Doctoral Thesis Proposal

Transmitter Localization in Classical and Quantum Sensor Networks

Author:
Caitao Zhan

Supervisor:
Professor Himanshu Gupta
Committee:
Professor Samir Das
Committee:
Professor C.R. Ramakrishnan

A thesis proposal submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

Department of Computer Science

October 27, 2022

STONY BROOK UNIVERSITY

Abstract

Professor Himanshu Gupta
Department of Computer Science

Doctor of Philosophy

Transmitter Localization in Classical and Quantum Sensor Networks

by Caitao Zhan

In shared spectrum systems, it is important to be able to localize simultaneously present multiple intruders (unauthorized transmitters) to effectively protect a shared spectrum from malware-based, jamming, or other multi-device unauthorized-usage attacks. We address the problem of localizing multiple intruders using a distributed set of classical radio-frequency (RF) sensors in the context of a shared spectrum system. In contrast to single transmitter localization, multiple transmitter localization (MTL) has not been thoroughly studied. The key challenge in solving the MTL problem comes from the need to "separate" an aggregated signal received from multiple intruders into separate signals from individual intruders. We solve the problem via a Baysian-based approach and a deep-learning-based approach.

After addressing multiple transmitter localization with a network of classical sensors, we explore a network of quantum sensors and continue the work of transmitter localization using the quantum sensors. Quantum sensor network is a network of spatially dispersed sensors that leverage the quantum properties of light and matter, e.g., quantum coherance and quantum entanglement. We pose our transmitter localization problem as a quantum state discrimination problem and use the positive operator-valued measurement (POVM) as a tool for localization in a novel way. Quantum entanglement is a critical resource for the task of distributed quantum sensing. So we also investigate an efficient way to distribute or route the entangled pairs (EPs). Routing EPs is challenging because of the no-cloning theorem and the long-distance direct transmission of qubit states being infeasible due to unrecoveralble errors. We develop a heuristic algorithm that efficiently route EPs in a quantum network.

For the proposed work, we plan to keep the investigation of transmitter localization with a quantum sensor network. POVM is the current quantum measurement we are using, it is general but not very practical. Instead of POVM, we plan to use the projective measurement in the computational basis. To better optimize the measurement process, parameterized quantum circuit (quantum machine learning) will be utilized.

List of Papers

Papers Included in this Thesis Proposal

- **Caitao Zhan**, Himanshu Gupta, Arani Bhattacharya, Mohammad Ghaderibaneh, "Efficient Localization of Multiple Intruders in Shared Spectrum System," IPSN, Sydney, Australia, Apr. 2020.
- **Caitao Zhan**, Mohammad Ghaderibaneh, Himanshu Gupta, "DeepMTL: Deep Learning Based Multiple Transmitter Localization," WoWMoM, Pisa, Italy, June. 2021.
- **Caitao Zhan**, Mohammad Ghaderibaneh, Himanshu Gupta, "DeepMTL Pro: Deep Learning Based Multiple Transmitter Localization and Power Estimation," Pervasive and Mobile Computing[1], 2022.
- **Caitao Zhan**, Himanshu Gupta, "Localize Transmitter in Quantum Sensor Networks", IEEE ICC 2022 (Under review)
- Mohammad Ghaderibaneh, **Caitao Zhan**[2], Himanshu Gupta, C.R. Ramakrishnan, "Efficient Quantum Network Communication using Optimized Entanglement Swapping Trees," IEEE TQE, 2022.

Other Papers

- Mark Hillery, Himanshu Gupta, **Caitao Zhan**, "Discrete Outcome Quantum Sensor Networks", Physical Review A (Under review)
- Mohammad Ghaderibaneh, **Caitao Zhan**, Himanshu Gupta, "DeepAlloc: CNN-Based Approach to Efficient Spectrum Allocation in Shared Spectrum Systems," (Under submission)
- Arani Bhattacharya, **Caitao Zhan**, Abhishek Maji, Himanshu Gupta, Samir R. Das, Petar M. Djuric, Selection of Sensors for Efficient Transmitter Localization, IEEE/ACM Transactions on Networking 2021.
- Himanshu Gupta, Max Curran, **Caitao Zhan**. Near-Optimal Multihop Scheduling in General Circuit-Switched Networks, ACM CoNEXT 2020
- Arani Bhattacharya, **Caitao Zhan**, Himanshu Gupta, Samir R. Das, Petar M. Djuric. Selection of Sensors for Efficient Transmitter Localization, IEEE INFOCOM 2020

[1] An extension of the WoWMoM conference paper
[2] Only the balanced-tree section is included

Poster

- **Caitao Zhan**, Himanshu Gupta, Mark Hillery, "Discrete Outcome Quantum Sensor Networks" [3], Stony Brook University, Graduate Research Day 2022

[3]The poster has the same name as a PRA paper under review, but the content is different, and will later be written as a new PRA paper

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Wireless sensor network (WSN) is a network of spatially dispersed and dedicated sensors that monitor and record the physical conditions of the environment and forward the collected data to a central location. WSN can measure environment conditions such as temperature, sound, pollution levels, humidity, wind and radio spectrum. WSN refers to classical sensor networks since everything it involves is classical. A sensor network becomes a *quantum sensor network* (QSN) when the sensors leverage some quantum properties of light and matter, such as quantum coherance and quantum entanglement. Quantum sensors are extremely sensitive to physical quantities such as magnetic field, electric field, quadrature displacement and phase shift in the optic field.

Classical. WSNs have various applications. In this thesis, the application we focus on is spectrum surveillance and monitoring for security and threat detection. The *core problem* involved in this application is *transmitter localization*, and in particular, multiple transmitter localization (`MTL`) as the number of transmitter present in an area could be more than one and localizing multiple transmitters are not independent. The reason for being not independent is that a sensor receives an aggregated power from multiple transmitters and separating the power from different multiple sources is impractical. That an aggregated received power not able to separete is a big challenge for `MTL`. Furthermore, in a shared spectrum paradigm, presence of an evolving set of authorized users (e.g., primary and secondary users) adds to the challenge.

The shared spectrum paradigm composes an important background for our `MTL` work. The RF spectrum is a natural resource in great demand due to the unabated increase in mobile (and hence, wireless) data consumption [5]. The research community has addressed this capacity crunch via development of shared spectrum paradigms, wherein the spectrum is made available to unlicensed users (secondaries) as long as they do not interfere with the transmission of licensed incumbents (primaries). The fundamental objective behind such shared spectrum paradigms is to maximize spectrum utilization, the viability of such systems depends on the ability to effectively guard the shared spectrum against unauthorized usage. The current mechanisms however to locate such unauthorized users (intruders) are human-intensive and time-consuming, involving FCC enforcement bureau which detects violations via complaints and manual investigation [63].

Motivated by above, we seek for an effective technique that is able to accurately localize multiple simultaneous intruders and even in the presence of dynamically changing set of authorized users. Our solution assumes a network of crowdsourced sensors wherein relatively low-cost spectrum sensors are available for gathering signal strength

in the form of received power. We introduce two different approaches to the `MTL` problem. The first approach is a hypothesis-driven Bayesian approach, viz. maximum a posterior approach, where wherein each hypothesis is a configuration (i.e. a combination of ⟨location, power⟩ pair of the potential intruders), and the goal is to determine the hypothesis that best explains the sensor observations. The second approach is a deep learning-based approach. First, we encode the sensors' observation data into an image. Then, we frame `MTL` as a sequence of two steps: image-to-image translation and object detection, each of which is solved using a trained CNN model. The first step of image-to-image translation maps an input image representing sensor readings to an image representing the distribution of transmitter locations, and the second object detection step derives precise locations of transmitters from the image of transmitter distributions. Besides the location, the transmission power is another property of a transmitter that we wish to estimate. We introduces some novel methods to estimate the power of multiple transmitters. We also introduce a novel interapolation method for received signal strength.

