Deep Learning based Wireless Localization for Indoor Navigation

This is a draft of the accepted paper in Mobicom 2020

ABSTRACT

Location services, fundamentally, rely on two components: a mapping system and a positioning system. The mapping system provides the physical map of the space, and the positioning system identifies the position within the map. Outdoor location services have thrived over the last couple of decades because of well-established platforms for both these components (e.g. Google Maps for mapping, and GPS for positioning). In contrast, indoor location services haven't caught up because of the lack of reliable mapping and positioning frameworks, as GPS is known not to work indoors. Wi-Fi positioning lacks maps and is also prone to environmental errors. In this paper, we present DLoc, a Deep Learning based wireless localization algorithm that can overcome traditional limitations of RF-based localization approaches (like multipath, occlusions, etc.). DLoc uses data from the mapping platform we developed, MapFind, that can construct location-tagged maps of the environment. Together, they allow off-the-shelf Wi-Fi devices like smartphones to access a map of the environment and to estimate their position with respect to that map. During our evaluation, MapFind has collected location estimates of over 100 thousand points under 8 different scenarios across two different spaces covering 2000 sq. Ft. DLoc outperforms state-of-the-art methods in Wi-Fi-based localization by 80% (median & 90^{th} percentile) across the 2000 sq. ft. spanning two different spaces.

CCS CONCEPTS

• **Computing methodologies** → **Tracking**; **Structured outputs**; **Cost-sensitive learning**.

KEYWORDS

WiFi, Path planning, Indoor Localization, Deep Learning

ACM Reference Format:
This is a draft of the accepted paper in Mobicom 2020. 2020. Deep Learning based Wireless Localization for Indoor Navigation. In *The 26th Annual International Conference on Mobile Computing and Networking (MobiCom '20), September 21–25, 2020, London, United Kingdom.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3372224.3380894

1 INTRODUCTION

The introduction of GPS and GPS-referenced public maps (like Google Maps, Open Street Maps, etc. [4, 14, 19, 42]) has completely transformed the outdoor navigation experience. In spite of significant interest from industry and academia, indoor navigation lacks further behind [25, 41]. We cannot use our smartphones to navigate to a conference room in a new building or to find a product of interest in a shopping mall. This is primarily because, unlike GPS, indoor navigation [1] lacks both reliable positioning and accurate indoor maps. First, while recent work [34, 35, 53, 57, 59, 67] has successfully built methods to locate smartphones indoors using WiFi signals to median accuracies of tens of centimeters, the errors are much larger (3-5 m) in multipath challenged environments that have multiple reflectors. These errors are reflected in the high 90 and 99-percentiles errors reported by the current systems. These high errors mean that people can be located in entirely different rooms or different hallways when walking indoors. Second, location-referenced indoor maps are scarcely available. In few cases, like airports and shopping malls, Google and a few other providers [18] create floor plans, but these floor plans are manually created and often need to be updated as floor plans change and they lack details such as cubicles, furniture, etc.

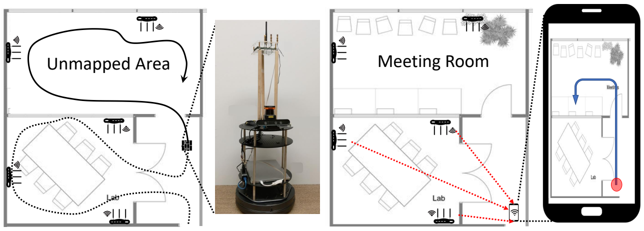


Figure 1: Overview: MapFind (left) is an autonomous platform that maps an indoor environment while collecting wireless channel data. The platform generates a detailed map of the environment and collects training data for DLoc. DLoc uses the training data to learn a model to accurately localize users in the generated map.

In this paper, we aim to solve both challenges. We propose a data-driven approach for indoor positioning that can implicitly model environmental effects caused by reflectors and obstacles, and hence significantly improve indoor positioning performance. In doing so, we build on recent trends in deep learning research that combines domain knowledge and neural networks to solve domain-specific problems. Specifically, we build a system, DLoc, that leverages existing research in indoor positioning in combination with neural networks to deliver state-of-the-art indoor positioning performance. We further developed an automated mapping platform, MapFind. MapFind is equipped with a LIDAR, and odometer to leverage SLAM (simultaneous localization and mapping) for creating indoor maps with minimal human effort. In addition, we deploy a WiFi device on MapFind to automatically generate labeled data for training DLoc. An overview of our system design is shown in Fig. 1.

Before we delve deeper into our design, let us briefly summarize how WiFi-based indoor positioning works today. WiFi access points measure WiFi signals emitted by a smartphone (see Fig. 2) and convert these signal measurements to location estimates in a two-step process. First, we convert wireless signals received at multiple antennas on an access point to a physical parameter, like the angle of arrival of the signal. This transform is independent of any environmental variables, and is (in most cases) a change of signal representation performed using some variant of the Fourier transform. Second, this physical information is converted into a location estimate using geometric constraints of the environment like the location of access points. In the absence of reflectors and obstacles (i.e. free space), this step can easily be performed using triangulation of line-of-sight paths. However, when the direct path

[1] GPS alone does not work in indoor scenarios [40, 49]

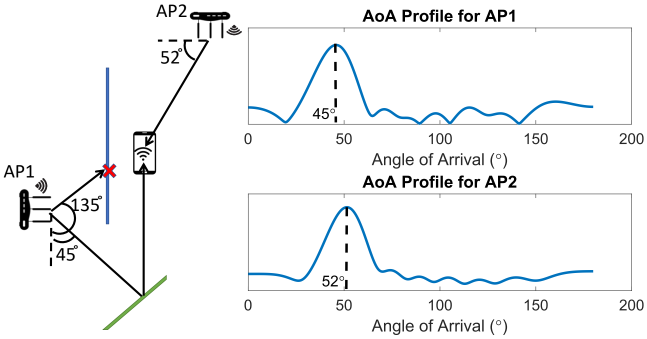


Figure 2: Traditional Localization Approach: WiFi signals transmitted by a smartphone are measured at multi-antenna access points. The access points infer angle of arrival information. However, in the absence of a direct path, this information is erroneous and can lead to large errors (3 to 5m).

between the client and the access point is blocked, this step leads to large errors in location estimates.

Intuitively, in the above example, we can identify the accurate location of the smartphone if we knew the exact size, shape, and location of the reflector. More generally, having an accurate model of the environment allows us to enable accurate positioning even when the line-of-sight path is blocked or when there are many reflectors present. However, obtaining this information about the environment from WiFi signal measurements is very challenging. This is primarily because radio signals undergo a complex combination of diffraction, reflection, and refraction when they interact with objects in the environment. Modelling these competing effects on wireless signal measurements and then using the measurements to build an explicit model of the environment is an unsolved (and extremely challenging) problem.

Instead, we use a neural network to build an implicit model of the environment by observing labelled examples. By observing how the environment impacts the relationship between wireless signals and ground truth locations, a neural network can learn an implicit representation of the relationship. Then, it can use this representation to identify locations of unlabelled points. Based on this intuition, we build DLoc. In doing so, we solve three key challenges:

(i) Incorporating Domain Knowledge into the Neural Network Design: In designing DLoc, we accomplish two objectives. First, we want the neural network to build on the decades of ground breaking WiFi positioning research [7, 11, 34, 53, 57, 64, 65, 67]. Second, we wish to leverage the recent advances in deep learning research, especially with respect to the usage of convolutional neural networks (CNNs) and 2D image based techniques in computer vision[28, 38, 52, 72, 76]. To meet these objectives, we represent the input to the neural network as a two-dimensional likelihood heatmap that represents the angle-of-arrival and time-of-flight (or distance) information (as shown in Fig. 3). The output of the network is represented as a two-dimensional Gaussian centered on the target location. This representation allows us to plug in state-of-the-art WiFi positioning algorithms to generate the input heatmaps. It also lets us frame the localization problem as an image translation problem, enabling us to access the vast array of image translation tools developed in deep learning research.

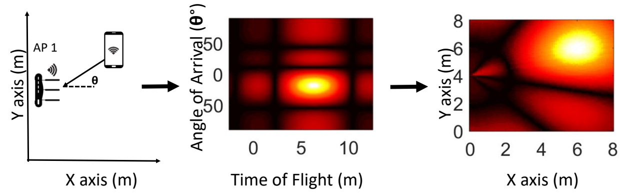


Figure 3: Input Representation: We use 2D heatmaps (center) to encode time-of-flight and angle-of-arrival information computed using state-of-art algorithms. These heatmaps are translated to the global Cartesian space (right) via polar to Cartesian conversion to encode the access point location.

(ii) Consistency over Access points: The standard approach to image translation problems is to use an encoder-decoder network. However, this approach won't directly apply to our scenario. Recall, our objective is to model the impact of objects in the environment on wireless signals. To do so, the network must see these objects appear at the same location consistently across multiple access points. However, this consistency requirement is violated because commercial WiFi devices have random time-of-flight offsets (due to lack of time synchronization between clients and access points). These offsets at a single access point could shift objects by distances as large as 10 to 15 meters for that access point. Thus an object that appears to be at the center for one access point will appear to be at the edge for another access point. This representation will change across different training samples because these time-offsets vary per packet. So, we need to teach our network about these offsets and enforce consistency across the access-points. To do so, we model our network as a single encoder, two decoder architecture. The first decoder is responsible for enforcing consistency across access points. The second decoder can then identify the correct location by leveraging the consistent model of the environment.

(iii) Automated and Mapped Training Data Collection: As is well-documented, deep learning approaches require a large amount of training data to work. To automate the process of data collection, we build a platform, MapFind, that uses a robot equipped with LIDAR, camera, and odometry to collect ground truth location estimates for corresponding wireless channels. To further optimize the data collection process, we build a new path planning algorithm that can minimize the time required to collect the optimal training data for DLoc. Our path planning algorithm ensures that we can map an environment and collect training data samples within 20mins for a 1500 sq. ft. space, while manual data collection for the same would take about 17hrs. This enables us to generate large scale location-referenced CSI data and efficiently deploy DLoc in new environments.

