VI-SLAM2tag: Low-Effort Labeled Dataset Collection for Fingerprinting-Based Indoor Localization

Marius Laska, Till Schulz, Jan Grottke, Christoph Blut, Jörg Blankenbach
Geodetic Institute and Chair for Computing in Civil Engineering & Geo Information Systems
RWTH Aachen University
Aachen, Germany
marius.laska@gia.rwth-aachen.de

Abstract—Fingerprinting-based approaches are particularly suitable for deploying indoor positioning systems for pedestrians with minimal infrastructure costs. The accuracy of the method, however, strongly depends on the quality of collected labeled fingerprints within the calibration phase, which is a tedious process when done manually in a static fashion. We present VI-SLAM2tag, a system for auto-labeling of dynamically collected fingerprints using the visual-inertial simultaneous localization and mapping (VI-SLAM) module of ARCore. ARCore occasionally updates its internal coordinate system. Mapping the entire trajectory to a target coordinate system via a single transformation results in large drift effects. To solve this, we propose a strategy for determining locally optimal sub-trajectory transformations. Our system is evaluated with respect to the accuracy of the generated position labels using a geodetic measurement and mapping (VI-SLAM) module of ARCore. ARCore can be used to train neural network based models that achieve a positioning accuracy of roughly 2 m. VI-SLAM2tag and the dataset are made publicly available.

Index Terms—indoor localization, fingerprinting, data collection, SLAM, ARCore

I. INTRODUCTION

Obtaining the position of a person or entity is the key requirement to offer location based services (LBS) such as navigation or point-of-interest (PoI) queries. LBS have become omnipresent in the last decade and are integrated into many mobile applications. LBS also have manifold potential applications for indoor environments, such as shopping malls, airports or hospitals [1]. Presently, for retrieving the location of a person or entity indoors, no gold standard such as global navigation satellite systems for outdoors exists. The field of indoor localization has been growing for the last few years resulting in various techniques and technologies that differ mainly in their trade-off between accuracy and costs [2]. Choosing the right positioning technology should be application driven by estimating the specific requirements in terms of required accuracy of the use case, deployability and accompanied costs. Especially, for mass-market and large-scale pedestrian indoor localization, systems are required that provide medium to coarse-grained localization accuracy at minimal deployment effort and infrastructure costs. One technique that is suitable for such scenarios is fingerprinting. The idea is that a unique fingerprint can be obtained for each location within the building, where the fingerprint consists of multiple signals observed by a sensor of the device which should be localized. The mapping between fingerprints and locations can be predicted by an algorithm. Recently, machine learning (ML) has been extensively used for this purpose [3]. Training a supervised ML algorithm requires fingerprints labeled with the corresponding position. The accuracy of trained algorithms mainly depends on the quality and quantity of labeled data. When collecting these data manually at fixed reference positions, this can become a tedious task, which increases the deployment effort of such systems. Several strategies have been proposed to tackle this issue: 1) given a set of manually labeled data, new fingerprints can be constructed (e.g. via a generative model or data augmentation techniques) [4], [5]; 2) by semi-supervised learning a set of manually labeled data can be used in combination with real fingerprints that are not labeled [6]; 3) crowdsourcing allows for splitting the labeling effort across multiple participants [7] and 4) auto-labeling can decrease the amount of human intervention that is required [8]. Combinations of all strategies are possible. 1) and 2) require a certain amount of labeled data and 3) only distributes the effort by acquiring more manual workforce. Therefore, 4) has the highest potential of lowering the barrier of practical deployments.

Manual labeling can be avoided by automatically obtaining ground truth data during data collection. Ideally, the ground truth should originate solely from the device (mostly smartphone) which is used to collect the fingerprints. This requires the fusion of several sensors. For example, Zee [9] utilizes the inertial measurement unit (IMU) of the smartphone and applies pedestrian dead reckoning within a particle filter. An additional sensor, which can drastically improve the localization quality is the smartphone camera. Augmented Reality

The dataset can be found at DOI:10.5281/zenodo.6801310. The implementations are hosted at https://github.com/laskama.