Quantum. In the quantum side, we use QSN, instead of WSN, to continue solving the problem of transmitter localization. Albeit classical sensors are omnipresent, there are big motivations to explore quantum sensors. Quantum sensing is an emerging field that leverages the quantum properties of light and matter at atomic/subatomic scales and has the potential to sense signals at an unprecedented level of precision. Quantum sensing brings new opportunities to new and well-established problems. For example, physicists in the year 2016 used squeezed quantum states to improve the sensitivity of the Laser Interferometer Gravitational-wave Observatory (LIGO) detector and successfully detected gravitational waves. In [127], researchers use some distributed quantum RF-photonic sensors to estimate the amplitude and phase of a radio signal. They showed the performance of sensing a global property of the RF wave is enhanced by leveraging a shared multipartite entangled state produced by squeezed light. In their experiments, the estimation variance of RF amplitude and phase both beat the standard quantum limit by over 3 dB. The precision improvement factor of $1/\sqrt{N}$ for N sensors is known as reaching the Heisenberg limit.

Motivated by the above, we aim to leverage quantum sensors to perform some canonical tasks and thus open a new avenue of research. The canonical task we picked is RF transmitter localization [128, 135]. We consider a network of quantum sensors distributed in a geographic area and a single transmitter active in the area to be localized. We pose the localization problem as a quantum state discrimination problem [12]. In our approach, the quantum sensor network reports a quantum state, and we discriminate the quantum state via positive-operator valued measure (POVM) and the POVM's output indicates the transmitter location. The key challenge here is the scalability challenge, i.e., the method's time and space complexity grows exponentially against the number of sensors and the method's localization accuracy decrease against the number of discrete locations. To solve the challenge, we propose a two-level POVM method that is comprised of a coarse-level POVM and a fine-level POVM. The two level idea is effective and can be generalized into three levels and more.

Quantum entanglement is a phenomenon that has no counterpart in the classical world. It is the physical phenomenon that occurs when a group of particles (electrons, photons, etc) are generated, interact, or share spatial proximity in a way such that the quantum state of each particle of the group cannot be described independently of the state of the others, including when the particles are separated by a large distance. In

short, quantum entanglement is a specially strong correlation between multiple particles. In our context of QSNs, entanglement can be served as a resource to enhance the perforance of the QSN. For example, the $1/\sqrt{N}$ improvement factor mentioned above requires the initial probe state as an entangled state. The entanglement pair resource is generated at a single node, but the quantum sensors are spatially distributed. Thus, a major problem is to distribute (or route) the entanglement to the quantum sensors at a potentially large distance apart. This a challenging problem in the field of quantum communication. Physical transmission of quantum states accross nodes can incur irreparable communication errors, as the no-cloning theorem proscribes making independent copies of arbitrary qubits. The establish of entanglement over long distance is challenging due to the low probability of success of the underlying physical process (short distance entanglement and swapping). In this thesis, we propose an efficient heuristic approach that efficiently route an entanglement pair in a quantum network.

1.2 Thesis Statement

The thesis statement is: **Transmitters can be localized efficiently and accurately with our designed methods in classical and quantum sensor networks.** Consider some transmitters to be localized a geographpical area. We deploy a distributed set of classical or quantum sensors in the area. We aim to efficently and accurately localize the transmitters by processing the data received from the sensors intelligently. Besides the core goal of transmitter localization, we also solve closely related problems such as transmission power estimation, receive signal strength interpolation and quantum network routing of entanglement.

1.3 Contributions and Organization of this Thesis

Towards the goal of our thesis we make the following contributions:

- In Chapter 2, we introduce an efficient hypothesis-based Bayesian approach `MAP`* for multiple transmitter localization (`MTL`) problem (Section 2.3.1); A closed-form equation for the estimation of transmission power (Section 2.3.2); A novel received signal strength interpolation method inspired from the power law distribution (Section 2.3.3); Extend `MAP`* to accommodate the presence of authorized users (Section 2.4).

- In Capter 3, we introduce a deep learning-based approach `DeepMTL` for the `MTL` problem (Section 3.3, Section 3.4); Extend `DeepMTL` via deep learning models to accommodate the presense of authorized users (Section 3.5); A deep learning-based approach that estimates the transmission power of multiple transmitters (Section 3.6).

- In Chapter 4, we introduce the concept of quantum sensor networks and the model of a quantum sensor (Section 4.2); In the context of quantum sensor networks, we pose a transmitter localization problem as a quanum state discrimination problem and introduce a novel quantum localization method `POVM-Loc` and `POVM-Loc Pro` based on positive-operator valued measure (Section 4.3).

- In Chapter 5, we introduce an efficient heuristic algorithm `Balanced-Tree` for routing an entanglement pair. The algorithm is Dijkstra-based, and the path

selection metric is a closed-form expression that models a path as a tree near accurately (Section 5.4).

- In Chapter 6, we described our proposed work of this thesis.

Chapter 2

Efficient Localization of Multiple Intruders for Shared Spectrum System

We address the problem of localizing multiple intruders (unauthorized transmitters) using a distributed set of sensors in the context of a shared spectrum system. In contrast to single transmitter localization, multiple transmitter localization (`MTL`) has not been thoroughly studied. In shared spectrum systems, it is important to be able to localize simultaneously present multiple intruders to effectively protect a shared spectrum from malware-based, jamming, or other multi-device unauthorized-usage attacks. The key challenge in solving the MTL problem comes from the need to "separate" an aggregated signal received from multiple intruders into separate signals from individual intruders. Furthermore, in a shared spectrum paradigm, presence of an evolving set of authorized users (e.g., primary and secondary users) adds to the challenge.

In this chapter, we propose an efficient algorithm for the `MTL` problem based on the hypothesis-based Bayesian approach called `MAP`. Direct application of the `MAP` approach to the `MTL` problem incurs prohibitive computational and training cost. In this work, we develop optimized techniques based on `MAP` with significantly improved computational and training costs. In particular, we develop a novel interpolation method, `ILDW`, which helps minimize the training cost. We generalize our techniques via online-learning to the setting wherein there may be a set of dynamically-changing authorized users present in the background. We evaluate our developed techniques on large-scale simulations as well as on small-scale indoor and outdoor testbeds. Our experiments demonstrate that our technique outperforms the prior approaches by significant margins, i.e., error up to 74% less in large-scale simulations and 30% less in real-world testbeds.

2.1 Introduction

The RF spectrum is a natural resource in great demand due to the unabated increase in mobile (and hence, wireless) data consumption [4]. The research community has addressed this capacity crunch via development of *shared spectrum paradigms*, wherein the spectrum is made available to unlicensed users (secondaries) as long as they do not interfere with the transmission of licensed incumbents (primaries). E.g., in the recent years, the FCC has made available the CBRS band, i.e., the 3550-3700 MHz band within the 3.5 GHz band, for shared commercial use to allow other users to utilize the otherwise low-usage band which was previously reserved for incumbent users including US Navy radar operators.

