LocIn: Inferring Semantic Location from Spatial Maps in Mixed Reality

Habiba Farrukh, Reham Mohamed, Aniket Nare, Antonio Bianchi, and Z. Berkay Celik

Purdue University
{hfarrukh, raburas, anare, antoniob, zcelik}@purdue.edu

Abstract

Mixed reality (MR) devices capture 3D spatial maps of users' surroundings to integrate virtual content into their physical environment. Existing permission models implemented in popular MR platforms allow all MR apps to access these 3D spatial maps without explicit permission. Unmonitored access of MR apps to these 3D spatial maps poses serious privacy threats to users as these maps capture detailed geometric and semantic characteristics of users' environments. In this paper, we present LocIn, a new location inference attack that exploits these detailed characteristics embedded in 3D spatial maps to infer a user's indoor location type. LocIn develops a multi-task approach to train an end-to-end encoder-decoder network that extracts a spatial feature representation for capturing contextual patterns of the user's environment. LocIn leverages this representation to detect 3D objects and surfaces and integrates them into a classification network with a novel unified optimization function to predict the user's indoor location. We demonstrate LocIn attack on spatial maps collected from three popular MR devices. We show that LocIn infers a user's location type with an average 84.1% accuracy.

1 Introduction

Mobile mixed reality (MR)[1] has become increasingly popular over the last decade with the release of dedicated headsets and apps that blend virtual content into users' real-world environments. Apart from gaming and entertainment, mobile MR applications have recently found utility in enabling interactive healthcare, education, and e-commerce experiences [20, 30, 34]. For instance, e-commerce apps like IKEA place [31] allow customers to experience how a product fits in their environment before purchasing.

MR apps require an elaborate description of the user's surroundings in three dimensions (3D) to localize the MR device and enable realistic assimilation of virtual content in the user's environment. To capture the user's 3D environment, common MR devices, including dedicated headsets and even off-the-shelf smartphones, are equipped with specialized sensors such as depth cameras and LiDAR sensors. For instance, HoloLens 2 equips a depth camera, and Apple's latest iPhone and iPad Pro employ a LiDAR sensor to capture the real-time depth of the surrounding space and objects [3]. These sensors capture the distance between the device and physical points in the environment to generate a *3D spatial map* of the environment.

As mobile MR adoption grows [49], there is increasing concern about the security implications of 3D spatial maps accessed by mobile apps [21, 73]. All MR apps require explicit user permission for camera access to deliver their functionality, i.e., integrate virtual content in the user's environment. Once camera permission is granted, MR apps on popular MR platforms [5, 7, 50] have access to the 3D spatial maps. MR apps' access to 3D spatial maps opens doors to a new type of reconnaissance attack where an adversary-controlled malicious app exploits the 3D spatial map of the user's environment to infer user's indoor locations (i.e., *semantic location*).

An adversary's ability to locate users enables them to launch physical attacks and case a target's environment for burglary and assault. With mobile MR use cases in entertainment, education, and retail, such threat broadly applies to our homes, businesses, educational facilities, and many others. Moreover, an adversary can combine the user's indoor location information along with the object and semantic properties of the user's environment to build a profile for delivering personalized ads and recommendations. An adversary can also exploit this to understand users' socio-economic status, accessibility requirements, and product preferences, as well as their identity, routines, and activities.

In this paper, we study how an adversary can exploit 3D spatial maps from MR devices to infer a user's indoor location. Prior work has explored indoor location inference from 2D and RGB-D images [16, 24, 42, 70, 76, 77]. However, the direct application of existing approaches to spatial maps is impractical because location inference from spatial maps requires a different type of feature extraction and optimization process due to its sparse, non-uniform, and dynamic nature.

[1] We use MR as an umbrella term for augmented and virtual reality.

To achieve this, a recent work [22] built a location classifier from spatial maps. This approach, however, suffers from two main limitations. First, it only uses the high-level geometric features of spatial maps without leveraging their semantic context. The lack of such semantics leads to poor accuracy in inferring different indoor environments. Second, it compares a given location of a user with a labeled database of that user's previously visited locations with the goal of finding whether the user has been in that location before. From an attack perspective, it fails to infer the location of users without knowing a priori the spatial maps of their indoor environments. These limit its attack practicability, ultimately making it infeasible to conduct a location inference attack in practice.

To this end, we present LocIn, a location inference attack on mobile MR devices which exploits 3D spatial maps to infer a user's location. We observe that indoor environments are uniquely characterized by their semantic context (e.g., objects and surfaces in the environment). Yet, unlike pixel arrays in images, a 3D spatial map is a set of unordered points with non-uniform density, making detecting objects and surfaces in the environment challenging. Therefore, we introduce a new location inference learning representation that composites the geometric and contextual patterns embedded in the spatial map to infer a user's location. We design a multi-task learning approach and build an end-to-end encoder-decoder network that can successfully infer the user's location from spatial maps captured by various MR devices.

LocIn first preprocesses the 3D points in a 3D spatial map through farthest point sampling [47] to remove outlier points and subsample the map to a fixed number of points. The preprocessed map is then fed as input to LocIn's spatial encoder, which extends a hierarchical neural network [59] to extract a spatial feature representation of the map. This feature representation embeds geometric and contextual patterns of the user's environment. This representation is invariant to dynamic changes (e.g., changes in viewing angle or size of the map) in spatial maps as users interact with the MR apps. Lastly, LocIn's encoder output is fed to LocIn's multi-task location decoder. The multi-task decoder performs 3D object detection and semantic segmentation to generate 3D bounding boxes of objects and point-wise labels for objects and surfaces in the environment. It then integrates the object and semantic context into a classification network to predict the location type (e.g., bedroom, office) of the input spatial map.

We present three studies to evaluate LocIn's effectiveness. In a first study, we evaluated LocIn on a dataset [13] consisting of ~1,500 spatial maps collected via an iPad Air 2 equipped with a depth sensor belonging to 13 unique indoor location types where MR devices are typically used. To demonstrate LocIn's effectiveness on MR devices with different depth sensing techniques, in a second study, we evaluated LocIn on a dataset [8] collected through an iPad Pro equipped with a LiDAR scanner consisting of ~5,000 spatial maps. The spatial maps belonged to 9 unique indoor location types in real-world homes. Finally, to show LocIn attack's practicality on dedicated headsets, we evaluated LocIn on our dataset of spatial maps collected via HoloLens 2. LocIn correctly predicted the location types from the spatial maps with an average accuracy of 84.1%. We also show the LocIn attack is robust against varying sparsity (number of points) and size of the 3D spatial maps.

In summary, we make the following contributions:

- We present a location inference attack on mobile mixed reality devices that exploits the 3D spatial maps captured from users' environments to predict their location type.
- We design LocIn, a multi-task learning framework that leverages an encoder-decoder architecture to infer the user's location by integrating the geometric and contextual patterns embedded in the spatial maps.
- We evaluate LocIn on 3D spatial maps captured using three MR devices from 13 unique location types. We demonstrate that LocIn can infer a user's location with an average accuracy of 84.1%, and it is robust against varying sizes and sparsity of the spatial maps.

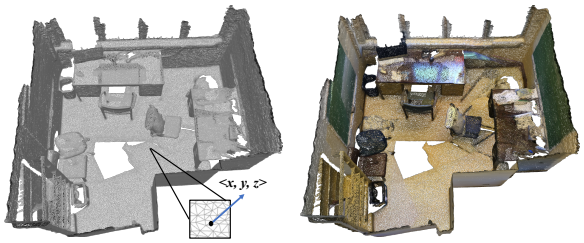
2 Background

Mixed Reality. The influx of reality-altering headsets and applications has brought mixed reality (MR) to the spotlight in recent years. Consequently, mobile industry has been striving to enable comfortable and realistic MR experiences for users. For instance, Google and Apple released ARCore [5] and ARKit [7] to enable MR app development for smartphones and tablets. Moreover, dedicated standalone headsets, such as HoloLens, have recently emerged with their own MR development platforms, e.g., Windows Mixed Reality Toolkit [50].

MR devices integrate virtual reality (VR) and augmented reality (AR) to allow users to visualize and interact with both real and virtual content in their own physical environment. MR use-cases range from social media apps (e.g., Snapchat filters [66]) and games (e.g., Pokemon Go [56]) to e-commerce (e.g., IKEAPlace [31]) and educational apps (e.g., chemical molecular structures [10], human anatomy [26]).

To enable mixed reality experiences, MR devices embed multiple sensors, such as RGB cameras and microphones that capture input from the user's surroundings. These devices are often equipped with an inertial measurement unit (IMU) to enable tracking of users' head and body movements. Some recent MR devices are also equipped with specialized depth sensors. For instance, HoloLens 2 uses a depth camera while the latest iPhones and iPads employ a LiDAR sensor to build an understanding of their surroundings [3, 27].

3D Spatial Maps. MR devices need to accurately locate themselves in relation to the physical world and understand the objects and surfaces in the environment to seamlessly superimpose digital content in the user's surroundings. Most MR



(a) (b)

Figure 1: An example of a spatial map captured with an MR device in an office (a) without and (b) with color.

devices rely on camera and sensing-based localization techniques to locate themselves in the user's environment [64]. These techniques require access to a detailed 3D digital representation of the user's environment, which is used by MR apps to overlay virtual content in the user's surroundings. To this end, commonly available MR devices - dedicated headsets and even off-the-shelf smartphones - are equipped with depth sensors that measure the distance between the device and points in the real environment. These depth sensors include Time-of-Flight (ToF) cameras (e.g., HoloLens), and LiDAR scanners (e.g., on iPhone 13 and iPad Pro).

The 3D mapping of the user's environment is shared with MR apps to allow virtual or augmented content to interact with the physical world, e.g., anchoring a virtual object on user's desk. MR devices provide the 3D representation of the users' environment to MR apps as a *3D spatial map*. The 3D spatial map is represented by a set of 3D points $\{P = p_1, \ldots, p_n\}$, where each point p_i is a vector of its (x, y, z) coordinate in space. Figure 1 shows an example of a 3D spatial map captured by iPad Pro, with and without the color information. Some MR devices also include color and normal vectors (e.g., HoloLens [67]) representing orientation for each point in the 3D spatial map.

3 Problem Statement and Threat Model

3.1 Motivation

MR Permission Models. The MR devices leverage the same permission models as traditional mobile operating systems e.g., iOS and Android, to control apps' access to sensitive data [4, 6, 28]. Since access to the camera is essential for MR apps to visually integrate virtual content in the user's environment, users must grant camera permission to allow MR apps to function as intended. Prior works have highlighted that sensitive data such as credit cards details, sensitive documents, or bystanders' facial identities could be revealed from images and videos, captured by MR devices [14, 33, 60, 68]. Consequently, several works have attempted to protect visual privacy by only sharing user-defined privacy-preserving visual features with apps [33, 37, 68], augmenting privacy markers (e.g., QR codes, RFID tags) into the real world to avoid sensitive content [60, 61], and defining fine-grained permissions for app's access to visual data [32, 63].

Significant efforts have been made to restrict MR apps from accessing users' private data through images and videos [14]. Yet, these apps must access the 3D spatial maps of the user's environment to deliver MR content. Hence, once camera permission is granted to an app, it can access the spatial map.

Security and Privacy Implications of Spatial Data. Access to 3D spatial maps of the user's environment poses serious privacy threats as these maps capture privacy-sensitive cues about the user's surroundings. For instance, a spatial map captures detailed characteristics of the environment, such as its geometric properties (e.g., length and width of the room), and embeds semantic information about the types of objects or surfaces present in it. Similar to how humans perceive an indoor environment based on its layout, objects, and surface properties [17], an adversary can extract these characteristics from the 3D spatial map to infer user's location, i.e., the type of indoor environment.

Unlike images and videos, 3D spatial maps are not easily interpretable by average MR users and, therefore, are not perceived as sensitive data yet, exacerbating their privacy threat. Moreover, spatial maps, unlike images, are not sensitive to lighting conditions, occlusions, or camera orientations/viewpoints which allows an adversary to infer private information about the user in a variety of scenarios.

An adversary can exploit spatial maps to infer a user's location (e.g., user is at their office or home), without explicitly requesting location permission from the user. The adversary could exploit this information to launch a physical attack (e.g., robbery or assault). The adversary could also gain a fine-grained understanding of the user's routine (e.g., when the user wakes up, goes to work) based on the inferred locations across time. This information can be leveraged by data brokers aiming at selling detailed user profiles to third parties.

An adversary can also combine the object and semantic segmentation information embedded in a spatial map with the inferred location to reveal additional private information about the user. First, an adversary could leverage the detected objects and the inferred location to understand users' socioeconomic status, accessibility requirements, and product preferences. For instance, a wheelchair in a bedroom or handles near a toilet might indicate a user's accessibility needs. Similarly, the type of appliances found in a user's environment could reveal their income level. Second, an adversary could use information about rooms' and objects' sizes and timing of a collected map to distinguish locations visited by a user (e.g., bed on day-1 vs. day-2) and infer users' identity, behavior, and activities. For instance, the number of beds in a room could reveal the familial structure of a user, and the presence of athletic equipment could indicate a user's hobbies. Lastly, with recent permissions for tracking user activities on mobile OS (e.g., Apple's app-tracking-permission), access to spatial

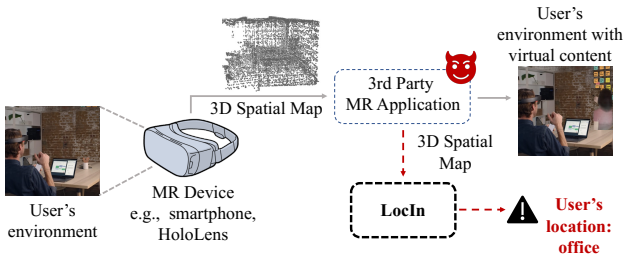


Figure 2: A malicious MR application accesses the 3D spatial map of a user's environment to integrate virtual content. The app can exploit this map to infer the user's location.

maps aid adversaries in building user profiles for delivering targeted ads based on their visited locations. This is privacy critical as users are unaware of their data being collected.

Therefore, in this paper, we set the foundation for uncovering users' private information inferable from spatial maps.

3.2 Problem Statement

We investigate how an adversary can exploit the 3D spatial maps captured by MR devices to infer the indoor location of a user, e.g., a user is in the office or bedroom. We consider the spatial map of a user's environment, $P = \{p_1, p_2, \ldots, p_n\}$, where n is the number of points in the map and each p_i consists of the 3D coordinates of a point. Given P, our goal is to infer the indoor location where the spatial map was captured.

To illustrate, consider the scenario in Figure 2 where a user installs a malicious virtual meeting app on their mixed reality device (e.g., iPad Pro with LiDAR sensor). The app requests camera permission from the user to display avatars of other meeting attendees. The user interacts with the app in their office to conduct a meeting with their colleagues. The malicious app accesses the 3D spatial map of the user's current surroundings and shares it with a remote server.

A remote adversary can exploit this spatial map to infer that the user is at their office and leverage this information to break into the house when the user is not home. The adversary could also exploit this information to gain a fine-grained understanding of the user's identity, routine, and preferences (e.g., when the user wakes up, goes to work, user's hobbies, etc.). Additionally, based on the user's inferred location, an adversary can create a detailed profile of a user to deliver unsolicited personalized ads to generate ad revenue. For example, a virtual meeting app could display ads for office furniture or work productivity tools while the user is in their office.

3.3 Threat Model

We consider an adversary with the goal of inferring the type of user's indoor location (e.g., bedroom, kitchen, office) while a user is using an MR device. The target device can be any MR device, capturing a 3D spatial map via the device camera equipped with a depth sensor or LiDAR scanner, e.g., a smartphone with MR capabilities and HoloLens. We assume the adversary has no prior knowledge about the target user's indoor environment, including its physical location, spatial maps from prior app usage, and location type. The adversary trains the attack model using publicly available datasets that include typical indoor location types [8, 13]. We discuss in Section 8 how the adversary can infer location types not observed during the training process.

An adversary provides the user with an MR app, which accesses the 3D spatial map of the user's surroundings while the user interacts with the app. The adversary can achieve this by (1) distributing the MR app on MR platforms' app stores and third-party MR forums, and (2) deceiving users into installing the MR app via phishing and other social engineering methods. We note that 3D spatial map can be accessed if the camera permission is granted to an app. As MR apps require camera access to deliver their basic functionality, camera permission is given to all MR apps. The app does not require any other permissions, such as permissions for the device location, or the images and videos recorded by the device; additionally, the 3D spatial map accessed by the app does not contain any color or normal vector information. We note that given the adversary deploys a malicious app on the target user's device, it has knowledge of the user's MR device model and its depth-sensing technology. Based on this, the adversary trains an attack model on publicly available datasets collected using similar depth sensors as the target device.

3.4 Design Challenges

C1: Extracting Location Cues from Spatial Maps. Given a 3D spatial map, intuitively, an adversary could train a classifier that extracts high-level features, such as structural and geometric properties (e.g., length and width or floor map) of the user's environment to infer the location. However, this is a challenging task as indoor environments of the same type vary significantly in their structural and geometric properties. For instance, the size of two bedrooms in a user's house may differ. Similarly, the geometric features of different locations may share similarities (e.g., structure of a classroom may share similarities with that of a conference room).

One potential solution to this problem is leveraging the semantic context of indoor locations to discriminate them better. To achieve this, an adversary could train a 3D spatial map model trained either on objects or 3D surfaces present in the environment. Yet, solely using objects or 3D surfaces yields incorrect location inference (demonstrated in Section 7) because, unlike image pixel arrays, a 3D spatial map is an unordered collection of points with a non-uniform point density, which makes detecting objects and 3D surfaces challenging. Therefore, to accurately infer the location, we build a feature extraction and loss minimization pipeline that simultaneously

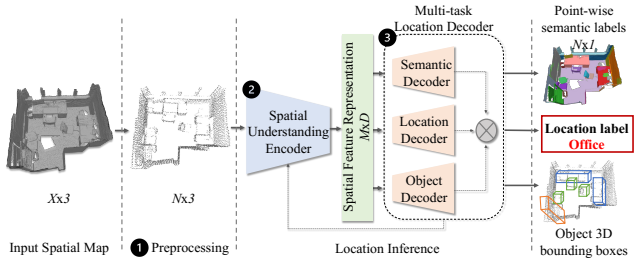


Figure 3: Overview of LocIn attack.

captures both geometric properties and the semantic context of the user's environment.

C2: Invariance to App Usage. Spatial maps captured by MR devices change dynamically in size and viewing angle when a user interacts with different MR apps. For example, while playing an MR game, a user may walk around the room, and while using a shopping MR app, a user may position products to specific locations in a real-world perspective. Hence, to infer the user's location from the captured spatial map, the location inference model must be invariant to spatial map transformations. For instance, translating or rotating the spatial map points should not change the model's location prediction. We leverage 3D point subsampling and hierarchical multi-scale spatial feature learning to overcome this challenge.

C3: Lack of Prior Knowledge and Generalization. To launch location inference attacks in practice, an adversary must be able to infer the target users' location without making any assumptions or prior information about their environments. One naive approach would be collecting spatial maps from a set of users while they are using an MR app and then manually-label them with a set of location labels. However, while this approach may yield an acceptable attack success rate for seen users, it is tedious and time-consuming and may fail to generalize to unseen users due to the bias to specific locations, and produce incorrect results (Section 7). To address this, we use public spatial map datasets with diverse location labels (in addition to the dataset we collected). Additionally, to eliminate the sparse, non-uniform, and noisy nature of such datasets, we leverage a data-driven upsampling method to generate dense points while improving proximity-to-surface and distribution uniformity in the 3D point representations.

4 LocIn Attack Overview

We present LocIn, a location inference attack for mobile mixed reality devices, which exploits 3D spatial maps to identify a user's indoor location. Figure 3 illustrates the overview of LocIn attack. Given that the sparsity of spatial maps varies based on the MR device and device usage duration, we first preprocess the input map by removing outlier points and subsampling the 3D points in the spatial map (❶). Here, we leverage farthest point sampling [47], resulting in a reduced size map with N points (addressing **C2, C3**).

As discussed in Section 3.4, inferring a user's location solely from geometric (i.e., structure and appearance of an environment) or semantic (i.e., objects and surfaces present in the environment) properties of an indoor environment can often lead to ambiguous results. To address this issue, we introduce a new location inference learning representation by combining the geometric and semantic properties of a 3D spatial map. For this, we use a multi-task learning approach and train an end-to-end encoder-decoder [9], which, in turn, successfully infers the user's location (addressing **C1**).

Specifically, we extract the geometric and semantic properties of the user's environment by extending a PointNet++-based hierarchical neural network encoder [59] (❷). The encoder extracts a spatial feature representation of the input spatial map (Z). Z is then fed to LocIn's multi-task location decoder (❸). The multi-task decoder is a composite model consisting of three components: (1) location decoder, (2) object decoder, and (3) semantic decoder. The location decoder consists of a fully-connected neural network classifier trained to predict the type of input spatial map's location. The object decoder extends a 3D object detection network [57] that detects the objects in the user's environment. Lastly, the semantic decoder leverages a segmentation network [59] to provide fine-grained information about the objects and surfaces present in the user's environment.

To ensure that the location decoder integrates the intrinsic patterns from object detection and semantic segmentation, we introduce a unified optimization function that combines the loss functions of the three decoders to train LocIn.

5 LocIn Design

5.1 Spatial Map Preprocessing

LocIn attack aims to infer the location of the user without any prior knowledge about the user's MR device. As different MR devices use different sensors to generate the 3D spatial map of the user's surroundings, the number of points in the 3D spatial map varies from one device to another. For example, the spatial map's point density captured via a HoloLens 2 with depth cameras is different from the spatial map obtained through the iPad's LiDAR scanner. Moreover, spatial maps' point density is largely dependent on the user's app usage. For instance, if the user moves quickly in their environment while interacting with the app, the spatial map has fewer points.

To infer location from spatial maps with varying densities, we transform the input spatial map to a fixed number of 3D points. Given an input map (P) of size $X \times 3$, with a 3D coordinate for each X point, we apply farthest point sampling [47] to the map to generate a transformed map of size $N \times 3$. Specifically, we select a random point and then iteratively sample points that are farthest from the selected samples. This en-

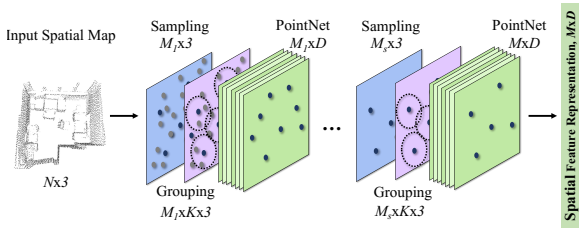


Figure 4: LocIn's spatial understanding encoder architecture based on a hierarchical neural network (PointNet++).

sures that the input spatial map is subsampled to a fixed size for efficient processing while providing sufficient coverage.

5.2 Spatial Understanding Encoder

To infer the user's location, a method is needed to understand the geometric properties and semantic context from the pre-processed spatial map. For this, we convert the input spatial map, P, into a high-level spatial feature representation Z that only encodes information relevant for uniquely identifying the user's location. For instance, for the spatial map of a bedroom, Z could represent features of objects or surfaces in the map that can help differentiate its location from others, e.g., 3D points of a bed and nightstand placed close to a wall.

To learn the spatial feature representation, instead of relying on hand-crafted geometric and semantic features, we leverage network architecture for learning point-wise features from 3D point clouds [58, 59, 72]. While LocIn's approach is not limited to a specific network, we extend PointNet++ [59] as an encoder to learn hierarchical features that preserve spatial localities in the spatial map at different contextual scales

Figure 4 illustrates the architecture of LocIn's encoder. The encoder's PointNet++-based hierarchical structure consists of multiple set abstraction levels with skip connections. At each abstraction level, a subset of points from the input spatial map is selected to ensure efficient computation and processed into a feature vector that represents the local context of the selected points. The set abstraction consists of three main operations: (1) sampling, (2) grouping and (3) PointNet feature extraction.

Iterative Spatial Sampling. The sampling operation in each set abstraction level leverages iterative farthest-point sampling to select a subset of the spatial map's points such that each point in the subset is the most distant from the remaining points in the subset. Therefore, given an input set of points of size $N \times 3$, the sampling operation generates a set of points of size $M \times 3$. In contrast to random sampling, the farthest point sampling ensures that the sampled points provide better coverage of the entire spatial map. This sampling operation is similar to LocIn's preprocessing step (Section 5.1). Yet, it is repeated at each set abstraction level with a decreasing number of samples, followed by the spatial grouping operation.

Spatial Locality Grouping. With the grouping operation, we generate groups of neighboring points for each point selected during the sampling operation. These groups represent local regions of the input spatial map used for feature extraction via PointNet [58]. To identify the neighboring points, we extend ball query algorithm [36] for the grouping operation to find all points within a specified radius of a given point. Ball query algorithm uses a divide-and-conquer approach to build a ball-tree i.e., a binary tree in which each node of the tree represents the set of neighboring points within a specific radius. Starting from the set of 3D points from the sampling layer as its root node, we iteratively build the ball-tree by selecting the farthest point from the centroid of root node points as the left child and the farthest point from this left child as the right child of the root node. The points in the root node are then assigned to the children nodes based on their distance from the node.

The grouping operation transforms the input spatial map of size $N \times 3$ into a set of groups of points of size $M \times K \times 3$. Here, M is the number of groups centered around the points selected via sampling operation, and K is the number of neighboring points found for each selected point. We note that K varies across groups, but the spatial feature vector extraction layer converts variable length groups to a fixed vector size.

Spatial Feature Vector Extraction. For each of the local spatial map regions (groups) extracted via the sampling and grouping operations, we extract a spatial feature vector that encodes the context of the local region through PointNet [58]. We first extract the coordinates of points in each of the M local regions relative to the centroid of the region. The resulting groups of coordinates are then fed as input to PointNet.

Given the set of points for a specific region, PointNet maps the points to a D-dimensional feature vector through a multi-layer perceptron (MLP) network and a max-pooling function Thus, the feature extraction layer of LocIn's encoder returns a spatial feature representation (Z) of size $M \times D$, capturing the local context of various parts of the input spatial map.

5.3 Multi-Task Location Decoder

We translate the problem of inferring location from the feature encoding of the spatial map into a classification task. Given the encoding Z of the user's environment obtained from the spatial encoder, LocIn's location decoder predicts the user's indoor location. For this, LocIn first extracts the high-level geometric features of the user's environment to infer its location. It then combines these geometric features with the map's contextual patterns through object and semantic decoders.

We consider pairs of spatial feature encodings and their location label,

$$\{(z_i, y_i)\}_{i=1}^{n} \sim P^n(z), z_i \in \mathbb{R}^D, y_i \in \Delta^c \tag{1}$$

where c is the number of location classes, and Δ^c is the set of c-dimensional probability vectors. Given this data, our goal is to learn a location decoder

$$f_t = \arg\min_{f \in \mathcal{F}_t} \frac{1}{n} \sum_{i=1}^{n} L_{loc}\left(y_i, \sigma\left(f\left(z_i\right)\right)\right) + \Omega(\|f\|) \tag{2}$$

where $\mathcal{F}_t$ is a class of functions from $\mathbb{R}^D$ to $\mathbb{R}^c$, the function $\sigma : \mathbb{R}^c \to \Delta^c$ is the softmax operation, the function $L_{\text{loc}} : \Delta^c \times \Delta^c \to \mathbb{R}_+$ is the cross-entropy loss

$$L_{\text{loc}}(y, \hat{y}) = -\sum_{k=1}^{c} y_k \log \hat{y}_k \tag{3}$$

and $\Omega : \mathbb{R} \to \mathbb{R}$ is an increasing function as a regularizer.

To infer the location from the spatial encoding, f_t can be implemented as a deep neural network that extracts high-level features of the spatial map useful for predicting its location. However, these features often lack distinctive local or global semantic patterns necessary for uniquely identifying a location (See our evaluation in Section 7). Moreover, the lack of labeled data (i.e., pairs of spatial maps and their location label) makes learning a generalized location classification model for various users challenging.

We observe that indoor environments are uniquely characterized by their semantic context i.e., the types of objects and fine-grained details about surfaces present in the environment. For instance, a bedroom can be uniquely identified because of the presence of a bed, and a kitchen can be identified if a stove is detected in the spatial map.

To integrate contextual information into our learning representation for location decoding, we propose a new composite learning representation for location decoding by leveraging the multi-task learning paradigm in transfer learning [9]:

$$f_s = \arg\min_{f \in \mathcal{F}_s} \frac{1}{n} \sum_{i=1}^{n} \left[\alpha L_{\text{loc}} + \beta L_{\text{obj}} + \gamma L_{\text{sem}} \right] \tag{4}$$

Here L_{obj} and L_{sem} are the loss functions for detecting objects and extracting semantic patterns from the input spatial encoding (detailed in Sections 5.3.1 and 5.3.2).

Through this learning representation, we learn a model to classify the location of a given spatial map while concurrently learning the contextual patterns from objects and surfaces in the map. Our representation is inspired by the teacher-student networks in knowledge distillation where the learning of a student model is guided by the teacher networks [25]. In our case, the location decoder is the student network, while the object and semantic decoders act as teacher networks. Object and semantic decoders benefit from stronger supervision during learning as the labeled data for these tasks includes ground truth for multiple objects and point-wise semantic class labels.We show in Section 7 that this composite learning representation helps improve LocIn's overall accuracy.

5.3.1 3D Object Decoder

The goal of LocIn's object decoder is to localize and recognize 3D objects present in the user's environment as indoor environments are characterized by the objects present within them. This process involves detecting the orientated 3D bounding box for each object from the spatial map as well as predicting the semantic class of each detected object.

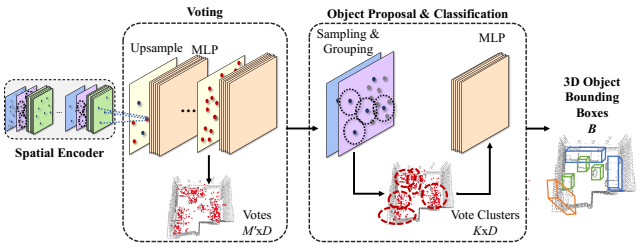


Figure 5: LocIn's object decoder architecture with a deep Hough voting, object proposal, and classification module.

To this end, we extend VoteNet [57], a voting-based network for object detection Figure 5 illustrates the architecture of LocIn's object decoder, consisting of two modules: a voting module and an object proposal and classification module.

Object Voting. The voting module adapts Hough transform [51] to generate votes for points in a 3D spatial map based on their distance to objects' centers. Since depth sensors used by MR devices typically only capture object surfaces, 3D object centers may not be close to any point. For accurate bounding box generation around object centers, VoteNet leverages the Hough voting to sample *seed points* which are close to object centers and generate votes based on their features.

We use the spatial encoding Z of size $M \times D$ as the input to the voting module. To ensure significant coverage of spatial map points for object detection, we first perform upsampling on the M points via multiple feature propagation layers and obtain spatial feature representation for M' points in the original map. The feature propagation layer interpolates the point features of the input points to output points by computing the weighted average of their three nearest input points' features and concatenates features from LocIn's encoder (forwarded through skip-connections) through an MLP network.

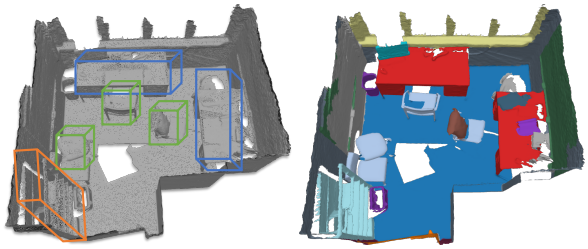
Given the set of upsampled points and their features, $M' \times D$, we generate votes for each point via a shared MLP with fully connected layers. The MLP computes the vote for each point p_i by predicting its offset from an object's center Δp_i through the following regression loss minimization:

$$L_{\text{vote}} = \frac{1}{M_o} \sum_{i} \|\Delta p_i - \Delta p_i^*\| \mathbf{1}[p_i \text{ on object surface}] \tag{5}$$

where M_o is the total number of points lying on the object surface, Δp^* is the ground truth displacement of the point p_i from the object's center it belongs to and $\mathbf{1}[p_i$ on object surface] indicates whether the point lies on an object surface.

The resulting votes represent the semantics of different parts of the objects in the environment.

Object Proposal and Classification. The object proposal and classification module groups and aggregates the votes generated by the voting module to generate object bounding box proposals. It first clusters the votes via uniform sampling and



(a) (b)

Figure 6: An illustration of (a) object detection and (b) semantic segmentation output. The color in (b) represents the semantic label for points in that region.

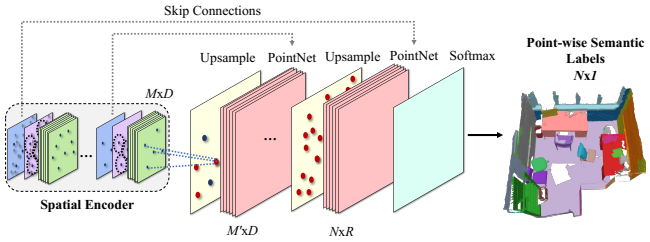


Figure 7: LOCIN's semantic decoder consists of upsampling and PointNet layers that generate point-wise semantic labels.

grouping according to spatial proximity and then processes them through a series of MLP and max-pool layers to generate the bounding box proposals. This generates a set of bounding boxes, B, for various objects in the input spatial map. These bounding boxes are represented as a multi-dimensional vector, including the center coordinates and size of the bounding box.

Object Detection Loss. The object decoder generates the bounding boxes for the detected objects and assigns the semantic class label to each object. Therefore, we train it by optimizing a multi-task loss function consisting of the vote (L_{vote}), objectness ($L_{\text{obj-cls}}$), 3D bounding box estimation (L_{box}), and semantic classification ($L_{\text{sem-cls}}$) losses.

$$L_{\text{obj}} = L_{\text{vote}} + \lambda_1 L_{\text{obj-cls}} + \lambda_2 L_{\text{box}} + \lambda_3 L_{\text{sem-cls}} \tag{6}$$

Objectness loss gauges whether the detected box proposals indeed belong to an object. For this, we categorize the detected object proposals based on their distance from the ground truth object center into positive ($< 0.3m$) and negative proposals ($> 0.6m$). Objectness loss is then computed via a cross-entropy loss normalized by the total number of proposals.

The box loss optimizes the detected parameters of the 3D bounding boxes, i.e., their centers $(x, y, z$ coordinates$)$, size (height, width, and length), and the heading angle along Z-axis. Thus, the loss is defined as a combined regression loss of the detected bounding box's center, heading angle and size.

The semantic classification predicts the object labels through standard cross-entropy loss ($L_{\text{sem-cls}}$).

5.3.2 3D Semantic Decoder

The objects in a given location type help distinguish it from other locations. However, the object decoder is sensitive to the sparsity of the input spatial map. The object decoder also does not consider planar surfaces, e.g., walls, floor, or ceiling of a room, or fails to detect smaller objects (See Figure 6a for an illustration of the object decoder's output).

To extract fine-grained contextual patterns from the user's environment, LOCIN leverages a semantic decoder that performs semantic segmentation for classifying each point in the input map to its semantic object class. For instance, all points belonging to walls are assigned the same label (as shown in Figure 6b). The semantic decoder's output is a set of homogeneous subsets of 3D points, where each subset represents a semantically meaningful object or surface.

Figure 7 shows the architecture of LOCIN's semantic decoder. LOCIN first performs feature upsampling on the spatial encoding, Z, of the input spatial map. This process ensures that the semantic labels are generated for all points in the spatial map. We employ a hierarchical feature propagation strategy, inspired by PointNet++ [59]. The spatial feature representation from LOCIN's encoder of size, $M \times D$, is propagated through a series of upsampling layers where each upsampling layer interpolates feature values, f, for the points in set abstraction level, l, of the encoder, at the coordinates of points in set abstraction level, $l+1$.

We perform the interpolation through the inverse distance weighted average [43] based on the k-nearest neighbors for each point. Specifically, for a given point p in layer $l+1$, the interpolated features are computed as:

$$f_p = \frac{\Sigma_{i=1}^{k} w(p).f_{p_i}}{\Sigma_{i=1}^{k} w(p)}, \text{ where } w(p) = \frac{1}{d(p, p_i)} \tag{7}$$

The interpolated features at each layer, f_p, are then processed through a PointNet layer consisting of convolution, shared fully connected and rectified linear unit (ReLU) activation layers to obtain semantically meaningful features for each point. The final interpolation layer generates R-dimensional semantic features for all N points in the input spatial map. Lastly, we apply a softmax function to the semantic features to obtain the per-point semantic labels S for the input map.

Semantic Segmentation Loss. The semantic segmentation decoder is inherently a classification network that assigns a semantic class to all input spatial map points. To train the semantic decoder, we leverage the cross-entropy loss function. Given pairs of 3D points and their semantic class label $\{(p_i, s_i)\}_{i=1}^{n}$, we compute the semantic loss by minimizing:

$$L_{\text{sem}}(s, \hat{s}) = -\sum_{k=1}^{j} s_k \log \hat{s}_k \tag{8}$$

By minimizing this loss, the semantic decoder assigns the semantic object label to each of the N points in the map.

5.3.3 3D Location Classifier

LocIn leverages the contextual patterns extracted from its object and semantic decoders to perform location classification. LocIn's location classifier predicts the user's location based on the composite learned representation of the object and semantic decoders, as shown in Eq. 4.

The location classifier processes the spatial feature representation Z obtained from LocIn's encoder through an MLP network which concatenates the skip-linked features from the encoder's intermediary set abstraction layers with Z. The concatenated features are then fed to a series of fully connected layers followed by a softmax operation that outputs the probability vectors for the c location classes.

6 Implementation

We implemented LocIn in Python 3.6 with PyTorch 1.2 [53].

LocIn's Network Architecture. We implement LocIn's spatial understanding encoder with four set abstraction layers and consider the output of last layer as the encoder's spatial feature representation (Z). For LocIn's object decoder, we implement two feature propagation layers and concatenate the skip-linked features from the encoder's second, third, and fourth set abstraction layers to the feature propagation layers' output. We use the output of the second feature propagation layer as the input *seed* points for generating votes and then process the votes for object proposal and classification through two MLP networks. We use the smooth-L_1 loss [62] for computing the box regression loss (L_{box}) and PyTorch's CrossEntropyLoss [12] for objectness ($L_{\text{obj-cls}}$) and semantic classification ($L_{\text{sem-cls}}$) losses.

LocIn's semantic decoder consists of four feature upsampling layers that upsample the points and features from LocIn's spatial encoder to $N \times R$ semantic features. Lastly, LocIn's location classifier uses three fully connected layers followed by a drop-out and softmax layer to predict the probability vectors for all location classes.

LocIn Training and Inference. To train LocIn's network, we subsample the input spatial maps' 3D points to a fixed number of $N = 4096$ points. To address spatial maps' rotation and scale variations, we augment our dataset by randomly flipping the maps in the horizontal direction, rotating the 3D points by $\pm 5°$ around the map's z-axis, and scaling the map by a factor of 0.85 to 1.15. We use the Adam optimizer [38] with a batch size of 8 and a learning rate of 0.001. We adopt a grid search to set the optimal values for α, β and γ in LocIn's optimization function (Eq. 4). We also empirically set the object decoder's loss parameters i.e., λ_1, λ_2 and λ_3 such that each component of the loss function is similar in scale.

At inference time, LocIn takes a 3D spatial map collected by an MR app and predicts the user's location through one forward pass of its encoder-decoder network. While LocIn's object and semantic decoders generate the bounding boxes for objects and semantic segmentation of the input spatial map, we only consider the location prediction as LocIn's output.

Table 1: Details of the evaluation dataset.

Dataset	MR Device	# of Location Classes	# of Object Classes	# of Spatial Maps	
				Training	Test
ScanNet	iPad Air2 with depth sensor	13	18	1201	312
ARKitScenes	iPad Pro with LiDAR scanner	9	17	4482	548
Holo3DMaps	HoloLens	5	8	-	20

7 Evaluation

We describe our experience of applying LocIn attack to 3D spatial maps collected from three popular MR devices. We show that LocIn achieves an average accuracy of 84.1% in inferring location on two publicly available spatial map datasets. We also demonstrate that LocIn's attack is generalizable to other MR devices equipped with different depth sensors.

We additionally show the effectiveness of our multi-task learning-based location decoding model through an ablation study and demonstrate LocIn's robustness against different spatial map sizes and point densities. Lastly, we compare LocIn with various baselines and prior work [22] for recognizing indoor locations and show it achieves higher accuracy.

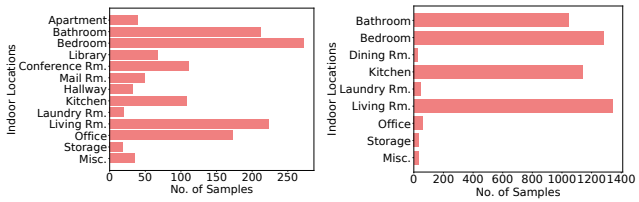
We present LocIn's results by focusing on the following research questions:

RQ1 How effective is LocIn in inferring users' location?
RQ2 How effective is LocIn's multi-task learning decoder?
RQ3 How does the sparsity of the 3D spatial map affect LocIn's performance?
RQ4 What is the impact of a 3D spatial map's size on LocIn's effectiveness?
RQ5 How generalizable is LocIn attack?
RQ6 How does LocIn compare against baseline methods?
RQ7 How does LocIn compare to other spatial attacks?

7.1 Evaluation Setup and Datasets

We evaluate LocIn's effectiveness on spatial maps captured from three MR devices, (1) iPad with depth sensor, (2) iPad with LiDAR scanner, and (3) HoloLens 2. The spatial maps from two iPad versions (with depth sensor and LiDAR scanner) are from two publicly available datasets [8, 13] while we create our dataset of HoloLens 2 spatial maps – Holo3DMaps. Table 1 presents the detailed statistics of the three datasets.

ScanNet Dataset. ScanNet [13] is a richly annotated dataset consisting of 1,513 3D spatial maps captured from 707 distinct indoor environments via a depth sensor attached to an iPad Air2. These spatial maps belong to 13 indoor location types and include annotations for 18 object categories.



(a) (b)

Figure 8: Distribution of the indoor location types across (a) ScanNet and (b) ARKitScenes datasets.

Figure 8a shows the distribution of the number of samples for each location type in the dataset. The "Miscellaneous" location type is assigned to samples that do not distinctively belong to the other location types. We use 1,201 maps from the dataset for training while the remaining for testing. Each spatial map, on average, includes 150K points with a spatial extent of 5.5m × 5.1m × 2.4m.

ARKitScenes Dataset. ARKitScenes [8] is the first indoor 3D spatial map dataset captured via the Apple LiDAR scanner. It consists of 5,048 spatial maps from 1,661 unique indoor environments in real-world homes. The dataset includes ground truth for the oriented 3D bounding boxes of room-defining objects belonging to 17 different object categories. However, the dataset does not contain the ground truth for location labels and point-wise semantic labels for spatial maps.

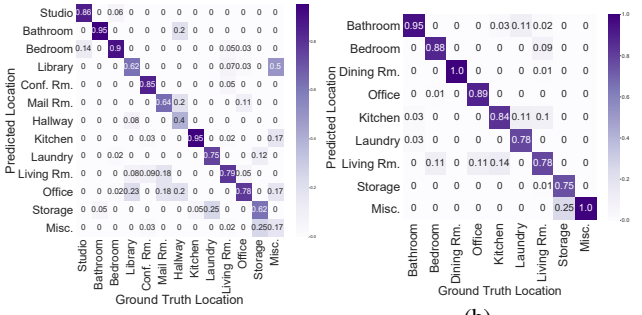
We designed a semi-automated approach to generate the ground truth for location and semantic segmentation labels for the dataset (Detailed in Appendix A). Figure 8b illustrates the sample distribution for each location type after labeling. We use 4,482 maps for training while the remaining for testing.

Holo3DMaps Dataset. To evaluate LocIn's effectiveness on various MR devices, we collected our own dataset of spatial maps using HoloLens 2 (there is no publicly available 3D scene understanding dataset captured using HoloLens). We scanned 20 different indoor environments where HoloLens is typically used. These environments are from 5 location types, including bedroom, living room, office, conference room, and kitchen (4 samples per class). We leveraged Microsoft MRTK's Spatial Mapping [67] to extract the 3D spatial maps of these environments. Two authors of this paper manually annotated the spatial maps to generate the ground truth for location, object detection, and semantic segmentation labels.

Ethical Considerations. We collected Holo3DMaps from public places (e.g., labs and common rooms) and a hotel on a university campus. For each environment, we ensure that no human subjects are present during the data collection process and do not collect any personally identifiable information (PII). For the hotel environment, we received permission from the hotel management to visit unoccupied hotel rooms and suites to collect the maps. We contacted our university's IRB office and got advised that IRB approval is not required since our environments do not include any human subjects and we do not collect any sensitive information.

Table 2: LocIn's overall attack effectiveness.

Dataset	Avg. Accuracy	Avg. Precision	Avg. Recall
ScanNet	81.6%	82%	81.6%
ARKitScenes	85.6%	85.7%	85.6%



(a) (b)

Figure 9: LocIn's confusion matrix on (a) ScanNet and (b) ARKitScenes dataset.

Evaluation Setup. We train the LocIn attack model on the training samples from the ScanNet and ARKitScenes datasets separately. We split each dataset into disjoint training and test sets such that each set's indoor environments are distinct. This splitting ensures that LocIn's reported results are independent of users and prior knowledge about their environments. We subsample each spatial map to 4,096 points through LocIn's preprocessing step. We perform all our experiments on a PC with 32 GB RAM and dual NVIDIA GTX 1080 Ti SLI GPUs. We provide LocIn's implementation details in Appendix 6.

7.2 Overall Effectiveness (RQ1)

We measure the effectiveness of LocIn through three evaluation metrics: average accuracy, precision, and recall. We compute the average accuracy as the number of correctly predicted location types in the test set. We calculate the precision for each location type as the average ratio of correctly predicted spatial maps to the total number of spatial maps classified to that type. We report recall for each location type as the average ratio of the number of correctly predicted spatial maps to the total number of spatial maps of the given type. Table 2 shows LocIn's results for ScanNet and ARKitScenes.

Evaluation Results with ScanNet. LocIn infers the location from the spatial maps with an average accuracy of 81.6% with 82% and 81.6% average precision and recall rate. Figure 9a shows the confusion matrix of the location inference results.

LocIn classifies indoor environments, including distinct object types, correctly with high accuracy. For instance, bedroom, bathroom, and kitchen include unique objects (e.g., bed, sink, and stove) that provide semantic context to LocIn's location decoder and are classified with $>$ 90% accuracy. In contrast, location types that typically lack such distinct ob-

(a) (b)

Figure 10: Examples of spatial maps of the "hallway" location type misclassified by LocIn.

jects have comparatively lower accuracy. For example, spatial maps of "library" type with only chairs and tables are misclassified to "office" since "office" spatial maps include the same object types and share similar geometric structures.

We found that the majority of the spatial maps for location types with low accuracy (e.g., "hallway") included objects commonly found in other location types or no objects, causing LocIn to misclassify them. To illustrate, we present two examples of spatial maps from the "hallway" class in Figure 10. These spatial maps share similarities with office environments; thus, they are misclassified.

Evaluation Results with ARKitScenes. LocIn achieves an average accuracy of 85.6% in inferring the locations of ARKitScenes spatial maps. It classifies the 9 location types with a precision and recall rate of 85.7% and 85.6%. Figure 9b presents the confusion matrix for ARKitScenes results.

The number of spatial maps per class in ARKitScenes is hugely imbalanced (as shown in Figure 8). However, because LocIn leverages the class weights in the cross-entropy loss function for its location classifier during training, it accurately infers the location types with few training samples. Similar to ScanNet, LocIn accurately detects distinct indoor locations (e.g., bathroom, bedroom, kitchen) with high accuracy.

LocIn Inference Time. We evaluate LocIn's inference time by measuring the average time taken to predict the user's location type from spatial maps collected through an iPhone 14 equipped with a LiDAR scanner. Overall, given a spatial map subsampled to 4,096 points, LocIn takes 0.89s on average to predict the user's semantic location. Specifically, LocIn's object decoder takes 0.64s on average to generate the bounding boxes for the detected objects and predict their labels. LocIn's semantic decoder predicts the semantic labels for each point in the spatial map within 0.25s on average. The low inference time of LocIn's multi-task framework allows an adversary to infer a user's location in real-world MR apps.

7.3 Effectiveness of Decoders (RQ2)

To understand how each component of LocIn contributes to its performance, we perform an ablation study by training three models: (1) location classifier without LocIn's multi-task optimization function, (2) LocIn's object decoder with location classifier and (3) LocIn's semantic decoder with location classifier and comparing their performance with LocIn. LocIn's location classifier in (2) and (3) is solely needed to infer the location labels from the detected objects and semantic features. Table 3 shows the average accuracy of LocIn compared to these simplified models.

Table 3: Effect of individual decoders in LocIn's multi-task decoder on its performance.

Dataset	LocIn$_{LOC}$	LocIn$_{OBJ}$	LocIn$_{SEM}$	LocIn
ScanNet	57%	80.1%	78.4%	81.6%
ARKitScenes	58.6%	83.7%	79.7%	85.6%

Location Classification. As discussed in Section 5.3, an adversary could infer a user's location by training a DNN classifier directly on the spatial maps. We compare LocIn to a location classifier (LocIn$_{LOC}$) trained on the spatial maps solely with the cross-entropy loss for classification (See Eq. 3).

Table 3 shows that LocIn$_{LOC}$ without LocIn's multi-task learning optimization function can only achieve 57% and 58.6% accuracy on ScanNet and ARKitScenes datasets. This is because without leveraging context (i.e., object and semantic patterns), the classifier extracts only high-level geometric features (i.e., structure and appearance of the environment). These features are unable to distinguish different indoor locations due to similarities in their structure and appearance, resulting in lower classification accuracy compared to LocIn.

Object Detection. We study the impact of object detection on LocIn's performance by building a model, LocIn$_{OBJ}$, that only leverages the LocIn's object decoder for classification i.e., we only consider L_{loc} and L_{obj} (in Eq. 4) for training.

We found that integrating object detection for location classification significantly improves the accuracy of the location classifier. With object decoder, LocIn$_{OBJ}$ can achieve an average accuracy of 81.9% that is ~24% higher than the model trained without object detection (LocIn$_{LOC}$). This gain in accuracy shows that objects uniquely characterize indoor location types and enable accurate discrimination between them.

Semantic Segmentation. We study how semantic decoder influences LocIn's performance by training a model, LocIn$_{SEM}$, that combines only L_{loc} and L_{sem} in Eq. 4. It improves accuracy by ~22% compared to the location classifier, LocIn$_{LOC}$. The semantic decoder helps LocIn extract fine-grained details (e.g., planar surfaces, sparse/small objects) about the environment that help in location classification.

We note that the accuracy gain with LocIn$_{SEM}$ is 2% less than that from LocIn$_{OBJ}$. This is because predicting pointwise semantic object labels is more prone to errors resulting from sparse or missing points on an object's surface. However, combining the object and semantic decoder in LocIn's

Table 4: LocIn's results with varying sparsity of spatial map.

Dataset	512	1024	2048	4096
ScanNet	71.7%	75.5%	78.4%	81.6%
ARKitScenes	75.6%	79.7%	83.4%	85.6%

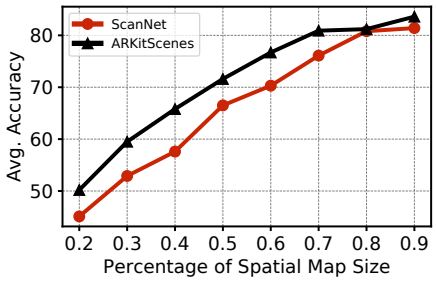


Figure 11: LocIn's performance with varying map size.

multi-task decoder provides higher accuracy in inferring locations than using them individually for location classification. This accuracy gain demonstrates the effectiveness of LocIn's unified multi-task network architecture.

7.4 Parameter Analysis

We evaluate the impact of the input spatial map's point density and size on LocIn's performance.

Spatial Map Sparsity (RQ3). The results reported in the previous sections are achieved on spatial maps with $N = 4096$ points. We now evaluate the impact of varying the number of points, N, subsampled in LocIn's preprocessing module on its effectiveness. Table 4 shows the accuracy of LocIn on spatial maps with varying sparsity.

LocIn achieves more than 70% accuracy on both datasets even when the input spatial maps have fewer points i.e., $N = 512$. We note that even with sparse maps, LocIn's location decoder extracts the environment's structural and semantic properties, sufficient for accurate location classification. This ensures that LocIn infers the user's location even if the map is captured while the user quickly scans their environment.

Spatial Map Size (RQ4). An important factor concerning LocIn's practicality is the input spatial map's size since MR device depth sensors have a limited field of view of the user's environment due to object/surface occlusions or their non-panoramic nature. We define size as the area of the user's environment captured by the MR device.

To this end, we evaluated the ScanNet and ARKitScenes datasets by creating submaps from each spatial map in the test dataset by randomly selecting a point in the map and cropping a bounding box around it. The resulting submaps only include a subset of objects and walls (including the case where no walls are sampled) present in the user's environment. The size of the bounding box is a percentage of the original map's size. We discard submaps with less than $N = 4096$ points.

Table 5: LocIn's performance on Holo3DMaps dataset.

Training Depth Senor	Avg. Accuracy	Avg. Precision	Avg. Recall
Indirect ToF (ScanNet)	85%	85%	86%
LiDAR (ARKitScenes)	45%	45%	60%

Figure 11 presents the average accuracy of LocIn on submaps generated from the two datasets as we vary the percentage of the indoor environment's size captured by the MR device. LocIn achieves an average accuracy of 69.1% in classifying the indoor location type when the spatial map captures only 50% of the environment. The accuracy improves to 77% as the spatial map size increases to 70%. LocIn effectiveness deteriorates as the size of the map decreases since the cropped submaps lack complete semantic and object details necessary for distinguishing the indoor location (e.g., cropped/missing bed in a bedroom's submap).

We note that we perform this evaluation on LocIn model trained on the complete spatial maps of the indoor environments. Hence, one possible approach to improve LocIn's robustness to map size would be to train on cropped submaps.

7.5 Generalizability of LocIn (RQ5)

We perform LocIn's main evaluation on the two publicly available datasets (i.e., ScanNet and ARKitScenes) collected using iPads with two different depth sensors. To evaluate LocIn's generalizability to other MR devices with different depth-sensing technologies, we collected our own dataset (Holo3DMaps) of spatial maps from indoor environments using Microsoft's HoloLens 2 equipped with indirect Time-of-Flight (ToF) depth sensor [27]. We evaluate LocIn's performance on Holo3DMaps through two models: (a) LocIn trained on a dataset collected using indirect Time-of-Flight depth sensor (ScanNet) and (b) LocIn trained on a dataset collected using LiDAR scanner (ARKitScenes).

Table 5 shows LocIn's effectiveness on Holo3DMaps on the two models. LocIn achieves 80% accuracy in inferring user location when trained on the ScanNet dataset. Interestingly, LocIn's accuracy deteriorates when evaluated on the model trained with ARKitScenes dataset. This is due to the resolution differences in the spatial maps captured by HoloLens 2 and iPad with LiDAR scanner. HoloLens 2 employs indirect Time-of-Flight (ToF) depth sensor that illuminates the observed scene with infrared light and uses the reflected light to calculate scene depth [75]. In contrast, iPad's LiDAR scanner (direct ToF) uses timed light pulses to measure scene depth, instead of illuminating the whole scene with modulated light like an indirect ToF sensor [11].

Due to the differences in depth calculation methods, the spatial maps of two devices have inherent differences in their 3D point cloud sparsity. Thus, if LocIn is trained on LiDAR

Table 6: LocIn's comparison with baseline approaches.

Dataset	Base-RF$_{obj}$	Base-RF$_{ground}$	Base-DN	LocIn
ScanNet	29.6%	41.7%	57.1%	81.6%
ARKitScenes	38.8%	44.2%	58.6%	85.6%

spatial maps (ARKitScenes), the features from spatial maps in Holo3DMaps are inconsistent with the features observed during training, causing LocIn to misclassify them. Contrarily, given that both ScanNet and Holo3DMaps are created using indirect ToF sensors, LocIn trained on ScanNet dataset generalizes well to Holo3DMaps. Therefore, an adversary can leverage LocIn trained on one MR device to attack a different device with the same depth-sensing technology.

7.6 Comparison with Baseline (RQ6)

We compare LocIn's effectiveness against three baseline approaches on ScanNet and ARKitScenes datasets. First, we compare LocIn with a naive ML classifier (Base-RF$_{obj}$) that leverages the objects in the user's environment. For this, we predict the semantic labels for objects using VoteNet [57] and generate a histogram of the objects present in a given location type. We then train a Random Forest classifier on these object class histograms to predict the location. We choose Random Forest classifier as it classifies the point cloud based on statistical features from the object histogram. Second, we compare LocIn against another Random Forest classifier trained on the object class histogram generated from ground-truth object semantic labels (Base-RF$_{obj}$). This eliminates the impact of errors in object detection and evaluate how object information without spatial information impacts location classification accuracy. Lastly, we compare LocIn with a baseline deep neural network that takes the spatial representation obtained from PointNet++ [59] as input and processes it through a series of fully connected layers followed by a softmax operation to predict the location class (Base-DN).

Table 6 shows the accuracy of the baseline models in contrast to LocIn. Base-RF$_{obj}$ achieves an accuracy of only 29.6% and 38.8% on ScanNet and ARKitScenes, respectively. This is because Base-RF$_{obj}$ does not consider the spatial and geometric information in the spatial map. We find that the accuracy for location types with distinct objects (e.g., bed in bedroom and toilet in bathroom) is higher than location types with no distinct objects (e.g., office and living room). Moreover, the errors in the object detection model contribute to inaccuracies in the object class histograms and, in turn, the location inference accuracy.

Base-RF$_{ground}$ achieves a higher location inference accuracy of 41.7% on ScanNet and 44.2% on ARKitScenes. While this model eliminates the impact of errors in object detection, it still does not capture the correlation between spatial, geometric, and semantic features of a spatial map. Similarly, Base-DN extracts high-level features, such as structural and geometric properties (e.g., length and width or floor map) of the user's environment. However, since indoor environments of the same type vary significantly in their geometric and structural properties (e.g., size of two bedrooms in a house), it is difficult to predict the specific location type, resulting in a low accuracy of 57.1% and 58.6% on ScanNet and ARKitScenes datasets, respectively. In contrast to these models, LocIn's multi-task learning framework performs feature extraction and loss minimization that simultaneously capture both geometric properties and semantic context of the user's environment to accurately infer the location.

7.7 Comparison with Prior Work (RQ7)

We compare LocIn with a recent work [22], the only prior work that leverages 3D spatial maps from an MR device to infer the user's indoor location. This work extracts 3D spin image features [35] from the input spatial map and employs nearest neighbor distance and deep learning-based 3D place recognizer (PointNetVLAD) [69] to determine if the map's feature vector matches with one of the maps from the user's previously visited locations. However, their attack has limited practicality as it assumes that the attacker has access to a labeled dataset of 3D spatial maps of the target user's indoor locations. Therefore, it only targets a specific user and her previously visited locations.

To evaluate how this work compares against LocIn in a realistic attack scenario (no prior knowledge about a user is available), we implement its DNN place recognizer [69] to extract global features from 3D spatial maps. Since our goal is to infer the location type, we replace its final Euclidean distance-based feature matching step with our location classifier that processes the spatial feature vector through an MLP followed by a softmax operation. We train this model on the ScanNet dataset with the same training parameters as described in [69]. It achieves an accuracy of 50.1% on ScanNet's test dataset which is much lower compared to LocIn's 81.6% accuracy. This significant difference in accuracy shows that LocIn is more effective in inferring users' location from spatial maps without any prior knowledge about the user's location.

8 Limitations and Discussion

Inferring Location of Complex Environments. LocIn can infer users' location from 3D spatial maps in various indoor environments with common objects (Section 7.2). However, if a user's environment includes unique objects or very few objects or planar surfaces, LocIn's effectiveness will mainly rely on its location classification decoder. This is because LocIn's object and semantic segmentation decoders cannot extract meaningful patterns from the input map. We conducted an additional experiment to evaluate LocIn's effectiveness on spatial maps where no objects are present in the user's

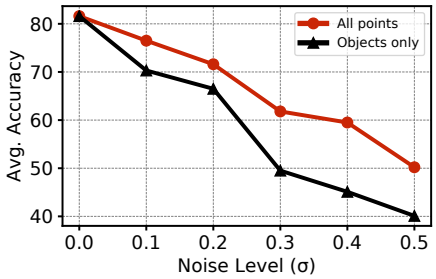


Figure 12: LocIn's performance with varying noise levels.

environment. In this case, LocIn infers the users' environment based on its geometric properties with an accuracy of 59.1%.

Generalization to Various MR Devices. We demonstrate LocIn attack on three popular MR devices that leverage depth cameras or LiDAR sensors to capture the spatial map of a user's environment. However, spatial maps from different MR devices have varying sparsity and non-uniform point density because various MR devices employ different sensors and depth calculation techniques to generate the 3D spatial maps [58, 75]. For instance, in contrast to LiDAR scanner-based devices, devices leveraging ToF depth cameras (e.g., smartphones, HoloLens) build spatial maps based on the reflection of light pulses emitted by the device. As a result, they experience difficulty in capturing a complete 3D representation of reflective surfaces and objects, such as mirrors, polished metal, or very dark surfaces, producing more sparse spatial maps. Although we show that LocIn attack is transferrable across devices that leverage the same depth sensor type, this inherent difference in spatial map properties requires building a specialized model for each MR device.

Predicting Unknown Location Types. We demonstrate in Section 7 that LocIn can effectively identify 13 indoor location types where MR devices are typically used. LocIn, however, is limited to inferring location types observed during its training process. Thus, if the user interacts with their MR device in a location not included in the LocIn's training, LocIn cannot correctly infer the location from the spatial map. An adversary can leverage outlier detection techniques to detect samples out of distribution from the known location classes [23, 74] and collect a more comprehensive dataset with diverse labels for training LocIn.

LocIn Counter Measures. One possible defense against LocIn is to limit the MR app's access to the raw 3D spatial maps and only share privacy-preserving features (e.g., planar surfaces or 3D points of a user-defined region) to enable MR content. Yet, this would affect MR apps' functionality as they rely on detailed spatial understanding to enable MR content, requiring further investigation into its usability for MR apps.

Another possible defense is to inject noise into the 3D spatial maps shared with the MR apps to force LocIn to misclassify the user's location. To test this defense, we conducted two sets of experiments on the ScanNet dataset. First, we applied random perturbations to all points in the spatial maps in our test dataset by adding Gaussian noise to their points' 3D coordinates and predicted their location using LocIn trained on raw spatial maps. Second, we applied random perturbations of varying intensity only to points belonging to objects present within the spatial maps. In both cases, we change the noise intensity by changing the standard deviation (σ) of the added noise. Figure 12 presents the change in accuracy with varying noise for both experiments. We observe that LocIn's accuracy drops to 61% when all points are perturbed in the spatial maps by a noise level of $\sigma = 0.3$. Similarly, when similar noise is added to only objects, LocIn's accuracy deteriorates to 49.5%. This decrease in accuracy occurs since noisy points in the spatial map make it difficult to detect the objects and semantic features of the user's environment.

Although noise injection reduces LocIn's effectiveness, its feasibility is limited as the perturbed spatial maps reduce usability for the MR apps. For instance, MR apps leverage the spatial map to localize a user in its environment [55]; thus, a perturbed spatial map may result in erroneous localization results, affecting the apps' functionality. Moreover, implementing this defense requires evaluating different spatial map usage scenarios to ensure app functionality is not affected.

9 Related Work

Privacy Leakage in Mobile Mixed Reality. Recent works have exposed attacks that leverage the multi-modal sensors on MR devices, similar to privacy leakages in other mobile devices [15, 18, 48]. For instance, a line of work proposed virtual keystroke detection side-channel attacks on MR devices by exploiting the channel state information (CSI) of WiFi signals [41], headset motion sensors [44, 45] and IMU sensors on MR device hand controllers [2]. Another work [65] proposed an eavesdropping attack through motion sensors on head-mounted MR devices to detect facial movements for inferring human speech. In this paper, we show a new privacy leakage from MR devices by exploiting the 3D spatial maps to infer the user's location type.

Location Inference Attacks. Several works exploit radio frequency (RF) signals emitted by commodity devices to infer users' location [1, 78]. However, these approaches require a physically proximate attacker to deploy sniffing devices near users. In contrast, LocIn attack operates remotely without requiring any additional devices or information about the user. Another line of attacks exploits mobile sensors, e.g., microphone [52] and IMU sensors [39], to infer users' location. Yet, these works only localize a user with respect to the mobile device and require an indoor map of the environment.

Previous approaches have also investigated indoor location inference from images and videos through hand-crafted features and deep learning models [16, 24, 70, 76, 77]. Yet,

these attacks are ineffective in various scenarios. First, these attacks are sensitive to lighting and occlusion, decreasing their location inference accuracy in low-lighting conditions or when objects are partially obscured. Second, the accuracy of these attacks is influenced by different camera orientations/viewpoints while capturing images/videos. Lastly, given users' privacy concerns surrounding apps' access to camera images and videos, several privacy-preserving approaches attempt to eliminate location attacks through visual data [32, 37, 63] and various MR devices limit apps' access to image and video data. In contrast, LocIn exploits spatial maps, shared with apps to enable device localization, to infer locations from different camera orientations/viewpoints regardless of low lighting and occlusion.

A recent work [22] leveraged spatial data on MR devices to recognize users' previously visited locations. While this work, similar to LocIn, exploits the spatial data on MR devices to infer private user information, its practicality is limited as it only recognizes a specific user's previously visited indoor locations. Contrarily, LocIn infers a user's location without any prior information through its semantic aware multi-task network and generalizes well to unseen users (Section 7.7).

Multi-task Learning on 3D Data. Several prior works have leveraged multi-task learning for scene understanding tasks (e.g., object detection, semantic segmentation, object classification) on 3D data [29, 40, 54, 57, 71]. Previous works [54, 71] simultaneously learned embedded features for 3D object instance segmentation and semantic segmentation through a combined loss function. A recent work [40] learned the shape of 3D objects and their labels simultaneously based on objects' curvature. A line of work [29, 46] used multi-task learning to improve a robot's ability to recognize a scene by combining color and geometric properties of 3D data. In contrast, LocIn leverages 3D objects and semantic context of a user's environment from 3D spatial data to infer its location.

10 Conclusions

In this paper, we present LocIn, a new location inference attack on mixed reality (MR) devices via 3D spatial data. LocIn exploits the 3D spatial maps accessible to MR apps to extract contextual patterns from a user's environment and infer their location. It leverages a multi-task learning approach to train an end-to-end encoder-decoder architecture that integrates these contextual patterns into a classification network for predicting users' location. Our evaluation on spatial maps collected from three MR devices demonstrates that LocIn can effectively infer an MR user's location type.

Acknowledgments

We thank our shepherd and the anonymous reviewers for their comments and suggestions. This work has been partially supported by the National Science Foundation (NSF) under grant CNS-2144645 and Google's ASPIRE Award. Any findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF or Google.

References

[1] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. Peek-a-boo: I see your smart home activities, even encrypted! In *Conference on Security and Privacy in Wireless and Mobile Networks*, 2020.

[2] Abdullah Al Arafat, Zhishan Guo, and Amro Awad. Vr-spy: A side-channel attack on virtual key-logging in vr headsets. In *IEEE Virtual Reality and 3D User Interfaces (VR)*, 2021.

[3] Apple unveils new ipad pro with breakthrough lidar scanner. https://www.apple.com/newsroom/2020/03/apple-unveils-new-ipad-pro-with-lidar-scanner-and-trackpad-support-in-ipados/, 2023. [Online; accessed 01-May-2023].

[4] Verifying device support and user permission. https://developer.apple.com/documentation/arkit/verifying_device_support_and_user_permission, 2023. [Online; accessed 01-May-2023].

[5] Arcore. https://developers.google.com/ar/, 2023. [Online; accessed 01-May-2023].

[6] Enable arcore. https://developers.google.com/ar/develop/java/enable-arcore, 2023. [Online; accessed 01-May-2023].

[7] Arkit. https://developer.apple.com/augmented-reality/, 2023. [Online; accessed 01-May-2023].

[8] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Yuri Feigin, Peter Fu, Thomas Gebauer, Daniel Kurz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. In *Conference on Neural Information Processing Systems*, 2021.

[9] Rich Caruana. Multitask learning. *Machine learning*, 1997.

[10] School vr subjects: Chemistry science resources. https://www.classvr.com/vr-ar-resources/science-chemistry-vr-teaching-resources/, 2023. [Online; accessed 01-May-2023].

[11] Ilya Chugunov, Seung-Hwan Baek, Qiang Fu, Wolfgang Heidrich, and Felix Heide. Mask-tof: Learning microlens masks for flying pixel correction in time-of-flight imaging. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[12] Cross entropy loss. https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, 2023. [Online; accessed 01-May-2023].

[13] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] Jaybie A De Guzman, Kanchana Thilakarathna, and Aruna Seneviratne. Security and privacy approaches in mixed reality: A literature survey. *ACM Computing Surveys (CSUR)*, 2019.

[15] Paula Delgado-Santos, Giuseppe Stragapede, Ruben Tolosana, Richard Guest, Farzin Deravi, and Ruben Vera-Rodriguez. A survey of privacy vulnerabilities of mobile device sensors. *ACM Computing Surveys (CSUR)*, 2022.

[16] Dapeng Du, Limin Wang, Huiling Wang, Kai Zhao, and Gangshan Wu. Translate-to-recognize networks for rgb-d scene recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] Russell A Epstein and Chris I Baker. Scene perception in the human brain. *Annual review of vision science*, 2019.

[18] Habiba Farrukh, Tinghan Yang, Hanwen Xu, Yuxuan Yin, He Wang, and Z Berkay Celik. S3: Side-channel attack on stylus pencil through sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2021.

[19] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.

[20] Jaris Gerup, Camilla B Soerensen, and Peter Dieckmann. Augmented reality and mixed reality for healthcare education beyond surgery: an integrative review. *International journal of medical education*, 2020.

[21] Jaybie A de Guzman, Kanchana Thilakarathna, and Aruna Seneviratne. A first look into privacy leakage in 3d mixed reality data. In *European Symposium on Research in Computer Security*, 2019.

[22] Jaybie Agullo de Guzman, Aruna Seneviratne, and Kanchana Thilakarathna. Unravelling spatial privacy risks of mobile mixed reality data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2021.

[23] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations (ICLR)*, 2016.

[24] Luis Herranz, Shuqiang Jiang, and Xiangyang Li. Scene recognition with cnns: objects, scales and dataset bias. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[25] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning and Representation Learning Workshop*, 2015.

[26] Holoanatomy software suite. https://case.edu/holoanatomy/, 2023. [Online; accessed 01-May-2023].

[27] Hololens 2 hardware. https://www.microsoft.com/en-us/hololens/hardware, 2023. [Online; accessed 01-May-2023].

[28] Jinhan Hu, Andrei Iosifescu, and Robert LiKamWa. Lenscap: split-process framework for fine-grained visual privacy control for augmented reality apps. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2021.

[29] Shengyu Huang, Mikhail Usvyatsov, and Konrad Schindler. Indoor scene recognition in 3d. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[30] Charles E Hughes, Christopher B Stapleton, Darin E Hughes, and Eileen M Smith. Mixed reality in education, entertainment, and training. *IEEE computer graphics and applications*, 2005.

[31] Ikea place. https://apps.apple.com/us/app/ikea-place/id1279244498, 2023. [Online; accessed 01-May-2023].

[32] Suman Jana, David Molnar, Alexander Moshchuk, Alan Dunn, Benjamin Livshits, Helen J Wang, and Eyal Ofek. Enabling {Fine-Grained} permissions for augmented reality applications with recognizers. In *USENIX Security Symposium*, 2013.

[33] Suman Jana, Arvind Narayanan, and Vitaly Shmatikov. A scanner darkly: Protecting user privacy from perceptual applications. In *IEEE symposium on security and privacy (S&P)*, 2013.

[34] Blooma John and Nilmini Wickramasinghe. A review of mixed reality in health care. *Delivering Superior Health and Wellness Management with IoT and Analytics*, 2020.

[35] Andrew E Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 1999.

[36] Ashraf M Kibriya and Eibe Frank. An empirical comparison of exact nearest neighbour algorithms. In *European conference on principles of data mining and knowledge discovery*, 2007.

[37] Y Kim, S Boorboor, A Rahmati, and A Kaufman. Design of privacy preservation system in augmented reality. In *IEEE Symposium on Visualization for Cyber Security*, 2021.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2014.

[39] Swarun Kumar, Stephanie Gil, Dina Katabi, and Daniela Rus. Accurate indoor localization with zero start-up cost. In *Annual international conference on Mobile computing and networking*, 2014.

[40] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3d instance segmentation via multi-task metric learning. In *IEEE International Conference on Computer Vision (CVPR)*, 2019.

[41] Zhen Ling, Zupei Li, Chen Chen, Junzhou Luo, Wei Yu, and Xinwen Fu. I know what you enter on gear vr. In *IEEE Conference on Communications and Network Security (CNS)*, 2019.

[42] Shaopeng Liu, Guohui Tian, and Yuan Xu. A novel scene classification model combining resnet based transfer learning and data augmentation with a filter. *Neurocomputing*, 2019.

[43] George Y Lu and David W Wong. An adaptive inverse-distance weighting spatial interpolation technique. *Computers & geosciences*, 2008.

[44] Shiqing Luo, Xinyu Hu, and Zhisheng Yan. Holologger: Keystroke inference on mixed reality head mounted displays. In *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2022.

[45] Ülkü Meteriz-Yıldıran, Necip Fazıl Yıldıran, Amro Awad, and David Mohaisen. A keylogging inference attack on air-tapping keyboards in virtual environments. In *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2022.

[46] Yuhang Ming, Xingrui Yang, Guofeng Zhang, and Andrew Calway. Cgis-net: Aggregating colour, geometry and implicit semantic features for indoor place recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[47] Carsten Moenning and Neil A Dodgson. Fast marching farthest point sampling. Technical report, University of Cambridge, Computer Laboratory, 2003.

[48] Reham Mohamed, Habiba Farrukh, Yidong Lu, He Wang, and Z Berkay Celik. istelan: Disclosing sensitive user information by mobile magnetometer from finger touches. *Proceedings on Privacy Enhancing Technologies*, 2023.

[49] Mixed reality market. https://www.fortunebusinessinsights.com/industry-reports/mixed-reality-market-101783, 2023. [Online; accessed 01-May-2023].

[50] Mrtk. https://github.com/Microsoft/MixedRealityToolkit-Unity, 2023. [Online; accessed 01-May-2023].

[51] Priyanka Mukhopadhyay and Bidyut B Chaudhuri. A survey of hough transform. *Pattern Recognition*, 2015.

[52] Rajalakshmi Nandakumar, Alex Takakuwa, Tadayoshi Kohno, and Shyamnath Gollakota. Covertband: Activity information leakage using music. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2017.

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 2019.

[54] Quang-Hieu Pham, Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[55] Francesco Pittaluga, Sanjeev J Koppal, Sing Bing Kang, and Sudipta N Sinha. Revealing scenes by inverting structure from motion reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[56] Pokemon go. https://pokemongolive.com/en/, 2016. [Online; accessed 01-May-2023].

[57] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[58] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[59] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 2017.

[60] Nisarg Raval, Animesh Srivastava, Kiron Lebeck, Landon Cox, and Ashwin Machanavajjhala. Markit: Privacy markers for protecting visual secrets. In *ACM International joint conference on pervasive and ubiquitous computing*, 2014.

[61] Nisarg Raval, Animesh Srivastava, Ali Razeen, Kiron Lebeck, Ashwin Machanavajjhala, and Lanodn P Cox. What you mark is what apps see. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2016.

[62] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 2015.

[63] Franziska Roesner, David Molnar, Alexander Moshchuk, Tadayoshi Kohno, and Helen J Wang. World-driven access control for continuous sensing. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014.

[64] Somaiieh Rokhsaritalemi, Abolghasem Sadeghi-Niaraki, and Soo-Mi Choi. A review on mixed reality: Current trends, challenges and prospects. *Applied Sciences*, 2020.

[65] Cong Shi, Xiangyu Xu, Tianfang Zhang, Payton Walker, Yi Wu, Jian Liu, Nitesh Saxena, Yingying Chen, and Jiadi Yu. Face-mic: inferring live speech and speaker identity via subtle facial dynamics captured by ar/vr motion sensors. In *International Conference on Mobile Computing and Networking (MobiCom)*, 2021.

[66] Snap ar. https://ar.snap.com/en-US/lens-studio, 2023. [Online; accessed 01-May-2023].

[67] Spatial mapping. https://learn.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping, 2023. [Online; accessed 01-May-2023].

[68] Robert Templeman, Mohammed Korayem, David J Crandall, and Apu Kapadia. Placeavoider: Steering first-person cameras away from sensitive spaces. In *Networks and Distributed Systems Security (NDSS)*, 2014.

[69] Mikaela Angelina Uy and Gim Hee Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[70] Anran Wang, Jianfei Cai, Jiwen Lu, and Tat-Jen Cham. Modality and component aware feature fusion for rgb-d scene classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[71] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively segmenting instances and semantics in point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[72] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions On Graphics (ToG)*, 2019.

[73] Nan Wu, Ruizhi Cheng, Songqing Chen, and Bo Han. Preserving privacy in mobile spatial computing. In *Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2022.

[74] Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Ali Ahmadzadeh, Xinyu Xing, and Gang Wang. {CADE}: Detecting and explaining concept drift samples for security applications. In *USENIX Security Symposium*, 2021.

[75] Yunfan Zhang, Tim Scargill, Ashutosh Vaishnav, Gopika Premsankar, Mario Di Francesco, and Maria Gorlatova. Indepth: Real-time depth inpainting for mobile augmented reality. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2022.

[76] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. *Advances in neural information processing systems*, 2014.

[77] Hongyuan Zhu, Jean-Baptiste Weibel, and Shijian Lu. Discriminative multi-modal feature fusion for rgbd indoor scene recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[78] Yanzi Zhu, Zhujun Xiao, Yuxin Chen, Zhijing Li, Max Liu, Ben Y Zhao, and Haitao Zheng. Et tu alexa? when commodity wifi devices turn into adversarial motion sensors. In *Network and Distributed Systems Security (NDSS)*, 2020.

A Labeling ARKitScenes Dataset

We describe our approach for generating ground truth labels for location type and point-wise semantic labels for the ARKitScenes dataset [8].

Table 7: Location type to object mapping.

Location Type	Object Type
t_0 : Living Room	sofa, fireplace
t_1 : Bedroom	bed
t_2 : Kitchen	stove, dishwasher, oven, refrigerator
t_3 : Bathroom	bathtub, toilet

Location Type Labels. We adopted a semi-automated annotation process to label each spatial map in ARKitScenes with its location type, Since ARKitScenes includes ground truth for the objects (including bounding boxes and object semantic labels), we leverage the fact that objects uniquely characterize indoor environments to assign an initial location label to each spatial map. Specifically, two authors of this paper developed a mapping (M) between typical indoor environments and the objects that uniquely identify them (Table 7). For instance, a bedroom must have a bed, and a kitchen must have a stove. This mapping includes four location types commonly observed in real-world homes where ARKitScenes is collected and a subset of objects from the 17 objects types in the ARKitScenes dataset. We assigned an label to each spatial map through the function $l = \text{argmax}(t_0, t_1, t_2, t_3)$ where

$$t_i = \Sigma(u_0, u_1, \ldots, u_m), \text{ where } u_j = \frac{k}{U} \tag{9}$$

Here, k is the number of objects of type u_j in the spatial map, m is the number of object types present in the mapping for location type (t_i), and U is the total number of objects in the input spatial map that belong to location type t_i in M.

Two authors then manually inspected and verified the initial labels by visualizing the spatial map and its associated images available in the dataset. We found that among 5,048 spatial maps in the dataset, 256 maps did not include any objects in the location-object type mapping in Table 7 and hence could not be labeled in the initial labeling process. The

two authors manually labeled these 256 samples individually and assigned them labels following the location types in the ScanNet dataset. The authors then met to discuss and reconcile differences. We assigned the "Miscellaneous" label to samples for which no conclusive location type could be derived from the spatial map and its corresponding images.

Semantic Segmentation Labels. We used the ground truth for 3D objects in the maps to assign the semantic label to each point based on the object the point belonged to. For instance, we assigned the semantic label "bed" to all points within the bounding box of the bed object in a given spatial map. This approach, however, only considers points that belong to one of the 17 object types annotated in the ARKitScenes dataset. These types do not include planar surfaces such as walls, floors, and ceilings. Hence, to annotate points on these planar surfaces, we adopted a semi-automated annotation approach.

We first identified an estimate of all planar surfaces in a given spatial map through Random Sampling Consensus (RANSAC) [19] algorithm for plane detection. For this, we employed an iterative procedure that randomly samples a subset of points from an input spatial map and fits a plane equation on these points. The number of points that satisfy the plane equation (inliers) in each iteration is used to calculate a confidence score for the plane equation. We then marked the plane equation, which achieves the highest confidence score as a planar surface in the spatial map. We used this process to identify the horizontal and vertical planar surfaces (along x and y axes of the spatial maps) only because these planes represent the walls, floors, and ceiling in the spatial maps.

Two authors of the paper manually inspected the plane detection output and corrected errors in the point-wise semantic labels. Through this procedure, we annotated all spatial maps in the dataset and assigned the points to one of 20 object types (17 objects in ARKitScenes and wall, floor and ceiling).