978-1-7281-6218-8/22/$31.00 © 2022 IEEE
(AR) frameworks like ARCore (Google) or ARKit (Apple) provide great potential for location estimation, since they offer out-of-the-box visual inertial simultaneous localization and mapping (VI-SLAM), which are optimized for mobile hardware (smartphones). Still, ARCore cannot be directly used to obtain absolute position tags as it only tracks the position within its local coordinate system. Furthermore, the local coordinate system is occasionally updated to keep consistency, which makes estimating a global transformation for mapping the entire trajectory challenging [10].

We present VI-SLAM2tag, a system consisting of a smartphone application that leverages ARCore for relative pose tracking and a post-processing approach based on landmark mapping to obtain position tags within a defined target coordinate system. To handle coordinate system updates of ARCore, we propose a mapping strategy that determines locally optimal sub-trajectory transformations. Using our system we collect a multi-floor dataset in our university building. The dataset is suitable for training models that require continuously labeled data such as RSS fingerprinting models or learned models for relative position estimation using IMU data [11].

The rest of the paper is structured as follows. We begin by discussing related work with respect to reducing the manual labeling effort of fingerprinting-based indoor localization and SLAM-based localization in section II. Afterward, we describe the VI-SLAM2tag workflow in section III and introduce the smartphone application by covering the fundamental concepts of ARCore in section IV. In section V we introduce two post-processing strategies for obtaining position tags within the target coordinate system. Our VI-SLAM2tag system is evaluated in section VI with respect to the accuracy of the obtained position tags. Finally, in section VII we introduce the dataset that we collected and show the baseline performance of two algorithms. The results are concluded in section VIII.

II. Related Work

A. Decreasing manual labeling effort for fingerprinting

While fingerprinting-based indoor localization does not require a dedicated positioning infrastructure, it relies on a rich set of labeled fingerprints to achieve a satisfactory localization accuracy. The most basic approach for setting up such a fingerprinting database is to collect fingerprints at predetermined reference points in the calibration phase. Using a collection device (smartphone) the surveyor statically stands over the reference position and collects one or more fingerprints. This procedure is very time-consuming, which makes fingerprinting-based positioning systems tedious to deploy, especially within large-scale environments such as shopping malls. A large body of research has been focused on reducing the manual labeling effort required for data collection, where works can be mainly grouped into 4 categories:

1) Data augmentation/generative models: Given an initial set of labeled fingerprints, the objective can be to artificially enlarge it. Inspired from data augmentation in image classification, one can slightly alter existing fingerprints. Sinha et al. [12] add noise to certain AP-RSS entries while being compliant to observed statistical quantities. Anagnostopoulos and Kalousis [4] propose proximity-based augmentation. They enlarge the dataset by combining fingerprints that have been collected within close proximity. Another approach is to train generative models that predict fingerprints even for locations that have not yet been visited before. Njima et al. [5] train a generative adversarial network which generates fake fingerprints (RSS only) data based on a small set of real collected labeled data. Subsequently, pseudo-labels of the generated RSS fingerprints are predicted.

2) Semi-supervised learning: Unlabeled data is much easier to collect, since it does not require human intervention and can be collected while freely walking around the site. Together with a core-set of labeled data it can be used to improve the learning process of neural networks. Chidlovskii and Antsfeld [6] train a deep variational autoencoder to utilize a large set of unlabeled data to enhance the performance of various localization models.

3) Crowdsourcing: Instead of relying on a single person that collects the labeled data, the surveying process can be split among several participants, possibly non-trained experts. While this drastically reduces the survey time, it introduces further risks of faulty data and requires incentives for user participation [7], [13], [14].

4) Semi-automatic labeling: The labeling effort can be drastically decreased by semi-automatically inferring position tags during site survey. By utilizing additional sensor sources, the location of the sensing device can be determined during data collection. Rai et al. [9] use a particle filter which utilizes the IMU and achieves further convergence by the already gained knowledge about annotated WLAN fingerprints of already visited locations. As an alternative sensor besides IMU, the smartphone camera offers great potential for relative position estimation via visual odometry. By detecting feature points and tracking them across successive camera frames, the change in position can be inferred. Tomazic and Skrjanc [8] perform online calibration of BLE signal strength mapping via Android ARCore.

The first two approaches still require a certain amount of labeled data and crowdsourcing only lowers the effort by acquiring more manual workforce. Therefore, the last category (semi-automatic labeling) has the highest potential of lowering the barrier of practical deployments and can be combined with the other three strategies.

B. VI-SLAM

Estimating and tracking the pose of a moving object from camera images are widely studied research subjects, also in the field of indoor localization. Prominent application scenarios are autonomous driving systems and robot navigation [15]. The applied methods can be differentiated between visual odometry (VO) and visual simultaneous localization and mapping (V-SLAM). VO derives poses relative to an arbitrary starting pose by comparing the camera movement between successive camera images. V-SLAM is based on VO and extends it by incorporating a 3D map into the process,
which is constructed during localization. The map enables optimizing the total trajectory and current pose via loop closure, when revisiting a previous location. For V-SLAM, typically a monocular or stereo camera setup is used. While depth (scale) can be obtained directly using a stereo setup, a monocular setup does not provide this information, so that additional sensors, such as an IMU, must be integrated via sensor fusion. The process is referred to as visual inertial SLAM then (VI-SLAM). While there exist multiple V-SLAM solutions, there are only a few that take inertial data into account. ORB-SLAM [15], [16] is a well-known framework for V-SLAM and recently was extended to VI-SLAM, but is not optimized for mobile applications. On smartphones, Augmented Reality (AR) has gained popularity in the past years, so that Google and Apple provide the frameworks ARCore for Android and ARKit for iOS to create AR experiences. Both frameworks contain highly optimized VI-SLAM-based pose tracking solutions for mobile devices. Given that fingerprints are mostly collected with smartphones, those frameworks offer great potential for automatically obtaining position tags. However, since they are mostly closed source and not originally intended for localization, it is challenging to design such a system. It has been shown that computing a mapping by simply aligning the origin and axes of ARCore’s local coordinate system and those of the real-world coordinate system does not suffice, since drift of up to 17m per 120m trajectory occurs [18].

III. VI-SLAM2TAG WORKFLOW

The goal of VI-SLAM2tag is to obtain sensor data (WiFi fingerprints and IMU data) that are automatically labeled with the position of collection within a specified building coordinate system (BCS). ARCore is utilized for providing the basis for annotating the sensor data, however, it tracks the smartphone position within its own local coordinate system, which we refer to as ARCore coordinate system (ACS) from now on. Simple transformations from ACS to BCS as described in [18] do cause severe drift effects. Instead, we propose to estimate a transformation by using corresponding point pairs of both coordinate systems. This is enabled by detecting manually placed landmarks within the environment of which we exactly know the position within the BCS. The workflow of VI-SLAM2tag is presented in Figure 1 and is described in the following.

Initially, a landmark configuration has to be chosen for a given floor of the building. Landmarks should be placed after each critical turn a data collector might take when walking around the building. Next, the landmarks need to be installed at the physical locations within the building and their positions within the BCS have to be exactly determined by geodetic surveying (e.g. using a total station). Now, an arbitrary amount of data can be collected while walking around with a smartphone that logs its position in the ACS along with sensor data (e.g. WLAN fingerprints and IMU data) for subsequent annotation. Once the collector passes a landmark, it is scanned via the smartphone to obtain its coordinates in the ACS. Afterwards, the estimated transformation of the coordinate pairs is used to transform the entire trajectory from ACS to BCS. Finally, the collected sensor data are annotated with the transformed coordinates to obtain a labeled fingerprinting dataset. In the following we will introduce the design of the smartphone application for data collection in section IV and subsequently explain how we obtain a mapping to the BCS, which is robust against failure of the SLAM-based positioning in section V.

IV. VI-SLAM BASED DATA COLLECTION (VIA ARCore)

A. Fundamental concepts of ARCore

In the previous section we identified that ARCore is a suitable solution for designing an easy to use labeled data collection system. In this section we will not cover the exact VI-SLAM algorithm behind ARCore, since its mostly closed source. For a more thorough explanation the reader is referred to the corresponding patent [19]. Instead, we want to point out the main principles and assumptions of ARCore which make it suitable for the task at hand. Those can be found in the official API documentation [10].