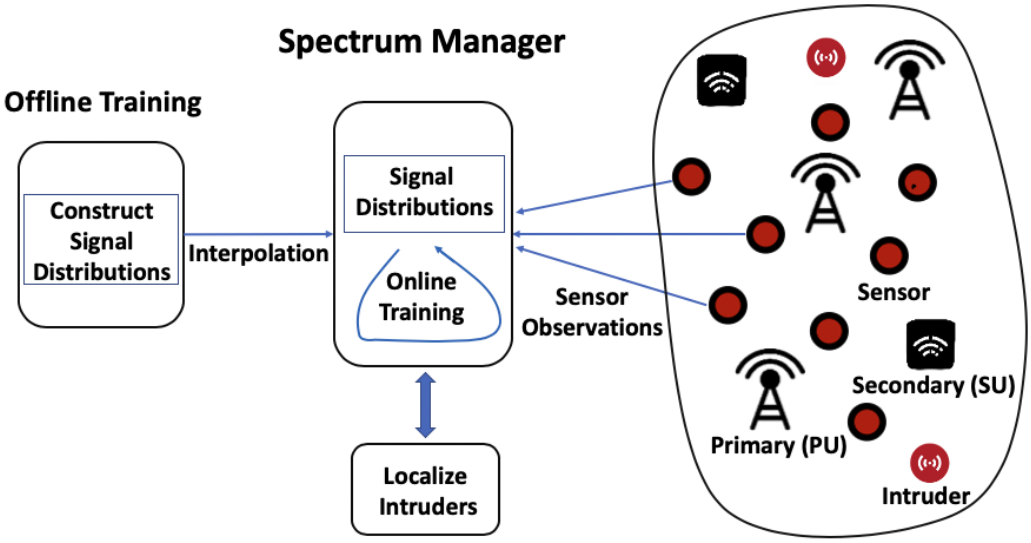


FIGURE 2.1: Overall approach to localize intruders in a shared spectrum system.

The increasing affordability of the software-defined radio (SDR) technologies makes the shared spectrums particularly prone to unauthorized usage or security attacks. With easy access to SDR devices [1, 2], it is easy for selfish users to transmit data on shared spectrum without any authorization and potentially causing harmful interference to the incumbent users. Such illegal spectrum usage could also happen as a result of infiltration of computer virus or malware on SDR devices. As the fundamental objective behind such shared spectrum paradigms is to maximize spectrum utilization, the viability of such systems depends on the ability to effectively guard the shared spectrum against unauthorized usage. The current mechanisms however to locate such unauthorized users (intruders) are human-intensive and time-consuming, involving FCC enforcement bureau which detects violations via complaints and manual investigation [63]. Motivated by above, we seek for an effective technique that is able to accurately localize multiple simultaneous intruders and even in the presence of dynamically changing set of authorized users. In the following, we begin with describing the multiple transmitter localization problem.

Multiple-Transmitter Localization (MTL). The transmitter localization problem has been well-studied, but most of the focus has been on localizing a *single* intruder at a time. However, it is important to localize *multiple* transmitters *simultaneously* to effectively guard a shared spectrum system. E.g., a malware or virus-based attachment could simultaneously cause many devices to violate spectrum allocation rules; spectrum jamming attacks would typically involve multiple transmitters. More importantly, a technique limited by localization of a single intruder could then be easily circumvented by an offender by using multiple devices. The key challenge in solving the MTL problem comes from the fact that the deployed sensor would receive only a sum of the signals from multiple transmitters, and separating the signals may be impossible. In addition, the other challenge that MTL in the context of shared spectrum system poses is the presence of authorized users—e.g., the incumbent users and the dynamic set of secondary users that have been allocated spectrum by the manager. To the best our knowledge, no prior localization work has considered the presence of authorized users.

The state-of-the-art technique for the MTL problem is the recent work [63], which essentially decomposes the MTL problem to multiple single-transmitter localization problems based on the sensors with the highest power readings in a neighborhood.

However, the technique has a few shortcomings: (i) it implicitly assumes a propagation model, and thus, may not work effectively in areas with complex propagation characteristics, (ii) it is not effective in the case of transmitters being located close-by, a key challenging scenario for MTL problem, and (iii) most importantly, it can't be extended effectively to incorporate background authorized users, a key requirement in the context of shared spectrum systems.

Our Approach. Transmitter localization is generally done based on observations at deployed sensors. In particular, as in prior works [63, 24], we assume a crowdsourced sensing architecture wherein relatively low-cost spectrum sensors are available for gathering signal strength in the form of received power. Our approach is a hypothesis-driven Bayesian approach, viz. *maximum a posteriori* (`MAP`) approach, wherein each hypothesis is a configuration (i.e. a combination of $\langle location, power \rangle$ pair) of the potential intruders, and the goal is to determine the hypothesis that best explains the sensor observations. This determination is done based on the distributions (gathered during a training phase) of sensor observations for each hypothesis. The `MAP` approach is known to have optimal classification accuracy, but (i) incurs prohibitive computation cost—exponential in number of potential intruders—when applied to the `MTL` problem, and (ii) requires significant amount of training cost. The focus of our work is to address these challenges, and design a viable `MAP`-based approach. In particular, using `MAP` as a building block, we develop an optimized approach that runs in polynomial time with minimized training cost. We extend our technique to work in presence of authorized users by incorporating online (real-time) training.

Motivation for `MAP`. Our motivation for using a `MAP`-based approach is multifold: First, with sufficient training data, `MAP` is known to deliver optimal classification accuracy for the MTL problem [42]. Second, the `MAP` approach doesn't assume any propagation model and thus works for arbitrary signal propagation characteristics. Third, it allows us to also estimate the intruder's transmit power, which can be very useful in some applications, e.g., where the penalty is proportional to the extent of violation. Last but not the least, it naturally extends to being able to handle a presence of an evolving set of authorized users.

Training Cost and Optimization. The benefits of a `MAP`-based approach come at a cost: the `MAP` framework requires prior training to build probability distributions (PDs) of sensor observations for each hypothesis. However, most of the training occurs offline, one-time, and can be automated e.g. via drones or robots. In our work, we develop strategies to minimize the training cost; in particular, we reduce the number of PDs to be constructed via a *novel interpolation scheme* suited to our unique setting, and evaluate the impact of reduced training on the localization accuracy. We note that the online training to incorporate presence of authorized users is needed only for the prevailing setting (of authorized transmitters and deployed sensors) and hence incurs minimal cost (see §2.4).

Overall Contributions. The goal of our work is to develop an efficient technique for accurate localization of simultaneously present multiple intruders in a shared spectrum system. The raw data are available at https://github.com/Wings-Lab/IPSN-2020-data. In this context, we make the following four specific contributions.

1. Design an efficient localization algorithm (`MAP`*) for the MTL problem, based on an optimal hypotheses-driven Bayesian approach. The designed approach predicts both locations and transmit powers of the intruders, and does not assume

any propagation model and thus, works for arbitrary signal propagation characteristics.

2. Extend the designed algorithm (`MAP`**) to localize effectively in the presence of background authorized users, i.e., primaries with possibly unknown parameters (e.g., location and transmit power) and an evolving set of secondary users.

3. Develop an effective interpolation scheme (`ILDW`) for our unique setting to reduce the one-time training cost of our scheme, without impacting the localization accuracy much.

4. Evaluate our techniques via large-scale simulations as well as over two developed testbeds (indoor and outdoor), and demonstrate the effectiveness of our developed techniques and their superior performance compared to the best-known techniques.

2.2 Problem, Related Work, and Methodology

In this section, we describe our model of the shared spectrum systems, formulate the `MTL` problem, and discuss related work. We also describe the building block of our approach, viz., a hypothesis-drived Bayesian localization approach (`MAP`).

Shared Spectrum System. In a shared spectrum paradigm, the spectrum is shared among licensed users (primary users, PUs) and unlicensed users (secondary users, SUs) in such a way that the transmission from secondaries does not interfere with that of the primaries (or secondaries from a higher-tier, in case of a multi-tier shared spectrum system [124]). In some shared spectrum systems, the location and transmit power of the primary users may be unavailable, as is the case with military or navy radars in the CBRS band [124]. Such sharing of spectrum is generally orchestrated by a centralized entity called *spectrum manager*, such as a spectrum database in TV white space [64] or a central spectrum access system in the CBRS 3.5GHz shared band [56]. The spectrum manager allocates spectrum to requesting secondaries (i.e., permission to transmit up to a certain transmit power at their location) based on their location, spectrum demand, configurations of the primaries, other active secondaries, prevailing channel conditions, etc.

Authorized and Unauthorized Users. Secondary users that have been explicitly given permission to transmit at their location are termed as *authorized users*; the primaries users are also considered as authorized users. Note that the set of authorized users evolve over time, as more and more SUs are allocated spectrum and as some SUs stop using the spectrum after a while. We can assume that each SU is allocated spectrum for a certain duration of time, after which it stops using the spectrum. Other users that transmit without explicit permission (for that given time) are referred to as *unauthorized users* or *intruders*.

Problem Setting and Formal Definition. Consider a geographic area with a shared spectrum. Without loss of generality, we assume a single channel throughout this paper (multiple channels are handled similarly). For localization of unauthorized users, we assume available crowdsourced sensors that can observe received signal in the channel of interest, and compute (totel) received signal strength indicator (RSSI)[1]. These sensors,

[1]We do not use angle-of-arrival (AoA) measurements [137] as they require additional and complex RF hardware.

being crowdsourced, may be at different locations at different times. At any given instant, the shared spectrum area has some licensed primary users and some active secondary users; the PU configurations may not be known as can be the case for military users. The centralized spectrum manager is aware of the set of active SUs at any time, as each SU request is granted for a certain period of time. In addition to the authorized users, there may be a set of intruders present in the area with each intruder in a certain "configuration" (see §2.2.2).

The `MTL` problem is to determine the set of intruders with their configurations at each instant of time, based on the set of sensor observations at that instant. See Figure 2.1. The basic `MTL` problem assumes no other transmissions (of authorized users) in the background. The more general `MTL` problem, where there may be an evolving set of authorized users in the background, is referred to as the `MTL-SS` problem. We address the `MTL` problem in §2.3, and then address the more general `MTL-SS` problem in §2.4.

2.2.1 Related Work

Localization of an intruder in a field using sensor observations has been widely studied, but most of the works have focused on localization of a single intruder [23, 43]. In general, to localize multiple intruders, the main challenge comes from the need to "separate" powers at the sensors [88], i.e., to divide the total received power into power received from individual intruders. Blind source separation is a very challenging problem; only very limited settings allow for known techniques [107, 72] using sophisticated receivers. In our context of hypotheses-driven approach, the challenge of source separation manifests in terms of a large number of hypotheses, a challenge addressed in §2.3. We note that (indoor) localization of a device [10] based on signals received from multiple reference points (e.g, WiFi access points) is a quite different problem (see [131] for a recent survey), as the signals from reference points remain separate, and localization or tracking of multiple devices can be done independently. Recent works on multi-target localization/tracking are different in the way that targets are passive [62, 32, 54], instead of active transmitters in this work.

In absence of blind separation methods, to the best of our knowledge, only a few works have addressed multiple intruder(s) localization, and none of these consider it in the presence of a dynamically changing set of authorized transmitters. In particular, (i) [63] decomposes the multi-transmitter localization problem to multiple single-transmitter localization problems based on the sensors with highest of readings in a neighbohood, (ii) [85] works by clustering the sensors with readings above a certain threshold and then localizing intruders at the centers of these clusters, (iii) [84] uses an EM-based approach. The techniques of [63, 84] assume a propagation model, while that of [85, 84] require a priori knowledge of the number of intruders present. We have compared our approach with [63, 85] in §2.5, while [84] has high computational cost and has also been shown to be inferior in performance to [63, 85] even for a small number of intruders. Other related works include (i) [49] that addresses the challenge of handling time-skewed sensors observations in the MTL problem, and (ii) [15] that addresses the sensor selection optimization problem for our proposed hypotheses-based localization approach.

2.2.2 `MAP`: Bayesian Approach for Localization

We localize intruders based on observations from a set of sensors. Each sensor communicates its observation to a centralized entity, the spectrum manager, which runs

an appropriate localization algorithm to localize the intruders. In particular, we use a hypotheses-driven Bayesian approach, as described below, where intruders are localized by determining the most-likely prevailing hypothesis; this is done based on joint probability distributions of the sensors' observations (constructed during a priori training). Below, we formalize the above concepts, and the basic localization approach.

Observation; Observation Vector. Throughout this paper, we use the term *observation* at an individual sensor to mean the received power over a time window of certain duration, in the frequency channel of interest (we assume only one channel). In particular, received power is computed from the FFT of the I/Q samples in the time window [23]. We use the term *observation vector* $\mathbf{x}$ to denote a vector of observations from a given set of distributed sensors, with each vector dimension corresponding to a unique sensor.

Hypotheses. Let $H_0, H_1, \ldots, H_m$ be the set of all hypotheses, where each hypothesis H_j represents a "configuration" of potential intruders. In this chapter, we largely assume an intruder's configuration to be comprised of just its location and transmit power, but the concept of configuration is quite general and could include any attributes (e.g., height, antenna direction, etc.) that affects how its transmitted signal is received at other locations. Moreover, for simplicity, we assume that each intruder transmits at a fixed power (which may be different for different intruders). Thus, in our context, a configuration is simply the set of (location, transmit power) pairs of the potential intruders. We assume a bounded number of intruders. We use H_0 to represent the hypothesis with no intruders. See Figure 2.2.

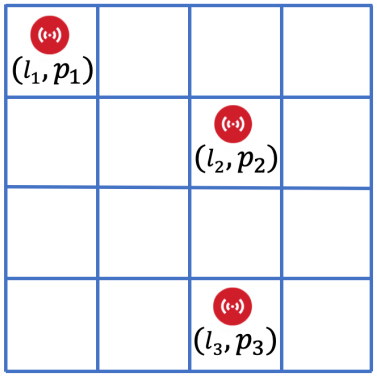


FIGURE 2.2: Illustration of a hypothesis formed of three transmitters.

If there is only one intruder, then each hypothesis represents the location and transmit power combination of the intruder, and determining the hypothesis is equivalent to localizing the intruder and estimating its power. If we allow multiple intruders at a time, the number of possible hypotheses can be exponential in the number of intruders; we will address this challenge in §2.3.

Inputs. For a given set of sensors deployed over an area, we assume the following available inputs, obtained via a priori training, data gathering and/or analysis:

- Prior probabilities of the hypotheses, i.e. $P(H_i)$, for each hypothesis H_i. Prior probabilities come from known knowledge about area, intruder's behavior, etc., and can be assumed to be uniform in absence of better knowledge.
- Joint probability distribution (JPD) of sensors' observations for each hypothesis. More formally, for each hypothesis H_j, we assume $P(\mathbf{x}|H_j)$ to be known for each observation $\mathbf{x}$ for the set of deployed sensors. The JPDs can be obtained from prior training, a combination of training and interpolation (§2.3.3), or even by assuming a propagation model to remove the training cost completely.

Maximum a Posteriori (MAP) Localization Algorithm. We use Bayes rule to compute the likelihood probability of each hypothesis, from a given observation vector $\mathbf{x}$:

$$P(H_i|\mathbf{x}) = \frac{P(\mathbf{x}|H_i)P(H_i)}{\sum_{j=0}^{m} P(\mathbf{x}|H_j)P(H_j)} \tag{2.1}$$

We select the hypothesis that has the highest probability, for given observations of a set of sensors. That is, the MAP Algorithm returns the hypotheses based on the following equation:

$$\arg \max_{i=0}^{m} P(H_i|\mathbf{x}) \tag{2.2}$$

The above MAP algorithm to determine the prevailing hypothesis is known to be *optimal* [42], i.e., it yields minimum probability of (misclassification) error. The above hypothesis-based approach to localization works for arbitrary signal propagation characteristics, and in particular, obviates the need to assume a propagation model. However, the above MAP algorithm does incur a *one-time* training cost to construct the JPDs.

2.3 MAP* Optimizing MAP for MTL

The MAP algorithm of §2.2.2 can be directly applied to localize multiple intruders with optimal localization accuracy. However, MAP incurs prohibitive computational cost especially for a large number of potential intruders. In particular, note that if there are L potential locations, up to T potential intruders, and W possible discrete transmit-power levels, then the hypotheses-driven MAP algorithm needs to consider $(LW)^T$ hypotheses—making its runtime complexity exponential in number of potential intruders, and thus, making it impractical for localizing even a moderate number of intruders present simultaneously. In addition, MAP also incurs a high training cost. In the following subsections, we develop an optimized algorithm called MAP* based on MAP but with significantly improved computational and training cost. We start with optimizing the computation cost in §2.3.1. In the following subsection §2.3.2, we derive a closed-form expression to efficiently estimate intruder's power in the *continuous* domain. Finally, we discuss optimizing the training cost via a novel interpolation scheme ILDW.

2.3.1 Optimizing Computation Time

Basic Idea. Note that the MAP's exponential time complexity is due to the exponential number of *combinations* of locations and/or powers of the potential intruders. To motivate our proposed optimized approach, consider a simple example of 2 intruders with fixed power p in a large area. Assume that the "transmission radius" r for power p is much smaller than the area; we define the *transmission radius* as the range till which the received signal is more than a certain noise floor. The key observation is that if the intruders are far away (isolated) from each other (specifically, more than $2r$ distance away), then they could be localized independently. If the intruders are closer, then there is a need to separate aggregated signal at some of the sensors and hence we must apply the standard MAP algorithm *within that "subarea"*; however, since each such subarea is small (a disk of $2r$ radius around each possible location), the computation time is reduced significantly. However, since we do not a priori know the configurations of intruders, we need to consider appropriate possibilities.

In essence, our optimized approach is a divide-and-conquer approach, consisting of a sequence of two procedures each of which is executed iteratively. The first procedure focuses on localizing "isolated" intruders (if any) independently, while the second procedure localizes the remaining intruders—by considering all possible subareas as suggested above. The challenge lies in modifying the MAP algorithm for each iteration of the above procedures—as the hypotheses to consider across iterations of the procedures are not disjoint. We now describe each of the procedures.

Procedure 1. Localize Isolated Intruders. Informally, in this procedure, we localize intruders that are sufficiently separated from other intruders. In other words, we localize intruders x that are surrounded by sensors that receive most of their received power from x. More formally, we localize an intruder x at location l if (i) l's "neighborhood" has at least 3 sensors that receive most of their power from x, and (ii) there are no other intruders in the "vicinity" of l. In essence, we iterate over all locations l, and localize an intruder at l if the above conditions are satisfied with high enough probability, based on the readings of sensors around l. The precise definition of neighborhood above must depend on x's transmission radius which depends on its transmit power; however, as x's transmit power is unknown, we iterate over smaller and smaller neighborhoods.

We now formally describe the procedure. Let R_p denote the transmission radius for a transmit power of p. Let R denote the maximum transmission radius, i.e.,

$$\max_p R_p.$$

In the below description, we use a fractional value f to define a neighborhood and vicinity size. We start f equal to 1, use a disk of radius fR_p as a neighborhood and $R + fR_p$ as the vicinity, and iterate over the procedure for reduced values of f.

(a) Let $f = 1$.

(b) For each location and power pair (l, p), compute $P(\mathcal{H}_{l,p}|\mathbf{x}_{l,p})$ using a form of Equation 2.1 over appropriate JPDs. Here:

- $\mathcal{H}_{l,p}$ represents the hypothesis that an intruder is at location l and using p transmit power. We also implicitly assume that there is no other intruder present within a distance of $R + fR_p$ from l; this ensures that the observations in $\mathbf{x}_{l,p}$ are only due to the intruder at l. See Figure 2.3.
- $\mathbf{x}_{l,p}$ represents the observation vector for all sensors, but the sensors that are within a radius of fR_p around l use an observation of "residual" received powers, as defined below, while the remaining sensors (outside the radius of fR_p around l) use an observation of the "noise floor" (in essence, we are "zeroing" the observations of the far-away sensors). See Figure 2.3.

(c) Denote (l, p) pairs that have $P(\mathcal{H}_{l,p}|\mathbf{x}_{l,p})$ higher than a certain threshold as *peaks*. If a location l is a peak and there are no other peaks within a distance of $R + fR_p$, then **localize an intruder at l with transmit power p.**

(d) For each sensor s, define its *residual received power* (RRP) as the total received power reduced by the sum of mean powers received from already localized intruders; the desired mean values are available from the given JPDs.

(e) Reduce f and go back to step #2 above, unless no new intruders were localized in (c) above. In our experiments, we used $f = 1, 1/2, 1/4$ and $1/8$.

The above procedure is partly inspired by the recent localization work [85]. However, instead of discarding sensors based on their individual power and clustering the rest as in [85], we "discard" sensors based on their neighborhood readings (i.e., likelihood $P(\mathbf{x}|H_i)$ values) and then "cluster" the remaining sensors. Also, we "cluster" iteratively, for smaller and smaller neighborhoods.

Procedure 2. Localize Intruders Situated Close-By. Once we have localized separated intruders as above, we now localize remaining intruders, if any, by applying the general `MAP` algorithm independently over "subareas" that still have some sensors with high-enough RRP (residual received power), but no intruder localized in the "vicinity."

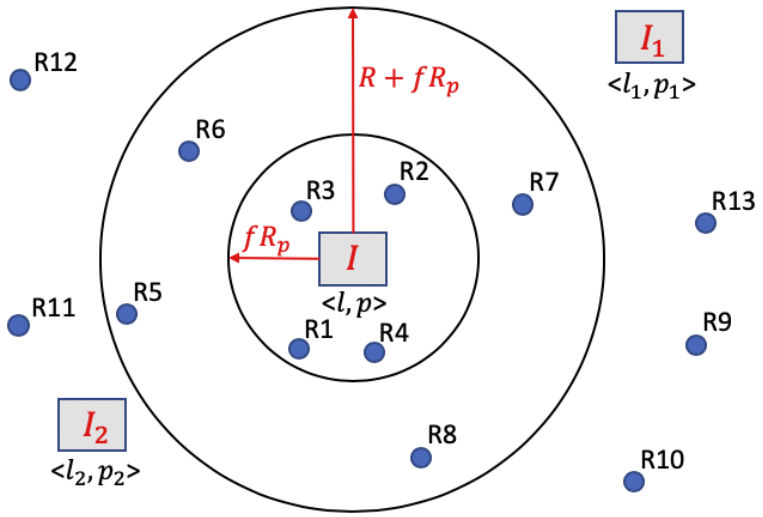


FIGURE 2.3: Illustration of Hypothesis $\mathcal{H}_{l,p}$ in Step (b) of Procedure 1. Here, the intruder I at location l is transmitting at power p, with no other intruder within a distance of $R + fR_p$ from I. The observation vector $\mathbf{x}_{l,p}$ consists of residual received powers from $R1$ to $R4$, and "noise floor" from the remaining sensors.

Formally, the procedure is as follows. Let T be the maximum number of intruders allowed within a disk of radius R, the maximum transmission radius.

(a) Let s be the sensor with highest RRP; if s's RRP is below a certain threshold (tantamount to noise), then quit.
(b) For $t = 2$ to T: Use `MAP` (from §2.2.2) to try to localize t transmitters within a disk of radius R around s, using observations of sensors within a radius of $2R$ from s. We use a certain threshold for a posterior probability, in a similar way as for Procedure 1.
(c) Update RRP of each sensor, and go to step (a) above.

Time Complexity. The worst-case time complexity of the first procedure is $O(LWG_R \log(G_R))$, where L and W are the number of potential locations (total grid cells) and transmit power levels respectively, and G_R is the maximum number of grid cells within a transmission range of an intruder. Here, the first term $O(LWG_R)$ is the time to compute the likelihood values in each iteration, since the number of sensors involved in each computation is at most G_R. Note that the number of iterations is bounded by $\log(G_R)$, as f is reduced by a constant multiplicative factor. The worst-case time complexity of the second procedure is $O(G_R(G_R)^T)$ where T is the maximum number of intruders allowed/possible in a transmission region (i.e., a circle of radius at most R). Thus, the overall time complexity of the above localization algorithm is $O(L.W.G_R.\log(G_R) + G_R.(G_R)^T)$. Generally, we would expect T to be a small constant, as more than 3 intruders in a R-radius region with a R transmission range would interfere with each other. If we also consider G_R as a small constant, the overall time complexity can be considered to be $O(L.W)$. In the following subsection, we further reduce the time complexity by removing the factor of W.

2.3.2 Intruder Power Estimation in the Continuous Domain

In this subsection, we derive a *closed-form* expression to estimate an intruder's power in the continuous domain, for the special case of single intruder and Gaussian probability distributions [53]. The derived result essentially removes the assumption of discrete

power levels, and reduces the number of hypotheses to consider by a factor of W. We use this result within Procedure 1 of previous subsection to further optimize its time complexity and performance.

Estimating Intruder Power, Given a Location. Consider the special case of a single intruder in an area. In this case, each hypothesis can be represented as $\mathcal{H}_{l,p}$, for each location l and power p of the potential intruder. Let us focus on a particular location l^* and the corresponding hypotheses $\mathcal{H}_{l^*,p}$. For a given observation vector $\mathbf{x}$, we wish to estimate the power P that corresponds to the hypothesis with maximum likelihood among the hypotheses $\mathcal{H}_{l^*,p}$.

$$P = \arg max_p P(\mathcal{H}_{l^*,p}|\mathbf{x})$$

The value P can be computed by computing $P(\mathcal{H}_{l^*,p}|\mathbf{x})$ for each p, but our goal is to derive a closed-form expression for P from the given JPDs; such an expression yield power estimate in continuous domain without computing $P(\mathcal{H}_{l^*,p}|\mathbf{x})$ for each possible discrete p.

For each sensor (location) j, let $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p^*})$ represent the probability distribution (PD) of j's observations $\mathbf{x}_j$ when the intruder is at l^* transmitting with power p^*, the power used at training. For a fixed l^* and p^*, the set of PDs $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p^*})$ are equivalent to the JPDs defined in §2.2 under the assumption of conditional independence[2]. Let us assume that the above PDs are Gaussian distributions [53], and thus, can be represented as $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p^*}) = N(\mu_j, \sigma_j^2)$ for a given l^* and p^*. In the above setting, the power value P that maximizes $P(\mathcal{H}_{l^*,p}|\mathbf{x})$ can actually be derived as a closed-form expression; we state the result formally in the below lemma.

Lemma 1. *Consider the special case of a single intruder in an area. For a specific location l^* and power p^* (the only power used during training), let $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p^*})$ represent the PDs of the sensor obversations at location j. Now, given the above PDs for various j and an observation vector $\mathbf{x}$, the power value $P = \arg max_p P(\mathcal{H}_{l^*,p}|\mathbf{x})$ is given by:*

$$p^* + \frac{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}(x_j - \mu_j)}{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}},$$

where $\gamma = \prod_{j=1}^{S} \sigma_j^2$ and S equals to the number of sensors in the neighborhood of l^*. ■

The proof is in Appendix A.1. Here, we give its intuition based on a special case. Consider the special case wherein each σ_j is 1 for all j. In this special case, the Lemma's equation reduces to $P = p^* + \frac{\sum_{j=1}^{S}(x_j - \mu_j)}{|S|}$, which implies that if each observation x_j is c more than its mean μ_j then P is also c more than p^*. We note that the above result does *not* extend to the case of multiple intruders. In short, the proof is a process of solving maximum likelihood esitmaion and multiple intruders introduce transcendental functions, thus cannot derive a closed-form solution.

Use of Lemma 1 in MAP*. For localization of multiple intruders, Lemma 1 can only be used in Procedure 1 of §2.3.1, due to its assumption of a single intruder. In particular, we can Procedure 1 of §2.3.1 as follows.

[2]PD $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p})$ can be computed $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p^*})$ for any p, as the path-loss can be assumed to be independent of the transmit power, and JPD $\mathcal{P}(\mathbf{x}|\mathcal{H}_{l^*,p})$ can be computed as product of PDs $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p})$ due to the conditional independence assumption.

- We replace R_p by R, the maximum transmission radius.
- For each location l, using Lemma 1, we first compute the power $p(l)$ such that the hypothesis $\mathcal{H}_{l,p(l)}$ has the most likelihood (among the hypotheses at l) using the observations from sensors within a radius of R.
- Then, in the rest of the procedure, we only consider the (location, power) pairs of the type $(l, p(l))$ for any l.

Rest of the Procedure 1 remains unchanged. The above change has two benefits. First, the powers predicted in Procedure 1 are now continuous rather than discrete. Second, the above removes the factor of W from the time complexity of `MAP`* and reduces it to $O(LG_R \log(G_R) + G_R(G_R)^T)$ which becomes $O(L)$ if we consider G_R and T to be relatively small constants.

2.3.3 `ILDW`: Optimizing Training Cost

As in supervised machine learning algorithms, our Bayesian approach also needs training data. We use the term *training* to denote the process of collecting data and building up the JPDs for the hypotheses. Note that this training phase is done only one-time,[3] and hence, a certain cost is acceptable. The training cost incurred during such data gathering depends greatly on the exact mechanism used for such purposes, e.g., drones with appropriate routes can be used to gather such data [91]. In general, the cost of training would depend on the number of JPDs that need to be constructed, with the cost reduced with reduction in the number of JPDs needed. In this subsection, we design effective *interpolation* schemes that are useful in reducing the number of JPDs gathered which in turn will reduce the overall training cost. Note that reduction in JPDs constructed from raw data is bound to negatively impact the accuracy—we will evaluate this trade-off in our evaluations and show that impact on accuracy is minimal even with significant reduction in training cost.

Probability Distributions. First, we note that making the following reasonable assumptions and observations can greatly reduce the number of JPDs/PDs to be constructed.

- If we assume conditional independence of sensor observations, then JPDs can be computed from independently constructed probability distributions (PDs) of received powers at *individual sensors*.
- Since received power at a sensor location x due to multiple transmitters is merely a sum of received powers [95, 63] due to individual transmitters, we can compute PD at x for a particular hypothesis involving a set S of intruders from PDs due to each individual intruder in S.
- Lastly, we need to only construct a PD for one transmit power for each transmitter and sensor location pair, since path-loss is independent of transmit power.

[3] JPDs depend on the channel state and hence, must be updated periodically to account for any changes in the environment (e.g., terrain, buildings, etc.); however, such environment changes are infrequent. Also, note that the online-training of §2.4 is done repeatedly, but only for specific sensors and authorized users, and thus incurs minimal cost. See [133] for spectrum sensing in both spatio and temporal domains.

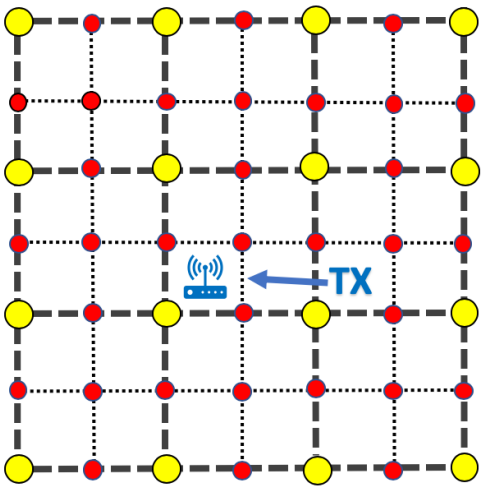


FIGURE 2.4: Training for PDs at coarse-grained locations (yellow bigger dots), while estimating PDs using interpolation at the remaining fine-grained locations (red smaller dots).

Based on the above observations, if there are L discrete locations in an area for sensors or intruders, then a `MAP`-based approach requires L^2 PDs. Below, we propose to minimize the number of PDs to be constructed via data gathering/training, by estimating the remaining unconstructed PDs via interpolation.

Minimizing Training Cost with `ILDW`. Consider a particular location l^* of a potential intruder. Our eventual goal is to compute the PD for each of the L possible sensor locations for this location l^* of a potential intruder; a PD may be computed either by constructing it directly from gathered sensor observations or by estimation via interpolation from the constructed PDs. In particular, for effective interpolation, we construct PDs at coarser-grid sensor locations, and estimate via interpolation the PDs at the remaining finer-grid locations. See Figure 2.4. The exact coarseness at which the PDs are constructed is determined by the accuracy of the interpolation scheme for a given area and/or the impact on localization accuracy due to estimated PDs. Below, we describe the interpolation scheme that we use for our purposes.

`ILDW` Interpolation Scheme. Consider a fixed transmitter location l^*, and let us assume locations $R_1, R_t, \cdots, R_n$ for which we know the path loss from l^*. Now, consider a new point R_0 for which we wish to estimate the path-loss from l^*. This is a traditional interpolation problem and well-known schemes such as inverse distance weighting (IDW), Ordinary Kriging (OK), k-NN, etc. have been evaluated even in the special context of signal strength or received power [24]. However, our specific context has an unique element. We *know* the location l^* of the transmitter from which the path-loss is being estimated—as we are in the training phase wherein we are gathering observations with transmitter at l^*. In light of the above unique element of our setting, and the observation of wireless signal characteristics, we use a custom interpolation technique which is a nontrivial modification of the IDW scheme, called *inverse log-distance weighting* (`ILDW`). The traditional IDW interpolation scheme estimates the path loss at R_0 by taking a weighted average of the path-losses at $R_1, R_t, \cdots, R_n$, with the weight being the inverse of the distance from R_0.

In our proposed `ILDW` scheme, we still estimate the path loss at R_0 as a weighted average of values at R_i's, but assign weights differently. In particular, we assign the weight for the point R_i as the inverse of the "distance" between R_0 and R_i in the

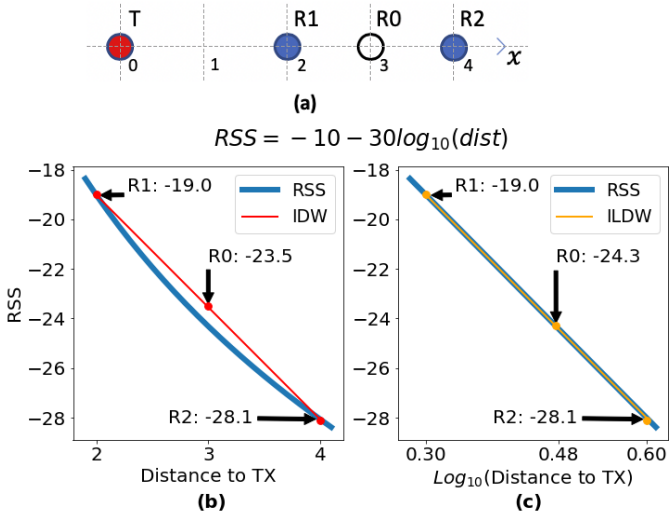


FIGURE 2.5: Illustration of `ILDW` vs. IDW. (a) Transmitter (T), points with known (R1 and R2) and unknown (R0) received signal strength (RSS) values. (b) Log-normal RSS function (= -10 - $30\log_{10}$(distance)) plotted for varying distance from the transmitter T, along with IDW-estimated RSS value at a point between R1 and R2. (c) Log-normal RSS function and ILDW-estimated RSS value at a point between R1 and R2, plotted on a logarithmic distance scale.

domain where each point is represented merely by its logarithmic distance from l^*, the known transmitter's location—i.e., each point R_i is mapped to a point $\log d(R_i, l^*)$ on a line. This mapping is motivated by the expectation that the actual path loss would be somewhat similar to the log-distance path loss. Thus, the weight for the point R_i is assigned to be

$$w_i = \frac{1}{|\log d(R_i, l^*) - \log d(R_0, l^*)|},$$

where $d()$ is the Euclidean distance function and the path loss at R_0 is estimated as:

$$\mathbf{u_0} = \frac{\sum_{i=1}^{n} w_i \mathbf{u_i}}{\sum_{i=1}^{n} w_i},$$

where $\mathbf{u_i}$ denotes the path loss at point R_i from l^*. In the above equation for weights, if denominator is zero, then we assign w_i to be equal to the maximum of the weights among the given points (and if all denominators are 0, each weight is assigned to be 1). For an illustration of the above scheme, see Figure 2.5. In the IDW scheme, R_1 and R_2 will get equal weights, but under the `ILDW` scheme they will get weights of 5.57 and 8.00 respectively. More importantly, it can be easily shown that, for log-distance path loss, `ILDW` estimates the path loss for R_0 accurately from two unknown points R_1 and R_2, if $d(R_1, l^*) < d(R_0, l^*) < d(R_2, l^*)$.

The above discussion has been on using `ILDW` for estimating path-loss values. In general, it can be easily used to estimate PDs from the PDs at neighboring points—essentially, we can use `ILDW` to estimate both the mean and standard deviation of a Gaussian PD from other means and standard deviations respectively.

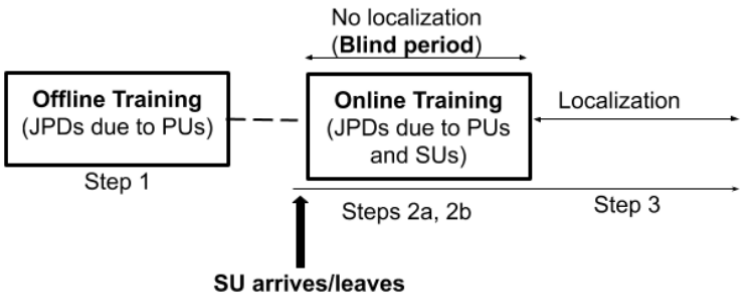


FIGURE 2.6: MAP**'s overall approach

2.4 MAP** Localizing in Presence of Authorized Users

We have implicitly assumed till now that the only transmitters present in the area are the intruders which need to be localized. In this section, we adapt our MAP* approach described in the previous section to the setting wherein there may be authorized transmitters in the background and the localization technique must take their presence into account. In particular, in a shared spectrum paradigm, there are primary users and an evolving set of active secondary users transmitting in the background. The key challenge comes from the fact that the set of authorized users is not static and changes over time as allocation requests are granted and/or active secondary users become inactive over time.

One simple way to handle background users is to just localize every transmitter, and then remove the authorized users. However, any localization approach (including ours) is susceptible to performance degradation with increase in number of transmitters to be localized, especially if some of them are situated close together. Thus, this simple approach of localizing every transmitter is unlikely to be effective, as shown in our evaluations, especially when the number of primaries and active secondaries can be large. Thus, here, we develop an approach based on learning PDs in real-time in response to changes in the set of secondary users.

MAP**: Localizing with Authorized Users. Our problem is to localize intruders in a shared spectrum system with fixed primaries and changing set of secondaries. Our MAP** approach uses a combination of a priori (offline) and online training to construct JPDs for appropriate hypotheses based on gathered observations, and then use these JPDs to localize intruders in real-time using the MAP* approach described in the previous section. We start with defining a few useful notations.

We use $\mathcal{R}$ to denote the set of (fixed) primaries, and $\mathcal{K}$ to denote the set of secondaries at a given instant, and $\mathcal{I}_j$ to denote the j^{th} configuration of intruders (we can assume the zero-th configuration to represent no intruders). We use $\tau = \mathcal{R} \cup \mathcal{K} \cup \mathcal{I}_j$ to denote the set to all transmitters (authorized and unauthorized) at a given instant. Finally, we use $\mathcal{P}(\mathbf{x}|(\tau = X))$ to denote the joint probability distribution (JPD) of observation vectors from the deployed sensors when the prevailing hypothesis is that the set τ of transmitters is X. MAP** is the sequence of following steps.

1. (Offline Step.) Construct JPDs $\mathcal{P}(\mathbf{x}|\mathcal{R})$ and $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{I}_j \cup \mathcal{R}))$ for all j. Since these JPDs are independent of the secondaries, they do not change and can be done once a priori.
2. (Online Steps.) Whenever $\mathcal{K}$ (set of secondaries) changes:
   (a) Construct JPD $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{R} \cup \mathcal{K}))$.

(b) Compute $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{R} \cup \mathcal{I}_j \cup \mathcal{K}))$ for all j, from above constructed JPDs, viz., $\mathcal{P}(\mathbf{x}|\mathcal{R})$, $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{I}_j \cup \mathcal{R}))$, and $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{R} \cup \mathcal{K}))$. See the below observation.

3. (Real-time Localization.) Periodically, each sensor sends its observation to a centralized entity (spectrum manager) which uses MAP* to localize any intruders present. Here, localization essentially means determining the most likely prevailing hypothesis among the hypotheses $\tau = (\mathcal{R} \cup \mathcal{I}_j \cup \mathcal{K})$, based on the JPDs $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{R} \cup \mathcal{I}_j \cup \mathcal{K}))$ constructed in earlier steps.

Note that steps 1 and 2a are essentially learning the authorized users' signal charecteristics and view them as the "background signals". If there are no authorized users, then the background signals are "quite". Else, then the background signals have some "sound". We now state the observation that forms the basis of JPD computation in Steps 2b; note that the noise due to sensor's hardware gets duplicated when "adding" two JPDs, but can be easily removed.

Observation 1. *The JPD* $\mathcal{P}(\mathbf{x}|(\tau = A \cup B))$ *and be computed from JPDs* $\mathcal{P}(\mathbf{x}|(\tau = A))$ *and* $\mathcal{P}(\mathbf{x}|(\tau = B))$. *Similarly, JPD* $\mathcal{P}(\mathbf{x}|(\tau = A))$ *can be computed from the JPDs* $\mathcal{P}(\mathbf{x}|(\tau = A \cup B))$ *and* $\mathcal{P}(\mathbf{x}|(\tau = B))$.

Blind Period due to Step 2. Note that the steps 2a and 2b construct or compute the JPDs needed for localization, and thus, during their execution, the localization cannot be done. Thus, it is important that the duration of this "blind period" in minimal. Fortunately, step 2b being a simple mathematic computation takes only in the order of milliseconds under efficient implementation, while 2a merely entails gathering a sufficient number of observations to construct the desired JPD which could take anywhere from milliseconds to a few seconds, as an observation takes only a fraction of a millisecond [23].

Mobility of Users and Sensors. We note that MAP* works seamlessly for mobile intruders and sensors, due to the constructed PDs. However, MAP** has the following limitation: the sensors must remain static in between two consecutive online-training periods (i.e., step 2 of above). If a sensor X moves, then either X's observation must be ignored, or that X needs to online-train itself in its new location (and there should be no intruders during this individual online-training phase). Note that active SUs are expected to remain static anyway, as they are allocated spectrum for a specific location.

2.5 Large-Scale Simulation Results

To evaluate our techniques in a large scale area (a few kms square), we conducted simulations over a geographic area using path-loss values from the Longley-Rice propagation model generated by open sourse software SPLAT! [79]. We describe the simulation setting below and discuss the results.

2.5.1 Settings

Generating Probability Distributions. To evaluate our techniques over a large area with 100s of sensor nodes, we need to run simulations with an assumed propagation model. We use the well-known Longley-Rice [28] Irregular Terrain With Obstruction Model (ITWOM), which is a complex model of wireless propagation based on many parameters including locations, terrain data, obstructions and soil condition etc. and such. We consider an area of 4km $\times$ 4km in the NY state and use the 800 MHz band for

SPLAT! We discretize the area using 40 vertical and 40 horizontal grid lines—yielding 1600 cells each of size 100m × 100m. To generate a probability distribution (PD) at a sensor location x due to a transmitter at location l transmitting at power p^*, we compute the received power at x using transmit power minus path-loss from SPLAT!, and use it as the mean of the probability distribution. For the complete PD, we assume Gaussian distributions and use a standard deviation between 1 and 3, with higher values for pairs (x, l) with smaller distance. As mentioned before, the PD due to multiple simultaneous transmitters can be computed as just a "sum" of the Gaussian distributions due to individual transmitters [95, 63].

Algorithms Compared. For the `MTL` problem, we compare our `MAP*` algorithm with `SPLOT` [63] and `CLUS` [85] (see §2.2.1). As mentioned before, [84] has been shown to be inferior in performance to both `SPLOT` and `CLUS` in their respective works, and thus, not evaluated here. `CLUS` uses k-means [90] for clustering, and needs to be provided with the number of clusters. To do a somewhat fair comparison, we provide `CLUS` with a *range* of the number of intruders and use the elbow-point method to pick the best number of clusters/intruders. In particular, the range of intruders passed to `CLUS` is 1 to $2x$, where x is the actual number of intruders present.

TABLE 2.1: Simulation Evaluation Parameters.

Param.	Value	Description
$Q_1^{'}$	0.6	Threshold for Procedure 1's hypothesis posterior
$Q_2^{'}$	0.1	Threshold for Procedure 2's hypothesis posterior
R	1000	Transmission radius when power is p^*, (m)
p^*	30	Transmit power during training, (dBm)
δ_p	2	Range of intruders' power is $[p^* - \delta_p,\ p^* + \delta_p]$

For `SPLOT`, we use the same set of parameters values as in [63] except that we use the confined area radius to be 800m for our large area setting ([63] only considered small 15m × 15m areas; 800m is roughly the maximum transmission radius in our large-scale setting and other values yielded worse results). Table 2.1 gives the main parameters of `MAP*` used in our evaluations. Recall that the transmission radius is the distance between the TX and RX for which the RX's RSS is at the noise floor (we use -80dBm).

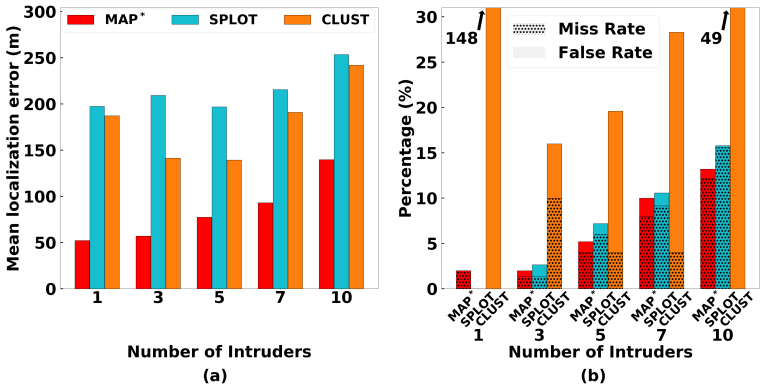


FIGURE 2.7: Localization performance of various algorithms in a large scale area, for varying number of intruders

2.5.2 Five Evaluation Metrics.

We use the following metrics to evaluate the localization methods.

1. Localization error (L_{err}).
2. Miss rate (M_r).
3. False alarm rate (F_r).
4. Power error (P_{err}).

The above metrics are best explained using a simple example. