SfM model.

There is an offset between system generated locations and the actual locations where the photos were taken. It is because

Fig. 9: A generated point cloud and positions of the generated crowdsourcing tasks marked on a library floor plan. Red circles show positions of generated photo collection tasks, while blue "X" denotes where the actual photo capture took place. Numbers indicate a sequence in which the tasks were completed. Green diamonds show positions of tasks where participants collected photos for featureless surfaces annotation.

of two reasons. 1) the system is only aware of an already reconstructed area. It generates a task inside an unknown area, close to the last mapped obstacles. It may happen that the location is inside an actual undiscovered obstacle. 2) the user reaches task location using our indoor positioning system that has up to 1 meter positioning error [14]. However, the offset does not notably influence the final result. In case a location is inside an obstacle, human workers then simply start a task as close to that place as possible. After the model is updated, the system updates its knowledge with newly generated obstacles and takes them into account while generating a next task. In a worst case, if a worker collects photos at a totally different location, the system will generate a new task to collect data at the same location again, until that location is fully covered.

4) Featureless surfaces reconstruction: As mentioned before, our system generated 6 annotation tasks. For each task location, a participant collected T photos that were facing the featureless surface. In this experiment we set $T = 4$. We have uploaded 6 sets of photos to the online annotation tool and asked 15 other participants to label them. We have collected in total 360 annotations.

C. Evaluation results

1) Indoor map reconstruction: To analyze QoI of indoor maps generated by opportunistic and unguided participatory crowdsourcing, we divided corresponding photo sets into 7 parts, 100 photos each. We then took our initial model and added parts of the photo sets one by one to generate evaluation datasets S_i $i = 1..7$. For each dataset S_i we built an SfM model and calculated map coverage (See Algorithm 1 lines 1-5). We compared the coverage by directly comparing non-zero cells of obstacles and visibility matrices of the generated map to cells of corresponding matrices obtained from the ground truth floor plan. We did not consider any cells that were outside the ground truth coverage map (white area in Figure 12d). We also measured the length of reconstructed outer bounds of the venue in every obstacles map and compared it to the ground truth. During the comparison, we set the bounds reconstruction threshold to $T = 0.15m$, meaning that two segments of the bounds will be considered as one, if a distance between them is less than T. Regarding our guided participatory sensing approach, we can analyze model coverage and compare it to a ground truth after each completed photo collection task. Thus, we have 11 datasets for the guided case that we compared to the ground truth map.

We first show how our system incrementally generated the library indoor map. Figure 10 shows how the library model improved after each photo set, in terms of obstacles map and visibility maps. Note that we show here the output after each of the photo collection task. Featureless surfaces annotation tasks are not visualised separately – a more detailed evaluation of such tasks is presented in Section V-C3). The figure clearly indicates that after each photo collection task the system was able to generate floor plans with a higher coverage. We also recognise, that all the area is roughly covered and surrounded by bounding obstacles - outer walls of the library. Our final floor plan reconstruction indicates 98.12% map coverage and it successfully reconstructs 100% of the outer boundaries.

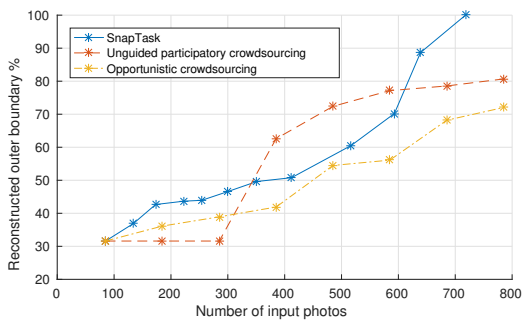
2) Comparison with other crowdsourcing approaches: Figure 11 indicates how the different aspects of generated maps improved with an increase of the amount of input photos.

Figure 11a shows how the length of reconstructed outer bounds changed with an increasing amount of input photos. In case of the opportunistic sensing, the length of reconstructed outer bounds increased steadily but only reached 72.04% of the total bounds. Participatory sensing approach performed better, achieving 80.69%, however after more than 500 photos the length of reconstructed bounds did not improve notably. SnapTask, on the other hand, was able to achieve 100% of outer bounds reconstruction using 11 photo sets and in total 633 images.

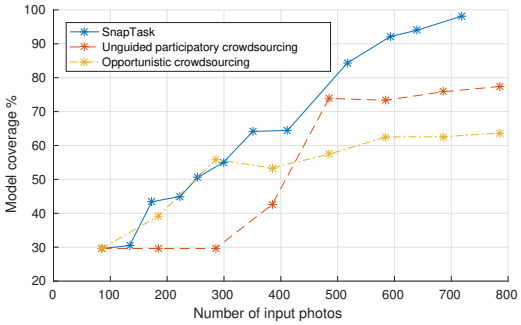
Figure 11b represents how model coverage changed with additional crowdsourced photos. It indicates a similar trend as in outer bounds reconstruction. Opportunistic approach reaches at its peak only 63.67% of coverage. Unguided participatory crowdsourcing approach converges at around 500 images, reaching at most 77.4% coverage. Additionally, for this approach we observe a high sudden increase of coverage after 3-4 photo sets were added. It is because random photos are sparse and SfM cannot reconstruct an object that is not covered by at least 3 different camera views. This did not happen with the opportunistic method data set, because subsequent frames from a video usually cover similar views. Our guided participatory approach expands area coverage gradually until it reaches 98.12% of coverage.

SnapTask did not reach 100% coverage in this case due to two reasons. In a few obstacles, we can observe certain white "empty" areas. This is because the SfM model from which the obstacles map was constructed had very sparse points inside a few obstacles, e.g. featureless parts of a table. Other white areas show spots that were too small for the algorithm to consider generating a new task there (we currently set the

Fig. 10: From left to right and from top to bottom: growth of visibility (green) and obstacles (black) maps after each photo collection task. Results of featureless surfaces annotation tasks are included but not distinguished separately. Final coverage map is presented in Figure 12c.

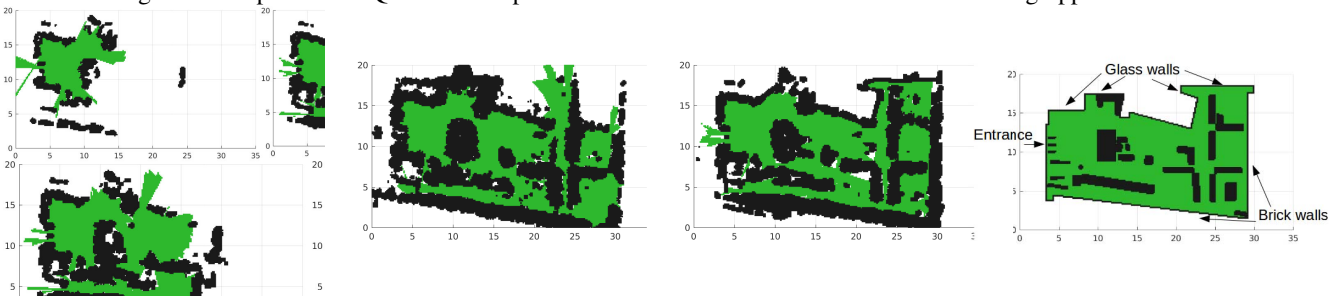


(a) Length of generated outer bounds



(b) Model coverage (obstacles and visibility maps combined) for different amounts of photos

Fig. 11: Comparison of QoI for floor plan reconstruction with 3 different crowdsourcing approaches.



(a) Opportunistic crowdsourcing (b) Unguided participatory crowdsourcing (c) Guided participatory crowdsourcing (SnapTask) (d) Ground truth map

Fig. 12: Comparison of final visibility (green) and obstacles (black) maps obtained by different crowdsourcing approaches and a ground truth.

minimum considerable area size of $2.25m^2$). Having smaller value would yield higher coverage rates, however, this would increase the number of tasks and collected photos.

As we see, unguided participatory sensing, performs well only when enough photos are collected. Moreover, when the new obstacles were reconstructed, the model and bounds coverage plateaus quickly and any additional photos do not add notable QoI.

To better understand the results, in Figure 12 we visualized the final floor plans obtained using the 3 different methods. It is clearly visible that some obstacles, especially parts of the outer wall are missing in Figures 12a and 12b. In case of the opportunistic sensing, participants did not visit certain areas of the library, and others were observed only for a short fraction of time (resulting in a very few extracted images for those areas). Regarding the participatory sensing, most of the areas were visited, however, a room in a top right corner was visited by very few participants, thus there were not enough photos to build the SfM model there. Another problem that both approaches faced was a missing glass wall (see Figure 12d). Bounds along some of the glass wall panels were reconstructed, because they either had posters, signs or pieces of furniture close to them. However, only our guided approach was able to pinpoint the missing glass wall locations and by utilizing the featureless surfaces annotation tasks complete the wall boundary there.

3) Featureless surfaces reconstruction: We evaluated how well our system can reconstruct featureless surfaces. For the

TABLE I: Analysis of Featureless Surfaces Reconstruction

Task #	Identified surfaces	Reconstructed surfaces	Precision	Recall	F-score
1	2	2	1.00	1.00	1.00
2	3	2	1.00	0.90	0.95
3	2	2	1.00	0.64	0.78
4	2	1	1.00	1.00	1.00
5	3	2	0.93	0.80	0.86
6	2	1	0.96	0.73	0.83

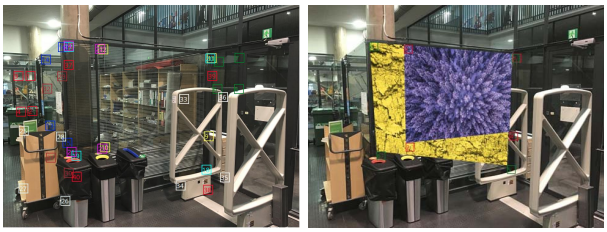


Fig. 13: Left: Initial annotations marked by workers. Right: Annotations after filtering and applying distinctive textures.

analysis of annotations collected from the featureless surfaces reconstruction tasks, we calculated how many surfaces were identified and how many of them were successfully reconstructed with SfM. We also measured ground truth lengths of featureless obstacles visible in the photosets. We then compared their lengths with lengths of objects reconstructed from annotated images.

We made sure, that SnapTask can indeed detect the featureless surfaces. While carrying out the first part of the evaluation, the system generated 6 tasks to annotate bounds of featureless surfaces. As Figure 9 shows, the tasks 1 and 3-6 were generated near glass walls of the library, while task 2 was generated near a featureless wall of a meeting room.

Table I summarizes results of featureless surfaces reconstruction tasks. Column "Identified surfaces" indicates how many reliably annotated surfaces we extracted from all (in our case from 15) annotations of a particular image set. Figure 13 shows how *SnapTask* identified reliable annotations and projected artificial textures before passing the images to an SfM reconstruction step. However, a few annotated surfaces were not reconstructed by SfM, thus "reconstructed surfaces" column shows how many of identified surfaces were successfully reconstructed. Precision, recall and F-score illustrates how well and how much of the ground truth wall did the annotated obstacles cover. SnapTask achieved on average 98.14% precision and 90.23% F-score in reconstructing featureless surfaces.

Only in cases 3 and 6 the recall was lower. It is because a featureless surface either stretched through a whole image width, or the object corners to be labelled were far away from the location where the photo was taken. However, large enough portions of obstacles were recovered to fully reconstruct a wall in an obstacles map (see Figure 12c).

VI. Related work

In this section we review the previous works related to spatial crowdsourcing, and VCS-based indoor mapping.

Spatial Crowdsourcing Spatial crowdsourcing refers to the new paradigm of data collection where participants need to collect data from specific locations [23]. In case of SnapTask, the task description includes the locations where participants are expected to shoot photos. Our focus is placed on determining task locations. This differs from previous works which mainly focused on selecting participants for spatial tasks. For example, Zhang et al.[24] and Song et al.[25] selected participants based on their current positions, in order to minimize incentive budgets while improving the QoI. Cheng et al. [26] extensively studied time constraints for spatial tasks. Our work complements previous works, as the participant selection mechanisms can be applied after task locations are calculated by SnapTask.

VCS-based Indoor Mapping Adoption of VCS for indoor mapping has been extensively studied in recent years. Jigsaw [6] utilizes crowdsourced images to build SfM models and derive obstacle bounds with vanishing lines. The researchers proposed two crowdsourcing tasks based on inertial sensors to measure distances between points of interest. ClickLoc [3] extends the idea of fusing visual and inertial data and further takes into account distances between two consecutive images to reconstruct indoor spaces more accurately. Crowd Map [4] utilizes crowdsourced sensor enriched videos and panoramic reconstructions to infer room bounds and construct floor plans. However, little research has been done for efficient data collection that ensures high QoI. Chen et al. [16] proposed a crowdsourced indoor map generation system, IndoorCrowd2D, which thrives to improve quality of the collected VCS data by warning if a user is moving too fast or facing featureless surfaces. Travi-Navi [27] utilized inertial sensor readings to predict quality of taken images. SnapTask defines QoI not only by image sharpness but also by the contribution to the creation of indoor maps. It guides participants to collect data that is needed for increasing the coverage of indoor maps and for reconstructing featureless surfaces.

VII. Conclusion

In this paper we present SnapTask, a VCS system that aims at building complete indoor maps in an efficient manner. It generates tasks that guide participants to capture photos of high QoI in unvisited areas, creates 3D models and further indoor maps from the collected photos. It also develops a novel approach for reconstructing featureless surfaces using crowdsourced annotations. Through an evaluation in a library we show that guided crowdsourcing in SnapTask generates data with higher QoI. In terms of model coverage, with a same amount of photos SnapTask outperforms unguided participatory and opportunistic crowdsourcing by 20.72% and 34.45%, respectively. In addition, with crowdsourced annotations, SnapTask manages to reconstruct featureless surfaces with 90.23% F-score. In the future, we plan to integrate incentive mechanisms and location-based participant selection into SnapTask to further improve the efficiency in data collection.

REFERENCES

[1] B. Guo, Q. Han, H. Chen, L. Shangguan, Z. Zhou, and Z. Yu, "The emergence of visual crowdsensing: Challenges and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2526–2543, 2017.

[2] S. Morishita, S. Maenaka, D. Nagata, M. Tamai, K. Yasumoto, T. Fukukura, and K. Sato, "Sakurasensor: quasi-realtime cherry-lined roads detection through participatory video sensing by cars," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2015, pp. 695–705.

[3] H. Xu, Z. Yang, Z. Zhou, L. Shangguan, K. Yi, and Y. Liu, "Indoor localization via multi-modal sensing on smartphones," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016, pp. 208–219.

[4] S. Chen, M. Li, K. Ren, and C. Qiao, "Crowd map: Accurate reconstruction of indoor floor plans from crowdsourced sensor-rich videos," in *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*. IEEE, 2015, pp. 1–10.

[5] Y. Wu, Y. Wang, W. Hu, X. Zhang, and G. Cao, "Resource-aware photo crowdsourcing through disruption tolerant networks," in *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*. IEEE, 2016, pp. 374–383.

[6] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li, "Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing," in *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 2014, pp. 249–260.

[7] J. Dong, Y. Xiao, Z. Ou, and A. Ylä-Jääski, "Utilizing internet photos for indoor mapping and localization-opportunities and challenges," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*. IEEE, 2015, pp. 636–641.

[8] H. Ma, D. Zhao, and P. Yuan, "Opportunities in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 29–35, 2014.

[9] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[10] N. Snavely, "Bundler: Structure from motion (sfm) for unordered image collections," 2006.

[11] C. Wu *et al.*, "Visualsfm: A visual structure from motion system," 2011.

[12] P. Moulon, P. Monasse, R. Perrot, and R. Marlet, "Openmvg: Open multiple view geometry," in *International Workshop on Reproducible Research in Pattern Recognition*. Springer, 2016, pp. 60–74.

[13] J. Dong, Y. Xiao, M. Noreikis, Z. Ou, and A. Ylä-Jääski, "imoon: Using smartphones for image-based indoor navigation," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 85–97.

[14] M. Noreikis, Y. Xiao, and A. Ylä-Jääski, "Seenav: Seamless and energy-efficient indoor navigation using augmented reality," in *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, ser. Thematic Workshops '17. New York, NY, USA: ACM, 2017, pp. 186–193. [Online]. Available: http://doi.acm.org/10.1145/3126686.3126733

[15] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao, "Automatically characterizing places with opportunistic crowdsensing using smartphones," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12. New York, NY, USA: ACM, 2012, pp. 481–490. [Online]. Available: http://doi.acm.org/10.1145/2370216.2370288

[16] S. Chen, M. Li, K. Ren, X. Fu, and C. Qiao, "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*. ACM, 2015, pp. 59–71.

[17] X. Zhang, Z. Yang, Y. Liu, J. Li, and Z. Ming, "Toward efficient mechanisms for mobile crowdsensing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 1760–1771, 2017.

[18] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[19] "Removing outliers using a statistical outlier removal filter," http://pointclouds.org/documentation/tutorials/statistical_outlier.php.

[20] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martinez, and J. Fernández-Valdivia, "Diatom autofocusing in brightfield microscopy: a comparative study," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 3. IEEE, 2000, pp. 314–317.

[21] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[22] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[23] Y. Zhao and Q. Han, "Spatial crowdsourcing: current state and future directions," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 102–107, July 2016.

[24] B. Zhang, Z. Song, C. H. Liu, J. Ma, and W. Wang, "An event-driven qoi-aware participatory sensing framework with energy and budget constraints," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, p. 42, 2015.

[25] Z. Song, C. H. Liu, J. Wu, J. Ma, and W. Wang, "Qoi-aware multitask-oriented dynamic participant selection with budget constraints," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4618–4632, 2014.

[26] P. Cheng, X. Lian, Z. Chen, R. Fu, L. Chen, J. Han, and J. Zhao, "Reliable diversity-based spatial crowdsourcing by moving workers," *Proceedings of the VLDB Endowment*, vol. 8, no. 10, pp. 1022–1033, 2015.

[27] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao, "Travi-navi: Self-deployable indoor navigation system," in *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 2014, pp. 471–482.