ARCore continuously tracks the smartphone’s pose within the virtual world and updates its understanding of the virtual world based on the collected data. It uses visual odometry for
computing relative pose changes by identifying matching feature points in consecutive image frames and tracking them over time. Since the smartphone camera is monocular, the scale cannot be inferred directly, but is obtained by fusing data of the smartphone’s IMU. In the following, italic words represent concrete objects/concepts within the ARCore framework. A location within the ACS is represented by a Pose. The current position can therefore be obtained by accessing the Pose of the Camera object. However, as ARCore’s understanding of the environment changes over time, it adjusts its model of the world to keep things consistent. This means that the ACS is updated, such that the camera pose might appear at notably different coordinates after the update. Therefore, it is problematic to use the Pose object of the Camera to obtain the user’s relative position across different camera frames, which is the motivation for our proposed mapping strategy introduced in section V-B.

ARCore introduces a concept called Anchors to attach objects to fixed locations within the virtual world. This is mainly done for supporting AR applications. One can place an object on a table and it will appear at the same exact position, even if ARCore updates its ACS. In our approach anchors will be used to represent the fixed locations of the landmarks within the virtual world.

ARCore offers a concept called Augmented Images for detecting previously defined 2D images within the virtual world. If such an image is detected it is automatically tracked from now on via placing an anchor at its estimated position. When developing our application, we utilize this concept for detecting our landmarks. Once a landmark appears close within the current camera frame, ARCore will identify it and automatically track it even if it leaves the camera frame subsequently. This allows us to log the position of each detected landmark (after it has been recognized the first time) for each camera pose obtained by the system.

B. VI-SLAM2tag smartphone application

To implement our data collection application, we extend the sample project computer_vision of the official sample application provided by the arcore-android-sdk [20]. The main activity implements the GLSurfaceView.Renderer interface, which consists of the onDrawFrame function that is called once a new camera frame is available. Via the Session object of ARCore we acquire the camera object of the current frame, which allows for accessing its current pose. For each new frame, we check whether an augmented image was detected. If a new image was detected we add it to a list of currently tracked images. In the onDrawFrame method we log the pose of the camera (of the current frame) and the poses of each detected augmented image together with a current timestamp of the smartphone. Simultaneously, we log all the data from the smartphone’s IMU including TYPE ACCELEROMETER, TYPE GYROSCOPE, TYPE MAGNETIC FIELD and TYPE ROTATION VECTOR. Furthermore, we continuously request WLAN network scans via the WifiManager. For each session that we run, we create four separate logging files:

- CameraPose.csv: holds the camera poses of each recorded frame
- LandmarkPose.csv: contains the current landmark poses (detected via augmented images) of each recorded frame
- Sensors.csv: contains all recorded sensors
- WiFi.csv: contains the results of the WLAN scans

Each entry in the files has an attached timestamp (in the same reference time) such that the entries can be matched. Note that the timestamps in CameraPose.csv & LandmarkPose.csv are identical, while the other two files have to be aligned to the recorded frame time stamps.

During post-processing we compute a mapping to transform the camera poses from ACS to BCS by utilizing the continuously logged landmark positions. Subsequently, we assign a position to each entry of the Sensors.csv and WiFi.csv files by applying a time-based matching strategy.

V. MAPPING TO BUILDING COORDINATE SYSTEM (BCS)

A. Global mapping via least squares

We identify a mapping from the positions of the landmarks in ACS to their corresponding positions in the BCS. This mapping can subsequently be used to transform all points of the trajectory. The ACS has 3 dimensions, however, we discard the z-axis and assume that a single trajectory does not cover multiple floors. Further, the landmark height and the smartphone height are kept mostly constant. The positions of our landmarks are continuously logged via the smartphone application (once they have been detected). We compute the median position among all logged values for each landmark. Finally, we estimate the parameters of a similarity transformation via least squares minimization given the two point patterns as introduced in [21]. In case the ACS stays mostly constant over the entire trajectory, this approach is suitable to achieve a highly accurate mapping.

B. Locally optimal mapping of sub-trajectories

ARCore continuously updates its internal world view, which results in an updated ACS. If these updates are significant, the logged positions of the static landmarks will deviate largely causing the global mapping strategy to fail. In order to account for these updates, we compute locally optimal mappings, which are based on the most recent understanding (ACS) of ARCore. We propose to partition the trajectory into sub-trajectories, which are defined as the paths between each distinct landmark pair and compute individual transformations for each sub-trajectory based on the landmark positions in the most recent ACS.

To identify the sub-trajectories, we determine the poses where to split the trajectory. We detect possible candidates whose distance to a landmark lies below a predefined threshold (e.g. 1m). For each of these candidate intervals we compute the local minimum. Those minima identify the poses at which we will divide the trajectory. The process is illustrated in Figure 2, the dotted lines represent the minima. Figure 3 shows how each sub-trajectory is locally mapped to the BCS. The individual similarity transformations of the sub-trajectories are
obtained by mapping the line between the landmark pair of the sub-trajectory from ACS to BCS by translation, uniform scaling and rotation. The resulting transformation is then applied to all poses of the given sub-trajectory. Since this approach uses the most recent understanding of the world by utilizing the current positions of the landmarks instead of their global averages, this approach is much more robust against non-static ACS, as we will demonstrate in the evaluation.

Since each sub-trajectory can be mapped individually, we can additionally discard certain segments that have a high probability of failure. We identify two main causes for failures: 1) ARCore loses track of previously visited landmarks. In particular, the camera pose and the landmark will be identical for a certain segment of frames. If we detect such cases, we set the recorded landmark position to “unknown”. Given a sub-trajectory where ARCore loses track of all landmarks, we will discard the whole sub-trajectory; 2) It might happen that the camera pose jumps between two consecutive poses. These jumps are of a magnitude which is impossible to cover during a single frame. If jumps occur during a sub-trajectory, we will discard the whole sub-trajectory. These two strategies allow us to identify parts of our walk where the VI-SLAM algorithm fails, while still recovering those parts where the algorithm was functioning as expected. In contrast, when applying the global mapping, we would have to discard the entire trajectory. Small flawed parts of the trajectory distort the global mapping excessively, rendering the whole trajectory unusable. The qualitative mapping results of both strategies on a trajectory that contains severe ACS updates are exemplified in Figure 4.

We place fixed control points on the floor and accurately register their positions in the BCS via geodetic surveying. Every time the user walks over a control point, he or she presses a button within the smartphone application such that the current timestamp is logged. Afterwards, we compare the post-processed position at the logged timestamp with the closest known control point. This evaluation strategy is meant for assessing the accuracy of the entire approach (including the post-processing) and roughly provides the final accuracy that is to be expected by the system.

| Trajectory selection | Metric | Error of Local | Error of Local (corrected) | Error of Global (LS) |
|----------------------|--------|----------------|--------------------------|---------------------|
| All trajectories     | Mean   | 0.91           | 0.53                     | 2.80                |
|                      | Median | 0.63           | 0.49                     | 0.54                |
|                      | Maximum| 4.51           | 0.87                     | 20.78               |
| Non-faulty trajectories | Mean   | 0.51           | -                        | 0.43                |
|                      | Median | 0.49           | -                        | 0.40                |
|                      | Maximum| 0.72           | -                        | 0.72                |

TABLE I: Comparison of location error [m] of all mapping algorithms.
In total we collected 24 reference trajectories on the 1st and 4th floor of our test building. We used two smartphones (LG V30 and OnePlus 6) and walked for a median duration of roughly 3 minutes per trajectory. Table I shows the obtained labeling accuracy, grouped into all trajectories and those where no localization error was detected by our mapping algorithm. The global mapping strategy has the highest accuracy on the non-faulty trajectories, since its least-squares mapping is more precise than locally mapping the sub-trajectories with only two corresponding points. Both mapping strategies achieve a median error of below 50 cm. However, on the flawed trajectories (e.g. updated ACS), the mean error of the global mapping rises to above 2m. In contrast, with the local mapping strategy, we can maintain a mean labeling error of just above 50 cm. We are interested in when the global mapping strategy fails, which is analyzed in Figure 6. It can be seen that the global mapping fails, if significant position updates of the logged landmark positions (fixed in real-world) happen due to ARCore updating its ACS. Since we utilize the median location of the landmarks for computing the global mapping, the high deviation in the logged landmark positions causes severe mapping inaccuracies and might even result in total failure as already shown in Figure 4. In contrast, the local strategy utilizes the most recent location of the landmarks in the current ACS to obtain a mapping for each sub-trajectory, such that even for critical trajectories a high accuracy can be maintained.

To check whether the system is robust against drift effects, we walked a long trajectory of 15 minutes. The smallest error of 57 cm was achieved by the global mapping, which is comparable to those of the significantly shorter trajectories. The obtained mapped trajectories are visualized in Figure 5 where the blue stars represent the positions of the control points and the black points represent the positions when the collector pressed the button to indicate that he or she just walked over a control point.

B. Accuracy estimation via total station (TS) tracking

The evaluation via comparison at ground truth control points can be used for roughly estimating the real-error among the whole trajectory. However, the measuring system itself,
Metric | Error of Local | Error of Local (corrected) | Error of Global (LS)
---|---|---|---
Mean | 0.73 | 0.18 | -
Median | 0.23 | 0.18 | -
Maximum | 2.68 | 0.31 | -

TABLE II: Mapping error [m] for total station ground truth evaluation.

introduces a certain error, because manually clicking the button while walking over a control point has an inherent uncertainty. Furthermore, we only know the ground truth position when walking over the control point, which is only a snapshot given the whole trajectory. The system is mostly meant for WLAN fingerprint collection, therefore we assume that an accuracy below a meter is sufficient for automatic data labeling. However, we are also interested in assessing the accuracy of shorter trajectories with a more accurate ground truth measuring system that also allows for more frequent ground truth values. Therefore, we utilize a total station (TS) (Leica Nova MS50) as an additional tracking system that provides accurate position information at a high sampling rate. The TS requires direct line-of-sight to the prisma installed at the monitored device (smartphone), which is why we will only apply it to trajectories in a corridor as depicted in Figure 7. The labeling accuracy of all collected trajectories is listed in Table II. We collected several trajectories with changing velocities, which did not cause notable positioning errors, however, when instantly running, the algorithm fails, which results in sudden position jumps in the recorded trajectories. Those are detected and discarded by the local (corrected) mapping strategy, which is why its mean and max errors are significantly lower. The high labeling accuracy is qualitatively shown in Figure 7, where the ground truth position (black) can hardly be distinguished from the post-processed trajectory (orange). During data collection, few people were present in the building, however, occasional acquaintances did not cause degrading performance. Out of curiosity, we simulated noise by a second persons that was heavily moving in the camera’s view, which resulted in failure of the VI-SLAM algorithm. Therefore, it is advisable to not perform data collection during peak times of building occupancy.

VII. THE GIAINDOORLOC DATASET

With the help of the introduced VI-SLAM2tag system we collect a dataset at our university building that spans 5 floors. The entire data collection took less than 2 hours for 2 persons, which includes the setup time at each floor (placing landmark images). The site survey resulted in a total of 2049 auto-labeled fingerprints (after automatically discarding possibly wrong labels), which stemmed from 4 different smartphones (OnePlus 6, LG V30, Samsung S20 Ultra, Samsung Galaxy A52S). Statistics on the dataset are depicted in Table III and Figure 8. The dataset allows for training supervised models at several tasks including WLAN- or magnetic-based fingerprinting for absolute position estimation but also relative position estimation via end-to-end deep learning on IMU data as proposed in [11]. In the following we will demonstrate how to the dataset can be used for training WLAN fingerprinting-based models. When requesting WLAN scans via the Android operating system, each recorded entry is tagged with the timestamp of collection. Those can slightly differ for the seen APs in a single scan as the scan might require several seconds depending on the hardware of the device. Since we obtain high frequent pseudo-ground truth labels via VI-SLAM2tag, we are technically able to assign accurate positions for each AP entry within a single network scan as depicted in Figure 9. In order to train our model in the standard way of having a single position label for each network scan (fingerprint), we average the recorded positions per scan to compute a global label for each scan (black stars). However, we want to emphasize that the dynamic collection in combination with the high frequent position annotations enable further advanced fingerprint generation following spatial data augmentation schemes.

Fig. 8: Distribution of fingerprints across devices and floors of the building.

Fig. 9: Visualization of obtained WLAN fingerprints of example trajectory. Each dot represents the smartphone position at the time of an obtained AP RSS reading. A single color is chosen per network scan. The black stars represent the average position of each scan, which is used for annotation.
A. Baseline fingerprinting performance

For assessing a reference performance on the dataset we choose two static models based on neural networks. A standard multi-layer perceptron with a 3D-regression output and the recently proposed multi-CEL model [22], which was explicitly designed for multi-floor localization using a single neural network. We utilize multiple train/test splits to answer two questions: i) what is the general localization accuracy across all devices by a random split into 30% test data and ii) how well does the algorithm generalize across devices by using the data of a single phone for testing only.

\[
\begin{array}{|c|c|c|c|}
\hline
\text{Test device} & \text{Model} & \text{Mean error [m]} & \text{Median error [m]} \\
\hline
\text{Random} & mCEL & 0.992 & 2.08 \quad 1.52 \\
\text{split (30\%)} & 3Dreg & 0.988 & 2.59 \quad 2.12 \\
\hline
\text{Galaxy} & mCEL & 0.987 & 2.68 \quad 2.02 \\
& 3Dreg & 0.979 & 3.36 \quad 2.53 \\
\hline
\text{LG} & mCEL & 0.992 & 2.59 \quad 1.71 \\
& 3Dreg & 0.967 & 2.93 \quad 1.94 \\
\hline
\text{OnePlus} & mCEL & 0.968 & 3.06 \quad 1.76 \\
& 3Dreg & 0.981 & 3.80 \quad 2.82 \\
\hline
\text{S20} & mCEL & 0.999 & 2.05 \quad 1.74 \\
& 3Dreg & 0.993 & 2.84 \quad 2.37 \\
\hline
\end{array}
\]

TABLE IV: Performance on several test data splits.

The results are shown in Table IV. Irrespectively of the train/test split, m-CEL outperformed the 3D-regression model with respect to floor detection accuracy and mean/median positioning error. It achieves a mean positioning error of roughly 2m. Furthermore, the model is able to generalize well across different devices, reaching a similar mean positioning accuracy as compared to the random split.

VIII. CONCLUSION

We presented VI-SLAM2tag for dynamically collecting auto-labeled fingerprints. It consists of i) a smartphone application that uses ARCore for tracking the position during fingerprint collection and ii) a post-processing module that transforms the position tags to the target coordinate system while being robust against coordinate system updates of ARCore. The obtained labeling accuracy was evaluated using two ground truth (GT) measuring systems. The control point-based GT system showed a labeling accuracy of roughly 50 cm even for trajectories of up to 15 minutes. To exclude measurement uncertainties, we additionally utilized a total station which indicates that for short trajectories a labeling accuracy of below 20 cm can be expected. This is sufficient for WLAN fingerprinting, since the prediction accuracy lies well above the labeling accuracy. Within 2 hours we collected a labeled dataset within a 5-floor building of our university and demonstrated that trained models achieved a positioning accuracy of roughly 2m. In future work we want to leverage the high frequent labels of the dynamically collected fingerprints via spatial data augmentation techniques. VI-SLAM2tag is made openly accessible to simplify future deployments of fingerprinting-based indoor localization systems.

REFERENCES

[1] A. Basiri, E. S. Lohan, T. Moore, A. Winstanley, P. Peltola, C. Hill, P. Amiran, and P. Figueiredo e Silva, “Indoor location based services challenges, requirements and usability of current solutions,” Computer Science Review, vol. 24, no. April, pp. 1–12, 2017.
[2] F. Zafari, A. Gkelias, and K. K. Leung, “A Survey of Indoor Localization Systems and Technologies,” IEEE Communications Surveys and Tutorials, vol. 21, no. 3, pp. 2568–2599, 2019.
[3] V. Bellavista-Parent, J. Torres-Sospedra, and A. Perez-Navarro, “New trends in indoor positioning based on WiFi and machine learning: A systematic review,” in 2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN). Lloret de Mar, Spain: IEEE, 2021, pp. 1–8.
[4] G. G. Anagnostopoulos and A. Kalousis, “ProxyFAUG: Proximity-based Fingerprint Augmentation,” in 2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN). Lloret de Mar, Spain: IEEE, Nov. 2021, pp. 1–7.
[5] W. Njima, M. Chafì, A. Chorti, R. M. Shabair, and H. V. Poor, “Indoor Localization Using Data Augmentation via Selective Generative Adversarial Networks,” IEEE Access, vol. 9, pp. 98 337–98 347, 2021.
[6] B. Chidlovskii and L. Antsfeld, “Semi-supervised Variational Autocoder for WiFi Indoor Localization,” in 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN). Pisa, Italy: IEEE, Sep. 2019, pp. 1–8.
[7] Q. Jiang, Y. Ma, K. Liu, and Z. Dou, “A Probabilistic Radio Map Construction Scheme for Crowdsourcing-Based Fingerprinting Localization,” IEEE Sensors Journal, vol. 16, no. 10, pp. 3764–3774, 2016.
[8] S. Tomažič and I. Škrjanc, “An Automated Indoor Localization System for Online Bluetooth Signal Strength Modeling Using Visual-Inertial SLAM,” Sensors, vol. 21, no. 8, pp. 2857, Apr. 2021.
[9] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, “Zec: Zero-effort crowdsourcing for indoor localization,” in Proceedings of the 18th Annual International Conference on Mobile Computing and Networking - Mobicom ’12. Istanbul, Turkey: ACM Press, 2012, p. 293.
[10] Google, “ARCore,” https://developers.google.com/ar/develop?hl=de.
[11] E. Nerurkar, S. Lynen, and S. Zhao, “System and method for concurrent odometry and mapping,” 2017.
[12] R. S. Sinha, S. M. Lee, M. Rim, and S. H. Hwang, “Data augmentation schemes for deep learning in an indoor positioning application,” Electronics (Switzerland), vol. 8, no. 5, May 2019.
[13] S. H. Jung and D. Han, “Automated Construction and Maintenance of Wi-Fi Radio Maps for Crowdsourcing-Based Indoor Positioning Systems,” IEEE Access, vol. 6, pp. 1764–1777, 2017.
[14] G. Pipelidis, N. Tsiamitros, E. Ustaoglu, R. Kienzler, and P. Nurmi, “Cross-Device Radio Map Generation via Crowdsourcing,” 2019.
[15] C. Campos, R. Elivira, J. G. Rodriguez, J. M. Montiel, and J. D. Tardos, “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM,” IEEE Transactions on Robotics, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.
[16] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System,” IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[17] Apple, “ARKit: Augmented Reality.” [Online]. Available: https://developer.apple.com/documentation/arkit.
[18] T. Feigl, A. Porada, S. Steiner, C. Löffler, C. Mutschler, and M. Philippens, “Localization Limitations of ARCore, ARKit, and Hololens in Dynamic Large-scale Industry Environments,” in Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. Valletta, Malta: SCITEPRESS - Science and Technology Publications, 2020, pp. 307–318.
[19] E. Nerarkar, S. Lynen, and S. Zhao, “System and method for concurrent odometry and mapping,” 2017.
[20] “ARCore SDK for Android,” Google AR, Mar. 2022. [Online]. Available: https://github.com/google-ar/arcore-android-sdk.
[21] S. Uneyama, “Least-squares estimation of transformation parameters between two point patterns,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, no. 4, pp. 376–380, Apr. 1991.
[22] M. Laska and J. Blankenbach, “Multi-Task Neural Network for Position Estimation in Large-Scale Indoor Environments,” IEEE Access, vol. 10, pp. 26 024–26 032, 2022.