We built DLoc and deployed it in two different indoor environments spanning 2000 sq. ft. area under 10 different scenarios. We summarize our results below:

- DLoc achieves a median accuracy of 65 cm (90-th percentile 160 cm) as opposed to 110 cm median accuracy (90-th percentile 320 cm) achieved by the state-of-the-art[34].
- When tested on an unknown environment, DLoc's performance continues to stay above the state-of-the-art (91 cm median error as compared to 173 cm).

- MapFind maps 2000 sq. ft. area and collects a total of 150,000 data points across 10 different scenarios that take 20 mins per scenario .
- MapFind's data selection algorithm on an average reduces the required path to be travelled by 2.6 × compared to naive random walk.

Our contributions are summarized below:

- DLoc is a novel deep learning framework for WiFi localization that frames localization as an image translation problem. Our neural network formulation incorporates domain knowledge in WiFi localization to deliver state-of-the-art localization performance.
- DLoc is the first algorithm that can correct for time-of-flight offsets without requiring additional instrumentation on client devices.
- MapFind is the first autonomous robot, which provides wireless channel state information with the map of the physical space using SLAM techniques. The mapping not only provides ground truth label for wireless channel state information but also generates a detailed map for map-based navigation.
- We collect a large dataset consisting of 150k points. We believe this dataset can spur innovation in deep-learning-based indoor positioning research, similar to what ImageNet[50] did for computer vision research. [2]

2 DEEP LEARNING BASED LOCALIZATION

As shown in Fig. 1, the system operates in two stages: mapping and localization. During the mapping phase, the MapFind bot, equipped with a WiFi device to collect wireless channel information, performs an autonomous walk through the space to map the environment. Simultaneously, the WiFi device on MapFind collects the CSI for WiFi packets heard from all the access points in the environment. At the end of its walk, the platform generates a map of the environment, and a log of the CSI-data collected at different locations. The CSI-data is labeled with the ground truth locations reported by the platform.

DLoc uses these CSI-log generated by MapFind to train a deep learning model. This model, once trained, can be used by users to locate themselves using their WiFi enabled devices (like smartphones). The users can also access the maps of the building by making calls to a centralized server. In this section, we describe the details of DLoc's algorithm. MapFind's design is described in Section 3. The implementation details and detailed evaluation of MapFind and DLoc are presented in Section 4 and Section 7 respectively. The dataset is described in 5. Few micro-benchmarks for DLoc are discussed in Section 6. Finally, we conclude with a discussion of related work in Section 8.

2.1 Motivation

In free space devoid of reflectors and blockages, wireless channels measured at an access point on a given frequency depend solely on the location of the client device. Let us say that the client is located at location X, then the signal, s, measured at the access point can be written as $s = \alpha(X)$ where α is a function In this case, the objective of any localization system is to simply model a signal-mapping function that maps signal measurements back to user location.

However, this problem is much more complex when there are multiple reflectors and obstacles in place. Let's say we denote the shape, size, and location of objects in our environment as a set of hyperparameters, Θ. Then, the signal s' can be written as $s' = \alpha'(X; \Theta)$. This mapping from reflectors in the environment to signal measurements is computationally very complex as it requires accounting for effects like reflection, refraction, diffraction, etc. In fact, commercial software for modeling such interactions by simulation take several hours to simulate small, simple environments [48].

For localization, we don't even have access to Θ, thus identifying the signal-mapping function corresponding to α' is more challenging than the forward problem of modeling α'. Our insight is that we can leverage neural networks to model the signal-mapping function as a black box. We are motivated by recent advances in deep learning that opt for black-box neural network representations over hand-crafted models and obtain superior performance. This approach allows us to create an implicit representation of the environment, and distills the impact of the environment on the location into the network parameters by observing ground truth data.

2.2 Incorporating Wireless Localization Knowledge for Input Representation

Recall, WiFi uses Orthogonal Frequency Division Multiplexing (OFDM) to divide its bandwidth (e.g., 20/40/80 MHz) into multiple sub-frequencies (e.g., 64 sub-frequencies for 20 MHz). So, the channel state information (CSI) obtained from each access point is a complex-valued matrix, H, of size $N_{ant} \times N_{sub}$. Here, N_{ant} is the number of antennas on the access point and N_{sub} is the number of sub-frequencies. How do we represent this matrix as an input to a neural network?

A naive representation would feed this complex-valued matrix to a neural network as two matrices: one matrix comprising the real values of H and another matrix comprising the imaginary values of H. However, this representation has three drawbacks. First, this representation doesn't leverage all the past work that has been done in WiFi localization, thereby making the task of localization unnecessarily complex for the neural network. Second, this representation is not compatible with most state-of-the-art deep learning algorithms that have been designed for images, speech, and text that do not use complex valued inputs. Finally, it does not embed the location of the access points, information that is necessary for localization.

As discussed before, in DLoc, we use an image-based input representation. We use a two-step approach to transform CSI data into images. Since recent deep learning literature focuses mainly on images [17, 26, 56, 63], this allows us to utilize existing techniques with localization-specific variations. In the first step, we convert the CSI-matrix, H, to a 2D heatmap, $H_{AoA-ToF}$, where AoA is the angle of arrival and ToF is the time-of-flight. The heatmap represents the likelihood of device location at a given angle and at a given distance. Conversion of CSI-matrix, H, to the 2D heatmap, $H_{AoA-ToF}$, can be achieved using two different methods that have been discussed in past work [6, 34]. We use the 2D-FFT transform employed in [6] (example heatmap in Fig. 3).

[2] Our dataset will be made publicly available upon release of the paper.

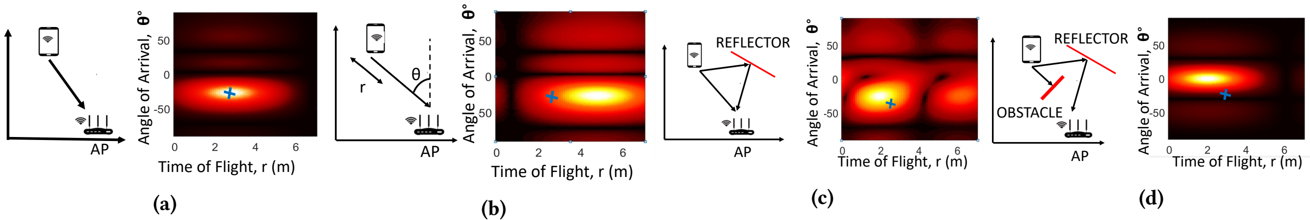


Figure 4: Challenges: DLoc needs to counter three error-inducing effects. First, random time-of-flight (ToF) offsets shift the ideal image (a) along the ToF axis (shown in (b)). The presence of reflectors add spurious peaks, as shown in (c). Finally, in (d), the absence of direct path makes the user device appear at a wrong location, in both angle and distance axes. ('x' denotes the actual location.)

While the $H_{AoA-ToF}$ is an interpretable image, it does not yet encode the location of the access points. To encode the location of the access points in these images, we perform a coordinate transform on these images. We convert these images to a global 2-D Cartesian plane. This transform converts $H_{AoA-ToF}$ to H_{XY}, a matrix that represents the location probability spread out over the X-Y plane. To perform this transform, we define an XY coordinate system where the APs location is (x_{ap}, y_{ap}) and the AoA-ToF heatmap is $H_{AoA-ToF}$. Given this information, we can estimate the XY heatmap, H_{XY} by a Polar to Cartesian transformation. Representing the data as H_{XY} gives us the ability to combine data from all the access points in the environment.

2.3 Deep Neural Network Design

At this point, one might wonder if we can just sum the images obtained for the different access points, and pick the maximum probability point as the target location. Unfortunately, it is not so simple as several challenges prevent this approach from being functional. First, the access point and client device are not time-synchronized with each other. As a result, the time-of-flight information has an unpredictable unknown offset making the peak move away/towards the AP as depicted in Fig. 4(b). Furthermore, this offset is different for the different access points. Therefore, the images corresponding to each access point have an arbitrary radial shift. Second, if obstacles block the direct path from the client to the access point, then there would be no signal visible at the target location as depicted in Fig. 4(d). In fact, the existing localization algorithms [34, 53, 57, 67, 68] fail at taking care of such non-line-of-sight (NLOS) cases, where there is no direct path and thus fail at estimating the accurate location of the client. Finally, WiFi typically has a bandwidth of 20 or 40 MHz, which corresponds to a distance resolution of 15 m and 7.5 m respectively. This means that for a 40MHz bandwidth signal if the direct path from the client to the access point is not separated from the reflections in the environment by at least 7.5 m, these paths cannot be separated as depicted in Fig. 4(c). Therefore, picking the correct location is not as straightforward as just picking the maximum intensity point in the summed heatmaps.

The input representation ensures that all of the above challenges can be framed as image translation challenges. As discussed before, our goal is to use deep learning based image translation and adapt it to solve the challenges for indoor positioning. We want to design the neural network such that it can create a consistent implicit representation of the environment, and then use this representation to output the correct location of the client.

Target Representation: The target of the network is also an image of the same dimensions as the input images. Since our network is an image translation network, it allows it to generalize to environments of arbitrary size. We do not need a separate network for every environmental space. Within the target image, instead of just marking the correct location as one and the rest zeros, we choose a target image with the correct location of the user marked as a Gaussian peak. This Gaussian peak representation is beneficial because it prevents gradient under-flows which are caused by the former approach. We denote this target image as a matrix, $T_{location}$.

Architecture: These optimal choices of input and output representations help us to generalize DLoc's implementation to any state-of-the-art image translation networks. We model our network as an encoder-decoder architecture, with two parallel decoders feeding off the encoder. The architecture is shown in Fig. 5. The two decoders focus on two objectives simultaneously: localization and space-time consistency. The encoder, $E : \mathcal{H} \rightarrow \hat{\mathcal{H}}$, takes in the input heatmaps, $\mathcal{H}$, corresponding to all the APs in the environment and generates a concise representation, $\hat{\mathcal{H}}$ that feeds into the location decoder and consistency decoder. $\mathcal{H}$ is a set of N_{AP} heatmaps, $H_{XY,i}$, one for i^{th} access point ($i = 1, 2, \cdot, N_{AP}$). The consistency decoder, $D_{consistency} : \hat{\mathcal{H}} \rightarrow \mathcal{Y}_{consistency}$ ensures that the network sees a consistent view of the environment across different training samples as well as across different access points. Where $\mathcal{Y}_{consistency}$ are output heatmaps, $Y^{i}_{consistency}$ corresponding to all the APs (say N_{AP}). It does so by correcting for radial shifts caused because of lack of time synchronization between access points and client devices. The location decoder, $D_{location} : \hat{\mathcal{H}} \rightarrow Y_{location}$ takes in the encoder representation, $\hat{\mathcal{H}}$ and outputs an estimate for the location, $Y_{location}$.

Our architecture(see Fig.5) is inspired by the Resnet generator implementation of [28], tweaked to fit our requirements. We add an initial layer of Conv2d with a 7 × 7 kernel followed by Tanh instead of the usual ReLU non-linearity to mimic a log-scale combining across the images over the depth of the network. Further, the Consistency Decoder network has 6 Resnet blocks while the Location Decoder has only 3 Resnet blocks as the insight behind offset compensation through consistency is harder to grasp than optimal combining across multiple APs to identify the correct peak. Further implementation details are discussed in Section 4.

Enforcing Consistency by Removing ToF offsets: As we discussed earlier, the target output of the location decoder are images that highlight the correct location using a Gaussian representation.

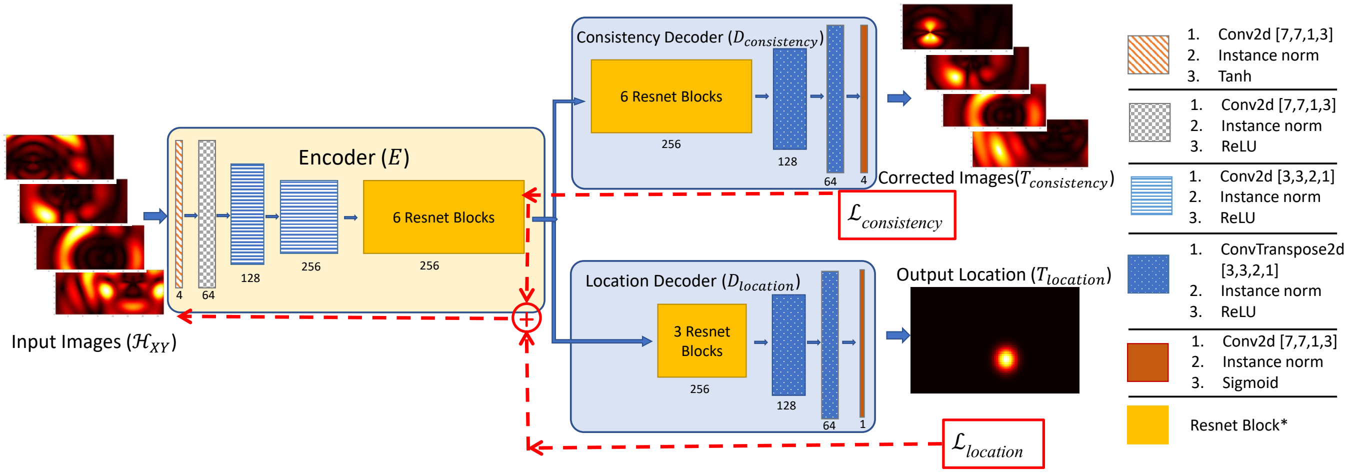

Consistency Decoder ($D_{consistency}$)
6 Resnet Blocks
256
128
64
4
Corrected Images($T_{consistency}$)
$\mathcal{L}_{consistency}$
Encoder (E)
4
64
128
256
6 Resnet Blocks
256
Input Images ($\mathcal{H}_{XY}$)
Location Decoder ($D_{location}$)
3 Resnet Blocks
256
128
64
1
Output Location ($T_{location}$)
$\mathcal{L}_{location}$
1. Conv2d [7,7,1,3] 2. Instance norm 3. Tanh
1. Conv2d [7,7,1,3] 2. Instance norm 3. ReLU
1. Conv2d [3,3,2,1] 2. Instance norm 3. ReLU
1. ConvTranspose2d [3,3,2,1] 2. Instance norm 3. ReLU
1. Conv2d [7,7,1,3] 2. Instance norm 3. Sigmoid
Resnet Block*

Figure 5: DLoc's architecture: DLoc takes the N_{AP} images obtained from the N_{AP} APs as inputs and generates an output image predicting the location with a Gaussian peak. For each Conv2d and ConvTranspose2d, the four values shown are [<kernel-height>,<kernel-width>,<stride>,<padding>]. The red dashed line shows the Loss back-propagation through the network.

A natural question at this point would be, how can we train the consistency network so that it can teach the common encoder to learn to correct for time-of-flight (ToF) offsets that are completely random? Recall, ToF offsets cause random radial shifts in the input heatmaps causing the same objects to appear at different locations in different training samples, as well as different access points. If we do not correct for these random radial shifts, the network will be unable to learn anything about the environment, thereby severely limiting its capability.

Therefore, the objective of the consistency decoder is to take in images from multiple access points that have these completely random ToF offsets and output images without offsets. Our in-sight to resolve this is to consider the images from multiple access points. Thus, while the radial shifts are completely random at each access point, when corrected for these radial shits the images across these multiple access points have a common peak at the correct user location. To achieve this consistency across multiple access points, it needs training data that has no-offset images as targets. We generate these no-offset target images using a heuristic described below.

We have access to images with offsets and the corresponding ground truth locations for the training data. We need to use this information to generate images without offsets. First, we use the image with offsets to identify the direct path. The direct path needs to have the same angle as the correct ground truth location but can have a different ToF (as shown in Fig. 4(a)). Within paths along the same angle, we pick the smallest ToF path as the direct path[3]. Let us say that the direct path has ToF τ'. Further, we calculate the distance between the ground truth location and the access point and divide it by the speed of light to get the expected ToF, say $\hat{\tau}$. Then, the offset can be written as $\tau' - \hat{\tau}$. We shift every point on the image radially (i.e. to reduce/or increase its distance from the origin) by this offset to correct for the offsets and create an offset compensated image. We denote this offset compensated image as a matrix, $T_{consistency}$.

We can, then, use the offset compensated images as targets to train the Consistency decoder. Note that, we have access to the offset compensated images only during training time. We cannot use the heuristic above to create such images if we do not know the true ground truth location (like during real-time operation). Finally, note that the consistency decoder has access to all N_{AP} access points at the same time. It would be impossible to correct for time-of-flight offsets if it had access to just one access point because the offsets for each access point are random. The network can use all N_{AP} access points to check that it applied the correct offsets by looking for consistency across the N_{AP} different APs.

Loss Functions: For both the location and consistency decoders, our inputs and targets are images. Hence, we employ L2 comparative loss for both decoders. Since the Location Network's output is very sparse by definition, we employ L1 regularization on its output image to enforce sparsity. The loss of the consistency decoder, $L_{consistency}$, is defined as:

$$\mathcal{L}_{consistency} = \frac{1}{N_{AP}} \sum_{i=1}^{N_{AP}} L2\Big[D_{consistency}(E(\mathcal{H})) - T_{consistency}\Big]_i \tag{1}$$

where N_{AP} is the number of access points in the environment and $T_{consistency}$ is the offset compensated image. Recall, $\mathcal{H} = [H_{XY,1}, H_{XY,2}, \cdot, H_{XY,N-AP}]$ denotes the input to the network and $D_{consistency}(E(\mathcal{H}))$ is just the output of the consistency decoder once it has been applied to the encoded version of $\mathcal{H}$. Similarly, we define the loss of the localization decoder as:

$$\begin{aligned} \mathcal{L}_{location} = L2\Big[D_{location}(E(\mathcal{H})) - T_{location}\Big] \\ +\lambda \times L1\Big[D_{location}(E(\mathcal{H}))\Big] \end{aligned} \tag{2}$$

where $T_{location}$ are the target outputs (Gaussian-representation of ground truth locations). Finally, the overall loss function is a sum of both $L_{location}$ and $L_{consistency}$ described above.

[3] If there is no path along the correct AoA (we know the correct AoA from the MapFind's location reports), we assume that the direct path is blocked and hence, do no operation and use the images as is.

Training Phase: During the training phase, we utilize labeled CSI data and the ToF offset compensated images to train DLoc end-to-end, where the losses flow and update the network weights as shown by the red dotted line in Fig. 5. With the loss functions defined as above, the network learns to remove the ToF offsets utilizing the Consistency decoder. Since the offset loss and location loss add up to update the Common Encoder, the location decoder gets access to the information regarding ToF offset compensation thus enabling it to learn and predict accurate user locations.

Test Phase: Once the model is trained, we no more need the consistency decoder and only the rest of the network is stored and would continuously run on a central server to which the APs report their CSI estimates from the client associated to them. Then this server upon request can communicate to the client, its location. The server can also send the map generated by MapFind corresponding to the APs location.

Discussion: We highlight a few interesting observations:

- WiFi operates on different bandwidths (20 MHz, 40 MHz, 80 MHz), and hence the input heatmaps can have a different resolution corresponding to the bandwidth. To train our model for all possible bandwidths, we collect training data on the highest available bandwidth (80 MHz in our case). For a subset of this data, we drop down the bandwidth to 20 MHz or 40 MHz chunks at the input of the network, but we stick to the high bandwidth (high resolution) images for the output of the Consistency decoder. This helps the Consistency decoder to not just learn about ToF offsets, but also learn some form of super-resolution to increase resolution.
- Our image translation approach allows the input and output images to be of any size without impacting the resolution. It allows the network to easily update to different environments that may have different size, shapes, and access point deployments.

3 MAPFIND: AUTONOMOUS MAPPING & DATA COLLECTION PLATFORM

Localization of a device is one part of the indoor navigation challenge. The other part is getting access to indoor maps. Descriptive indoor maps, rich in feature information and landmarks, are scarcely available. Furthermore, manually generating these maps becomes increasingly expensive in frequently re-configured environments like grocery stores, shopping malls, etc. Additionally, when designing neural networks, a key challenge is the cost of collecting data. Naturally, if we were to manually move around with a phone in our hand to thousands of locations and measure the ground truth locations, it would take us weeks to generate enough data to train a model for DLoc. We solve both these problems by designing MapFind, a mapping and automated data collection platform. In building MapFind, we strive to meet the following goals:

- **Autonomy:** It should be an autonomous platform for collecting location-associated wireless channel data (CSI data). The data-association between CSI and the map will allow DLoc to provide map-referenced locations.
- **Accuracy:** The collected CSI should be reliable and labeled with accurate location of the WiFi device.
- **Efficiency:** It should collect a diverse set of data points for DLoc in short time (within an hour or two).
- **Ease of Replication:** It should be simple to use and open-source, allowing it to be an ubiquitous platform for testing future WiFi localization algorithms.

3.1 System Design

MapFind builds on extensive research in the SLAM (Simultaneous Localization And Mapping) community that uses autonomous robots equipped with LIDARs, RGBD cameras, gyroscopes and odometers to navigate an environment. We use the publicly available RTAB-Map SLAM framework[36, 37] and Cartographer[21] to create an accurate 2D occupancy grid map as shown in Fig 6(a). Furthermore, given a descriptive map of the environment, these frameworks also provide the locations of the bot.

To achieve autonomous navigation, MapFind works in two stages. Firstly with the user's aid, it navigates the environment as shown in Fig 6 (c). In this stage, SLAM works by capturing data from these sensors and combining this data into an accurate map of the environment. Next, during the autonomous data collection phase, MapFind uses this map to match features it has previously seen before to accurately localize itself in real time.

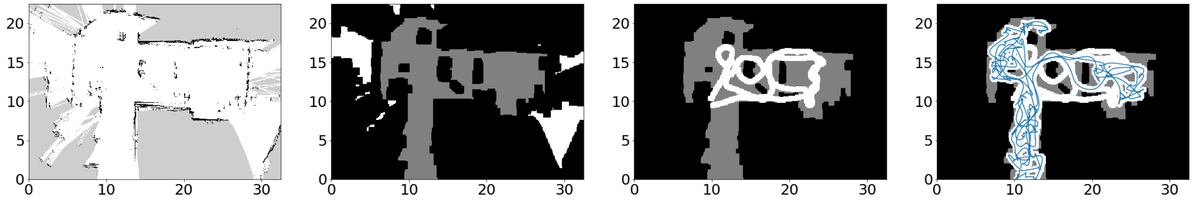
In addition to obtaining location information from these SLAM frameworks, we equip the robot with a WiFi device to collect CSI and tag each CSI measurement with the location measurement. We manually align the local coordinate systems of the access points deployed in the environment with MapFind's global coordinates. Further, to synchronize the timestamps across the bot and the access points, we collect one instance of data for each of these access points at the start of the data collection. This 'sync-packet' provides the start time across all the access points and our bot. This consistent time stamp allows us to associate each CSI measurement with a location provided by MapFind. Thus, MapFind provides the map of the space it explored and the location-referenced CSI data of the environment.

3.2 Path Planning Algorithm

Recall, another objective of MapFind is to generate large-scale training data for DLoc. The data collected during mapping alone is insufficient to train the model as it lacks enough diverse examples to allow DLoc to perform accurate localization. As mentioned above, we want to create an efficient data collection mechanism that allows us to collect diverse training data for new environments.

We achieve this by developing a novel path-planning algorithm which balances between optimizing the path length, area covered and the computation required. We begin by first selecting a random set of points $\mathcal{P} = \{(x_i, y_i) | i \in [1, 2, 3, \cdots, N]\}$. Next, we reject the points which lie on or near any obstacles identified in the occupancy grid. Hence, we obtain a filtered set of points $\mathcal{P}^F = \{(x_i, y_i) | i \in [1, 2, 3, \cdots, N']\}$.

A globally optimal path would search through all the possible paths between the points in $\mathcal{P}^F$. But this would not scale well with the number of size of the space and this problem is NP-hard [16]. To approximate this optimal path, we only search over the m closest point to our current point and search all the paths which pass through d nodes. Hence, we set up an m-ary tree up to a depth d and choose the path which best maximizes for the area



(a) (b) (c) (d)

Figure 6: MapFind Design: (a) MapFind uses an autonomous robot to travel around and creates an indoor map. We segment this map to identify the spaces reachable by the bot (shown in grey in (b)). The path taken by the robot to create the map is shown in (c). This path doesn't provide sufficient coverage to create a diverse training set for DLoc. Therefore, we use a new path planning algorithm to optimize coverage (covered regions shown in white in (d)). Axes in meters.

covered and minimizes the path length. We start from the origin $\mathbf{p}_0 = (x_0, y_0)$ and find the m closest points in $\mathcal{P}^F$ and label it $\mathcal{P}_1$. Using a Probabilistic Road Map (PRM) [30], we trace a trajectory to each of these points. For each of these paths, we calculate the path length L^1_{1i} and coverage C^1_{1i}. Here, coverage of a path is the neighboring space within a radius R_{CSI} traversed by the bot. We assume that the channel within R_{CSI} of the WiFi device on the bot is approximately constant. We would like to maximize the coverage and simultaneously minimize the path length, and hence define the ratio $R^1_{1i} = \frac{C^1_{1i}}{L^1_{1i}}, \ \forall i \in [1, 2, \cdots, m]$.

This defines our computation for the depth 1. We continue this in a recursive manner, for each of the m points, up to a depth d. Hence, at any depth $1 < k \le d$, we have m^2 ratios for paths between the points in $\mathcal{P}_{k-1}$ and $\mathcal{P}_k$:

$$R^k_{ij} = \frac{C^k_{ij}}{L^k_{ij}}, \quad \forall i, j \in [1, 2, \cdots, m].$$

Now, at each depth k and for the path between $\mathbf{p}_i \in \mathcal{P}_{k-1}$ and $\mathbf{p}_j \in \mathcal{P}_k$, we concatenate the m paths given by the lower rung of the recursion with the current path. Next, among these m paths, we take a greedily return the path which maximizes R^k_{ij} to the upper rung of the recursion. We also further filter the points in $\mathcal{P}^F$ which lie within R_{CSI} of this path to reduce the overlap of paths down the line. Hence, $\mathbf{p}_i \in \mathcal{P}_{k-1}$ receives m paths from depth k's m points, and the recursion continues. Note that at depth d, we return $R^d_{ij} = 0, \ \forall 1 \le j \le m$. At the end of the recursion we will have an approximate optimal path from $\mathbf{p}_0$ to some $\mathbf{p}_f \in \mathcal{P}_d$. For our next iteration, we will repeat the above procedure starting at $\mathbf{p}_f$ and continue this iteration until we obtain the required coverage or traverse all the points in $\mathcal{P}^F$

An alternative to using the random path generated above would be to use a more structured path generated by methods described in [9]. These would be more globally optimal and would provide better coverage. But, we employ the random paths for the following two reasons. First, by randomly traversing the space, we bring more temporal variance to the CSI data collected. Concretely, we may revisit similar areas in the environments at random times, hence modelling the CSI data over time. Second, we would like to extend our current work to user-device tracking as well. Following a random path will better mimic a human trajectory and hence create a more realistic dataset.

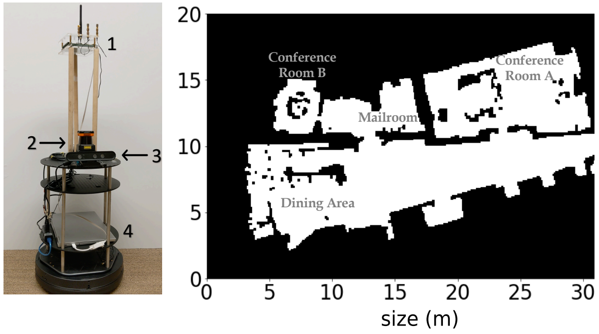


Figure 7: MapFind (left) is an autonomous robotics platform that creates a map (right) for indoor navigation and collects ground truth labeled CSI data for neural network training.

4 IMPLEMENTATION

We describe details of our implementation below.

MapFind: We implement MapFind by mounting an off-the-shelf wireless transmitter provided by Quantenna [47] onto the Turtlebot2 [55] platform, a low cost, open-source robot development kit. As shown in Fig. 7, we mount the Hokuyo UTM-30LX LIDAR (2) at an appropriate height to capture most of the obstacles in our environment. We place the Astra Orbecc RGBD Camera [44] (3) close to the LIDAR [23] and match their point clouds for accurate registration. The Quantenna Wi-Fi card (1) is placed on an acrylic platform that rests in a layer above the LIDAR. This acrylic base is supported by dowels mounted on the highest layer of the turtlebot in such a fashion that they do not obstruct the LIDAR's field of view. Further note that the Quantenna is placed at a height where an average user might hold their phone to collect representative data. The Turtlebot 2 is controlled via Robot Operating System (ROS-Kinetic) using a laptop equipped with 8th Gen Intel Core i5-8250U mobile processor and 8GB of RAM (4), giving us access to a large number of packages for SLAM and navigation. We use Probabilistic Road Map [30] to chart an obstacle-free path with 75 nodes and a maximum edge length of 3m. A key advantage of MapFind's design is that both the robot and the WiFi card can be replaced by suitable alternatives making the design very flexible.

DLoc: Fig. 5 summarizes the design of DLoc's Deep Neural Network. We implement this architecture in PyTorch[1] while the network architecture is inspired from the recent generator model implemented in [28]. Especially the Resnet blocks implemented in

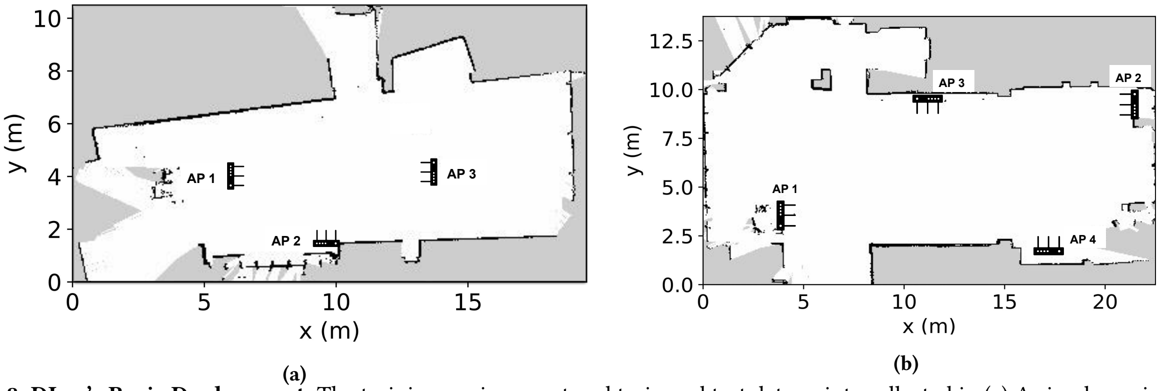


Figure 8: DLoc's Basic Deployment: The training environment and train and test data points collected in (a) A simple environment that spans 500 sq. ft. space with 4 access point.(b) A complex environment that spans 1500 sq. ft. space with 4 access points.

the design are borrowed from the author's implementation in [28]. For all the layers in the network we do not use any dilation. For the ConvTranspose2d layers, the parameter *output_padding* = 1. We further employ InstanceNorm [56] instead of the standard BatchNorm layer for normalization as the recent research [56, 61, 63, 76] shows better performance using InstanceNorm over batch normalization for image-to-image translation networks.

For training our model, we use a learning rate of $\alpha = 1e - 5$ with no rate annealing, and maintain a batch size of 32 across the whole set of experiments. We follow an Adam optimizer [33] schedule for our gradient descent optimizer with L2 weight regularization with $weight - decay = 1e - 5$. The regularization parameter $\lambda = 5e - 4$.

5 DATASET

In any deep learning model, the quality of the datasets plays a key role. So, in this section, we go into the details of the data we have collected to evaluate our model. We deploy MapFind in two different spaces which together span 2000 sq. ft. area to acquire labeled Wi-Fi channel state information. The two spaces (Fig. 8) are real-world deployments with rich multipath (plasma screens, concrete pillar, metal structures, etc.) and non-line-of-sight scenarios. We deployed off-the-shelf Quantenna APs in each space (4 in the larger area and 3 in the smaller one), which estimate the timestamped CSI data. The Quantenna APs are scheduled to estimate channel once every 50ms. So, we navigate the MapFind robot at a constant speed of 15 cm/sec to avoid any doppler effects and also to be able to ping all the APs for one location without causing drift in locations. These channel estimates are then sent to a central server, and along with MapFind's ground truth estimates, this becomes the data on which we train and test DLoc. We have collected data under a total of 8 different scenarios described below:

- We first deploy and test our algorithm in a *simple space* of 500 sq. ft. with direct path available most of the time. We collect data in this setup under three different scenarios at different times, two datasets collected on different days with the basic setup shown in Figure 8(a) and 1 extra scenario where we add reflector to the given environment along the edge of the space.
- We also deploy and test our algorithm in a *complex space* of 1500 sq. ft. where AP 4 placed in the environment is hidden behind a wall, thus collecting significant NLOS data. Furthermore, the wall of plasma television screens behind AP3 creates a multipath rich environment. We collect data in this setup under five different scenarios at different times, two datasets collected at different times of a day with the basic setup shown in Figure 8(b) and the 4 different scenarios with different settings of furniture as shown in Figure 12.

In each of these scenarios we let the bot explore the space for about 20 minutes to simultaneously map and collect CSI data across all the access points in the given environment. With this setup we collect 12,500 data points for a given scenario on an average. Thus we collect about 100,000 datapoints overall in multiple scenarios that are diverse in space and time. We split this dataset into multiple parts to appropriately train and test the network for each of the experiments mentioned in section 7,

6 MICROBENCHMARKS

Before we delve into the evaluation of DLoc and MapFind, let's start with few microbenchmarks. In particular, we show the output of the consistency decoder and the final location estimate for a given set of input heatmaps for 40MHz bandwidth signal from all the 4 APs. These results are when the network is trained and tested on data from the setup shown in Figure 8(a). Where DLoc's network has been trained as described in Section 2.3. In Fig. 9 we show sample inputs from the 4 APs to DLoc network in the top, the corresponding outputs from the consistency decoder below them and their corresponding targets at the bottom. The images shown are generated during test time at a location which the network has not seen during its training phase. The green cross is the ground truth label for this specific record of data, while the blue cross is the location predicted by DLoc's location decoder output image. Further in the images shown the brighter the pixel value the higher the likelihood. We highlight four aspects of these images below.

Offset Removal: We first analyze the performance of the consistency network in removing the ToF offset. As can be seen from the first row of images, the input heatmaps to DLoc's network have offset, in that the peaks of the maxima do not coincide with the ground truth label, though the direction of the peak looks accurate. It can also be observed with the right amount of shift of the peak along the correct direction will make all of the maxima across the 4

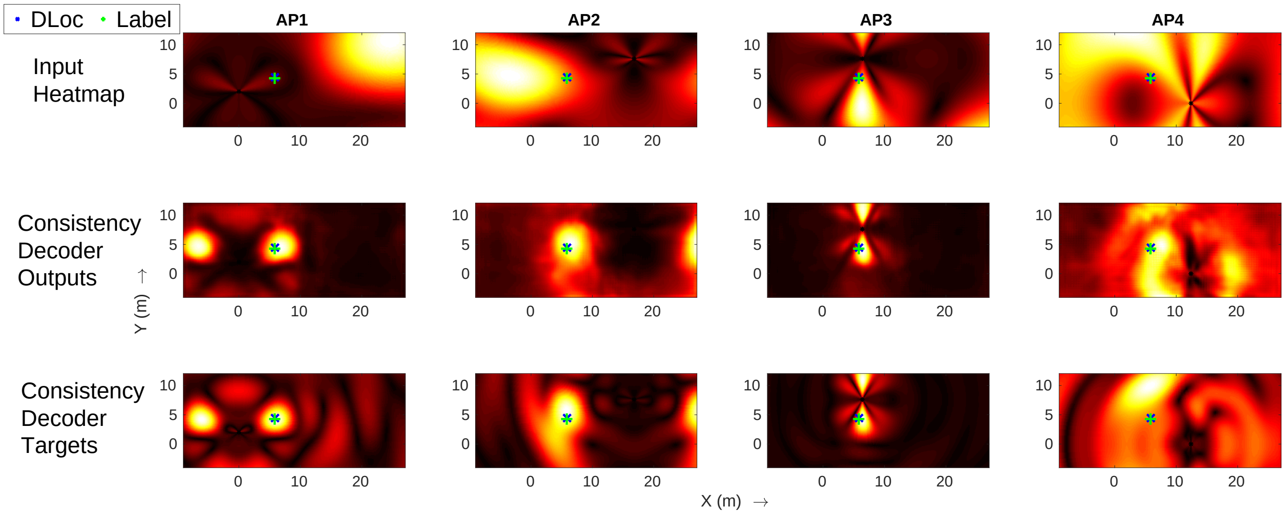


Figure 9: Micro-benchmark: Outputs of DLoc's consistency decoder show that DLoc achieves multipath resolution, corrects for the ToF offset and performs super-resolution for lower than 80MHz bandwidth signals. The green '+' shows the ground truth label reported by MapFind and the blue 'x' shows the Location predicted by DLoc.

AP heatmaps to coincide at the correct user location. That is exactly what we want the consistency network to learn. As we can see in the bottom row of images corresponding to the input heatmaps, the peaks of the heatmap images now coincide with the correct location of the ground truth label. Thus, we can see that the consistency decoder has corrected for the ToF offset for each heatmap image by enforcing consistency across all the images through the consistency loss.

Multipath resolution: Apart from ToF offset removal, DLoc should also be able to resolve multiple paths and identify the direct path, for which we can observe the input and consistency decoder's output of AP4. As shown in the figure, AP4 suffers from severe multipath. Because of this, we can see two clear maxima along with two different angles in the input heatmap. The location estimate chosen by the location decoder is towards the correct angle as can be seen by the blue cross overlapping with the green cross corresponding to the ground truth label. This shows that the network can learn to resolve the correct path from multiple paths.

Bandwidth Super-resolution: As discussed earlier, we have trained the consistency decoder with target heatmaps corresponding to 80MHz bandwidth signal, while the input heatmaps to the encoder belong to 40MHz bandwidth signal for the same location. This training approach should enable the network to learn super-resolution to better resolve multiple paths in the environment. Now let us focus on the input heatmap of AP2 and the corresponding output of the consistency network. We can observe that the input heatmap on the top has a more wide-spread likelihood maxima, while the bottom image apart from ToF offset removal also shows a tighter likelihood maxima. This proves that the training procedure employed for DLoc helps us in achieving super-resolution and generalization over various signal bandwidths of the user as further discussed in Section 7.2.

Evidence for generalization across space: As mentioned earlier, this specific location that we show these images for has not been encountered by the network during the test phase, unlike in fingerprinting where every location is looked at at least once. We still observe from these images that the network generalizes the offset removal, multipath-resolution, and bandwidth super-resolution across the space in the locations the network has not been trained. We can further observe from the outputs of AP4, where though the target image does not have the peak at the correct location due to the lack of direct path, the network corrects for the case thus giving an appropriate location output by looking at the consistency across all the 4 APs. Thus, these sample images become an initial evidence to the generalization of DLoc which is further shown in Table 1 and detailed in section 7.2.

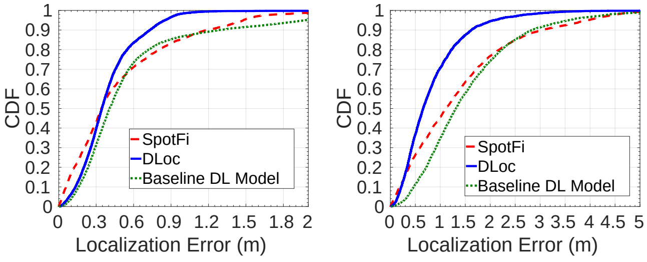
7 EVALUATION

The labeled data described in 5 is used for training DLoc. We compare DLoc to two baselines: SpotFi [34] and a baseline deep learning model [5]. For both these baselines, we do a best-effort reimplementation of the respective systems. To evaluate DLoc on the simple environment, the training and test data are taken from the same dataset where the 70% is used for training and the 30% for testing. Similarly for the complex environment we train on 80% data collected on two different time instances and test on the rest 20%.

7.1 DLoc's Performance

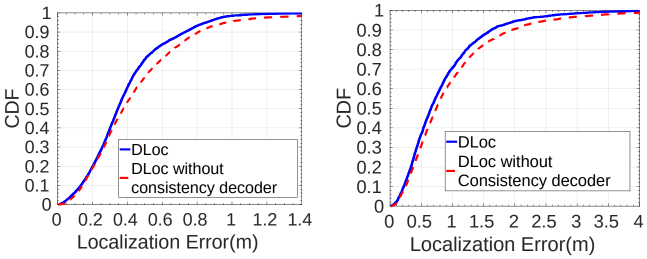
The outputs of DLoc are 2D images with location intensities. We take the index of the maximum value in these images and scale it down with the grid size to get the location estimated by DLoc. We report the distance between DLoc's estimated position and the actual ground truth label provided by MapFind. We show the CDF of these errors for DLoc's location estimates in Fig. 10a,b and compare these results with the state-of-the-art SpotFi baseline and a Baseline DL model.

For Fig. 10a, the experiments are conducted under a smaller simpler space of 500 sq. ft. with only 3 access points. From the results in this smaller space we can see that while SpotFi and DLoc have almost the same median error of 36cm, DLoc outperforms



(a) (b)

Figure 10: DLoc Results: DLoc outperforms state-of-the-art models (SpotFi and Baseline DL model) in localization (a) in a simple 500 sq. ft. space and (b) in a complex 1500 sq. ft. space



(a) (b)

Figure 11: Ablation Study: Dloc's performance with and without consistency decoder in both (c) Simple Environment (500 sq. ft.) and (d) Complex Environment (1500 sq. ft.)

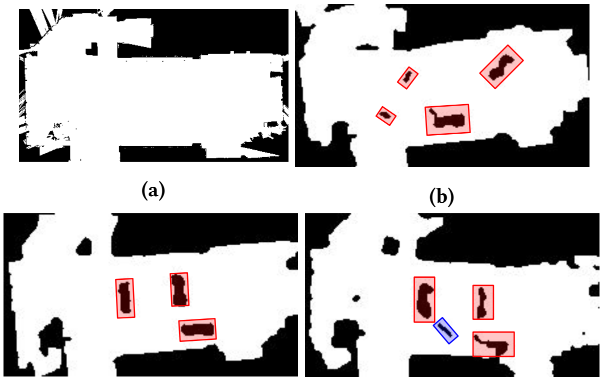
SpotFi by 2× at 90th (and 99th) percentile. While DLoc achieves 70cm (and 1m) localization error at 90th and 99th percentile, SpotFi goes up to 140cm (and 2m), Further we can see that Baseline DL model performs much worse than even SpotFi at both median and 90th percentile, with the decrease in number of total receiver antennas (3 APs with 4 antennas each).

From the Fig. 10b, we can clearly see that in a complex space of 1500 sq. ft., where the median localization error for DLoc is 64cm, the median localization error for SpotFi is 110cm and Baseline DL model is 126cm. Further, we make a case that DLoc can characterize a given environment and thus achieve lower errors at 90th (and 99th) percentile. This can now further be validated with the results in Fig. 10b, where the 90th (and 99th) percentile localization error for DLoc is 1.6m (and 3.2m) the same for SpotFi goes up to 3m (and 4.8m) and for the Baseline DL model goes up to 2.8m (and 4.5m). Therefore, DLoc outperforms the SpotFi algorithm and Baseline DL model at both the median and 90th percentile.

Further, to understand the importance of the consistency decoder, we train and test the encoder with just the location decoder (without consistency decoder) and plot the results in Fig. 10c,d. We can see that in the absence of the consistency decoder, the performance of the network goes down (error goes up from 36 cm to 48 cm and 65 cm to 80 cm) in Fig. 10c and Fig. 10d respectively. This is because it becomes harder for the network to model the environment due to inconsistent inputs being supplied to it.

7.2 DLoc's Generalization

To understand what DLoc is learning, it is important to understand the generalizability of the network and to do that, we look at three specific scenarios of generalizability.



(a) (b) (c) (d)

Figure 12: Multiple Furniture Setups: We test the generalizability of DLoc across multiple setting of *Complex Environment* shown in (a), which we refer as Furniture Setup-1 (no furniture) (b) Furniture Setup-2, (c) Furniture Setup-3, (d) Furniture Setup-4. Furniture highlighted in 'red' and reflector highlighted in 'blue'

Trained on Setup	Tested on Setup	Median Error (cm)		90^{th}%ile Error (cm)	
		DLoc	SpotFi	DLoc	SpotFi
1,3,4	2	71	198	171	420
1,2,4	3	82	154	252	380
1,2,3	4	105	161	277	455

Table 1: Complex Cross Environment Testing: Median and 90th percentile errors when trained and tested on across different setups of the complex environment as shown in Fig. 12.

Across Furniture and Reflector Motion: In a daily office setup scenario, the furniture would move around once in a while and it is important that our algorithm does not break with slightest of changes in these objects. We set up 5 different scenarios for our case study. In Setup-1, there is no furniture. In scenario 2-4 we added some furniture and changed the furniture's position around for each scenario as shown in Fig. 12. Additionally in Setup 4, we place the furniture as is in Setup-3 and add an additional 1.5 m × 2 m reflector to the environment. These variations in the environment add more NLOS scenarios than the original Setup-1 (especially in Setup-4 with an additional reflector in the field). For this setup, we do cross testing where we train on different setup's data and test on a completely different setup's data. The median and 90th percentile localization errors are reported in Table 1. From this table, we can see that DLoc is robust to furniture motion. It deteriorates slightly when an additional reflector is added, when there are increased NLOS data points, but is still more robust than SpotFi.

Across Bandwidths: As mentioned earlier, the user device does not always have access to higher bandwidths, and thus we train our network on different bandwidth data all the while retaining 80MHz target images for the consistency decoder. Doing this makes the network learn "super-resolution" of the given image as discussed in Section 6. Here we show the performance of DLoc when different bandwidth signal input heatmaps are given as inputs to the network. As shown in Fig. 10e, we can see that the median localization error decreases marginally for DLoc from 65cm for 80MHz to 72cm for both 40MHz and 20MHz signal data. In contrast, SpotFi could

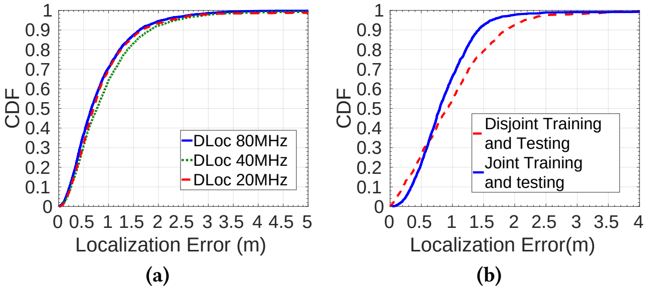


Figure 13: Generalization: DLoc's performance generalizes to inputs from (e) different bandwidths and (f) different space overlaps.

achieve a median error of 110 cm even with 80 MHz of bandwidth. This shows that DLoc can effectively operate with input data of varying bandwidth.

Across Space: Though MapFind's path-planning algorithm optimizes the coverage of the given environment, we cannot always cover each and every location on the map. This makes it important to look at DLoc's generalizability across space. To quantify this, we split the training and testing datasets into two disjoint spatial regions. The path segment covered in the region belonging to $X \in (10\text{m}, 14\text{m})$, $Y \in (6\text{m}, 8\text{m})$ is used for testing, while the rest of the path's data is used for training the network. We compare this scenario with a joint training scenario when the training and test points are sampled from the entire space and show the comparative results in Fig. 10f. We can see that the disjoint training and testing very closely follows the trend of the joint training and testing, showing the generalizability across spaces of DLoc in a given environment.

7.3 MapFind's Performance

Ground Truth Accuracy: We test the accuracy of the ground truth reported by MapFind by using a HTC Vive VR system. The HTC Vive performs outside-in tracking and hence provides accuracy upto mm-level in dyanamic scenarios [8]. We test two SLAM algrithms – RTAB-Map and Cartographer [21] – in a 4 m × 4 m environment, the maximum allowed grid size by HTC Vive. We find the median error to be 5.7 cm and 7.3 cm for RTAB-Map and Cartographer respectively and report the errors in Fig. 14. Since RTAB-Map performs slightly better, we use RTAB-Map for MapFind's design.

Impact of Label Errors on DLoc: Since our labels (obtained using MapFind) have a median error of 5.7 cm, we wish to study the impact that this has on DLoc, since DLoc is trained using these labels. To achieve this objective, we design a simple experiment. We use the HTC Vive to collect a dataset of 2500 points in a 4 m × 4 m space in the larger environment. We use both MapFind and the VR system to train DLoc and tabulate the test errors in Table 3. As shown, the impact of the errors in labelled data is minimal. This is primarily because of the high accuracy achieved by MapFind. Finally, note that, while the VR system is more accurate, it is limited in the range it can cover. Therefore, we limit our evaluation in this subsection to a 4 m × 4 m space.

Path Planning Performance: To characterize the performance of our path planning algorithm, we compare our performance with a naive random-walk approach and a Greedy graph traversal approach. We fine-tune the greedy algorithm for its best performance. For the multi-agent search, we set the number of neighbors, $m = 5$, the radius of coverage, $R_{CSI} = 70cm$, and the depth of search, $d = 5$, to lower computational overhead. Our path length and coverage to path length ratio are characterized in Table 2. As shown, our path planning algorithm can reduce the path traveled as well as optimize the coverage as compared to a random walk and greedy algorithms. This allows MapFind to efficiently collect data for new environments.

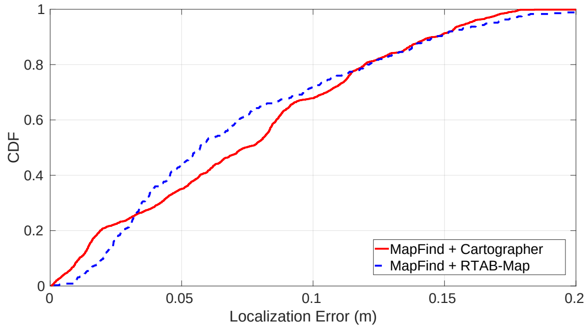


Figure 14: MapFind's Accuracy: We use MapFind to generate labelled data for training DLoc This figure shows the accuracy of the labels generated by MapFind using two different SLAM algorithms.

Algorithm	Path Length (m)	$\frac{\text{Coverage}}{\text{Path Length}}$(km^{-1})
Multi-Agent (ours)	322.5	2.35
Greedy	378.1	2.15
Random Walk	851.5	1.04

Table 2: Performance of MapFind's Path Planning Algorithm

Trained with	Median Error (cm)		90^{th}%ile Error (cm)	
	DLoc	SpotFi	DLoc	SpotFi
VR	89	173	171	316
MapFind	94	172	187	316

Table 3: DLoc trained on MapFind's and VR system's reported location and compared against VR system's locations; SpotFi compared against both VR and MapFind's reported locations

8 RELATED WORK

Our work is related to and draws on three lines of research:

Indoor Mapping: SLAM is well studied for creating maps in indoor and outdoor locations and is used by Google cars, Clearpath robotics, Sanborn, and other organizations [12, 18, 22, 51, 54]. Most of these mapping platforms focus only on building maps and primarily rely on GPS for location. A subset of SLAM platforms [20, 24] use Wi-Fi, but they collect coarse-grained information like signal strength. In contrast, we instrument a WiFi platform which can extract detailed fine-grained information about the wireless channel like channel state information, which is quintessential to achieving accurate indoor localization. Furthermore, we develop a new path planning algorithm that optimizes the time required for collecting training data in a given environment.

More concretely, in the world of path planning, there exists coverage path planning (CPP) [9] and map exploration (ME). CPP algorithms generate structured paths achieving coverage of around 95%, whereas ME algorithms use Probabilistic Road-Maps [30] or Rapidly-searching Random Trees (RRT) [39] to randomly explore the map to find a route between two points. Since we want to produce a pseudo-random path, we perform a multi-agent search up to a depth, d on paths generated by a PRM, essentially borrowing ideas from both CPP and ME.

WiFi Localization: WiFi-based localization is a well-studied topic with extensive work ranging from RSSI based localization [7, 11, 45, 69, 77] to channel state information (CSI) based localization [2, 3, 29, 34, 35, 46, 57, 60, 64–68, 70]. In recent times, it has been established that CSI works better for localization achieving sub-meter median accuracies [34, 57]. However, CSI-based WiFi localization algorithms suffer from problems caused by the environment (such as multipath effect, non-line of sight, etc.). These problems have been extensively studied in literature [3, 29, 34, 35, 57, 64–68]. The typical solution to these problems has been to design heuristics to identify the direct path and to try to subdue the effects caused by the environment. This approach fails when the direct path is completely blocked or when the environmental effects shift the inferred location of the direct path. We take a different approach. We use deep learning to implicitly model the environment and to use this model to predict correct locations. Thus even when the direct path is blocked, our model can use reflected paths for positioning. As we show in our results, this allows us to achieve better median accuracy as well as better 90-percentile and 99-percentile accuracy.

There has been some recent work in localization using deep neural networks [5, 13, 43, 62, 75]. We differ from this work along four axes. First, instead of using complex-valued wireless channels as inputs to neural nets, we frame our problem as an image translation problem. This allows us to leverage the state-of-the-art image translation research (which is known to outperform complex-valued neural networks). Second, we explicitly model the effects caused by off-the-shelf devices like time-of-flight offsets. This allows us to model consistency into the network, thereby allowing it to learn about objects in the environment. Third, CSI data varies randomly even with minute changes to the environment but representing them as XY images removes this randomness and helps us model the environment better for localization. Finally, we augment our algorithm with an automated data collection platform that can efficiently collect data for a new environment, making our system easy to deploy. Using MapFind and DLoc together, we enable large scale real-world data for WiFi-localization. In the future, we hope everyone can use our data to form a standardized test.

Deep learning for Image translation: Our key insight of representing channel state information as two-dimensional images helps us to pose all of our localization algorithms as image-to-image translation problems. This representation allows us to use extensive literature on image-to-image translation that has been very well studied in the computer vision community [28, 38, 52, 72, 76]. These algorithms have utilized generator models paired with appropriate targets and loss functions to solve many image-to-image translation problems like image denoising [10, 71], image super-resolution [15, 28, 32], image colorization [27, 31], and real-to-art image translations [26, 27, 76]. While there has been some work that utilizes rf-based techniques that utilize machine learning to solve through-wall human pose estimation [58, 73, 74], our paper is the first to present general principles for using ideas of image-to-image translation for the localization problem. Specifically, the data distribution of the indoor WiFi localization data is different from all image translation work, and RF-based pose estimation data. Thus, we define a tailored target and loss functions. Moreover, we create mechanisms to model Wi-Fi specific issues like time-of-flight offsets. Finally, we note that our model is adaptive enough to incorporate future advances in image-to-image translation research. In short, we can use ideas of the loss function and the training procedure to apply to GANs [17] and any future developments on image-to-image translation algorithms.

9 LIMITATIONS AND FUTURE WORK

This paper presents: (a) DLoc: a deep learning model for indoor localization, and (b) MapFind: an autonomous mapping platform that also collects training data for DLoc with minimal human effort. Together, these platforms provide a framework for indoor navigation. We conclude with a discussion of the limitations of our current design and ideas for future work:

- Our robot is limited to 2D mapping, and we perform 2D localization. Some applications, like indoor virtual reality, might require 3D localization and mapping. We believe our model allows for a natural extension to 3D, by replacing 2D images with 3D images (similar to how videos are modeled in computer vision research). However, this extension remains a part of future work.
- The speed of the robot is 15 cm/s which limits our data collection efficiency. Faster robots can reduce the data collection time to a few minutes. However, we opted for cheaper robots to ensure that the system can be easily replicated and used by others in the community.
- Our model relies on angle-of-arrival and time-of-flight information. Past work [64, 65] has shown how to extract other information like angle-of-departure, doppler shift, etc. Incorporating this information into heatmaps will present additional information to the network, and is a part of future work.
- In the future, we plan to build on our work to demonstrate large scale, real-time, location-based applications for entire buildings and shopping malls.
- Past work like [57] has shown that localization systems relying on angle of arrival and time-of-flight achieve similar performance with human-held devices and robotic devices like drones. Since DLoc relies on this information at the input, we expect it to generalize to human-held devices. However, we acknowledge that there might be additional effects that might not have been modeled in the deep learning architecture and is a limitation of DLoc and is exciting possible future work.

REFERENCES

[1] [n. d.]. PyTorch. ([n. d.]).
[2] Fadel Adib, Zach Kabelac, Dina Katabi, and Robert C. Miller. 2014. 3D Tracking via Body Radio Reflections *(NSDI)*.
[3] Fadel Adib and Dina Katabi. 2013. *See through walls with WiFi!* Vol. 43. ACM.

[4] Apple. [n. d.]. Apple Maps. ([n. d.]).

[5] Maximilian Arnold, Jakob Hoydis, and Stephan ten Brink. 2019. Novel Massive MIMO Channel Sounding Data applied to Deep Learning-based Indoor Positioning. In *SCC 2019; 12th International ITG Conference on Systems, Communications and Coding*. VDE, 1–6.

[6] Roshan Ayyalasomayajula, Deepak Vasisht, and Dinesh Bharadia. 2018. BLoc: CSI-based accurate localization for BLE tags. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*. ACM, 126–138.

[7] Victor Bahl and Venkat Padmanabhan. 2000. RADAR: An In-Building RF-based User Location and Tracking System *(INFOCOM)*.

[8] Miguel Borges, Andrew Symington, Brian Coltin, Trey Smith, and Rodrigo Ventura. 2018. HTC Vive: Analysis and accuracy improvement. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2610–2615.

[9] Richard Bormann, Florian Jordan, Joshua Hampp, and Martin Hägele. 2018. Indoor Coverage Path Planning: Survey, Implementation, Analysis. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1718–1725.

[10] Antoni Buades, Bartomeu Coll, and J-M Morel. 2005. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 2. IEEE, 60–65.

[11] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. 2010. Indoor Localization Without the Pain *(MobiCom)*.

[12] clearpath. [n. d.]. Clearpath Robotics. ([n. d.]). https://www.clearpathrobotics.com/.

[13] Marcus Comiter and HT Kung. 2018. Localization Convolutional Neural Networks Using Angle of Arrival Images. In *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–7.

[14] OpenStreetMap Community. [n. d.]. Open Street Maps. ([n. d.]).

[15] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2016. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence* 38, 2 (2016), 295–307.

[16] Enric Galceran and Marc Carreras. 2013. A survey on coverage path planning for robotics. *Robotics and Autonomous systems* 61, 12 (2013), 1258–1276.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[18] Google. [n. d.]. Google indoor maps. ([n. d.]). https://www.google.com/maps/about/partners/indoormaps/.

[19] google. [n. d.]. Google Maps. ([n. d.]). https://www.google.com/maps.

[20] Zakieh Sadat Hashemifar, Charuvahan Adhivarahan, and Karthik Dantu. 2017. Improving RGB-D SLAM using wi-fi. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 317–318.

[21] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. 2016. Real-time loop closure in 2D LIDAR SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1271–1278.

[22] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. 2016. Real-Time Loop Closure in 2D LIDAR SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 1271–1278.

[23] Hokuyo. [n. d.]. Scanning Range Finder. ([n. d.]).

[24] Joseph Huang, David Millman, Morgan Quigley, David Stavens, Sebastian Thrun, and Alok Aggarwal. 2011. Efficient, generalized indoor wifi graphslam. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 1038–1043.

[25] Infsoft. [n. d.]. Bluetooth Low Energy Beacons. ([n. d.]).

[26] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2016. Image-to-Image Translation with Conditional Adversarial Networks. *arxiv* (2016).

[27] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. *arXiv preprint* (2017).

[28] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*. Springer, 694–711.

[29] Kiran Joshi, Steven Hong, and Sachin Katti. 2013. PinPoint: Localizing Interfering Radios *(NSDI)*.

[30] Lydia Kavraki, Petr Svestka, and Mark H Overmars. 1994. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*. Vol. 1994. Unknown Publisher.

[31] Sachin Kelkar, Chetanya Rastogi, Sparsh Gupta, and GN Pillai. 2018. SqueezeGAN: Image to Image Translation With Minimum Parameters. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–6.

[32] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1646–1654.

[33] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[34] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. 2015. SpotFi: Decimeter Level Localization Using Wi-Fi *(SIGCOMM)*.

[35] Swarun Kumar, Stephanie Gil, Dina Katabi, and Daniela Rus. 2014. Accurate Indoor Localization with Zero Start-up Cost *(MobiCom)*.

[36] Mathieu Labbe and Francois Michaud. 2013. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics* 29, 3 (2013), 734–745.

[37] Mathieu Labbe and FranÃğois Michaud. 2019. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics* (2019).

[38] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. 2014. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 149.

[39] Steven M LaValle. 1998. Rapidly-exploring random trees: A new tool for path planning. (1998).

[40] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. 2007. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37, 6 (2007), 1067–1080.

[41] locatify. [n. d.]. Indoor Position System using BLE beacons. ([n. d.]).

[42] microsoft. [n. d.]. Bing Maps. ([n. d.]). https://www.bing.com/maps.

[43] Michał Nowicki and Jan Wietrzykowski. 2017. Low-effort place recognition with WiFi fingerprints using deep learning. In *International Conference Automation*. Springer, 575–584.

[44] Orbbec. [n. d.]. Astra RGB-D camera. ([n. d.]).

[45] Nissanka B Priyantha, Anit Chakraborty, and Hari Balakrishnan. 2000. The cricket location-support system. In *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM, 32–43.

[46] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2013. Whole-home Gesture Recognition Using Wireless Signals *(MobiCom)*.

[47] Quantenna. [n. d.]. Quantenna 802.11ac WiFi Card. ([n. d.]).

[48] REMCOM. [n. d.]. Wireless Insite. ([n. d.]).

[49] Chris Rizos, Gethin Roberts, Joel Barnes, and Nunzio Gambale. 2010. Experimental results of Locata: A high accuracy indoor positioning system. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*. IEEE, 1–7.

[50] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.

[51] sanborn. [n. d.]. SPIN Indoor Mapping. ([n. d.]). https://www.sanborn.com/spin-indoor-mapping/.

[52] Yichang Shih, Sylvain Paris, Frédo Durand, and William T Freeman. 2013. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 200.

[53] Elahe Soltanaghaei, Avinash Kalyanaraman, and Kamin Whitehouse. 2018. Multipath Triangulation: Decimeter-level WiFi Localization and Orientation with a Single Unaided Receiver. In *MobiSyS*.

[54] A. J. B. Trevor, J. G. Rogers, and H. I. Christensen. 2014. OmniMapper: A modular multimodal mapping framework. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 1983–1990. https://doi.org/10.1109/ICRA.2014.6907122

[55] Tutlebot. [n. d.]. Turtlebot Robotics. ([n. d.]).

[56] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022* (2016).

[57] Deepak Vasisht, Swarun Kumar, and Dina Katabi. 2016. Decimeter-Level Localization with a Single Wi-Fi Access Point *(NSDI)*.

[58] Fei Wang, Sanping Zhou, Stanislav Panev, Jinsong Han, and Dong Huang. 2019. Person-in-WiFi: Fine-grained Person Perception using WiFi. *arXiv preprint arXiv:1904.00276* (2019).

[59] Ju Wang, Hongbo Jiang, Jie Xiong, Kyle Jamieson, Xiaojiang Chen, Dingyi Fang, and Binbin Xie. 2016. LiFS: low human-effort, device-free localization with fine-grained subcarrier information. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 243–256.

[60] Jue Wang and Dina Katabi. 2013. Dude, Where's My Card?: RFID Positioning That Works with Multipath and Non-line of Sight *(SIGCOMM)*.

[61] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[62] Xuyu Wang, Lingjun Gao, and Shiwen Mao. 2017. BiLoc: Bi-Modal Deep Learning for Indoor Localization With Commodity 5GHz WiFi. *IEEE Access* 5 (2017), 4209–4220.

[63] Yuxin Wu and Kaiming He. 2018. Group Normalization. In *The European Conference on Computer Vision (ECCV)*.

[64] Yaxiong Xie, Jie Xiong, Mo Li, and Kyle Jamieson. 2016. xD-track: leveraging multi-dimensional information for passive wi-fi tracking. In *Proceedings of the 3rd Workshop on Hot Topics in Wireless*. ACM, 39–43.

[65] Yaxiong Xie, Jie Xiong, Mo Li, and Kyle Jamieson. 2018. mD-Track: Leveraging Multi-Dimensionality in Passive Indoor Wi-Fi Tracking. *arXiv preprint arXiv:1812.03103* (2018).

[66] Jie Xiong and Kyle Jamieson. 2012. Towards fine-grained radio-based indoor location. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems &*

Applications. ACM, 13.

[67] Jie Xiong and Kyle Jamieson. 2013. ArrayTrack: A Fine-grained Indoor Location System *(NSDI)*.

[68] Jie Xiong, Karthikeyan Sundaresan, and Kyle Jamieson. 2015. ToneTrack: Leveraging Frequency-Agile Radios for Time-Based Indoor Wireless Localization *(MobiCom)*.

[69] Zuwei Yin, Chenshu Wu, Zheng Yang, and Yunhao Liu. 2017. Peer-to-peer indoor navigation using smartphones. *IEEE Journal on Selected Areas in Communications* 35, 5 (2017), 1141–1153.

[70] Moustafa Youssef and Ashok Agrawala. 2005. The Horus WLAN Location Determination System *(MobiSys)*.

[71] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing* 26, 7 (2017), 3142–3155.

[72] Richard Zhang, Phillip Isola, and Alexei A Efros. 2016. Colorful image colorization. In *European Conference on Computer Vision*. Springer, 649–666.

[73] Mingmin Zhao, Tianhong Li, Mohammad Abu Alsheikh, Yonglong Tian, Hang Zhao, Antonio Torralba, and Dina Katabi. 2018. Through-wall human pose estimation using radio signals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7356–7365.

[74] Mingmin Zhao, Yonglong Tian, Hang Zhao, Mohammad Abu Alsheikh, Tianhong Li, Rumen Hristov, Zachary Kabelac, Dina Katabi, and Antonio Torralba. 2018. RF-based 3D skeletons. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 267–281.

[75] Xiaolong Zheng, Jiliang Wang, Longfei Shangguan, Zimu Zhou, and Yunhao Liu. 2017. Design and implementation of a CSI-based ubiquitous smoking detection system. *IEEE/ACM Transactions on Networking* 25, 6 (2017), 3781–3793.

[76] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

[77] Xiuyan Zhu and Yuan Feng. 2013. RSSI-based algorithm for indoor localization. *Communications and Network* 5, 02 (2013), 37.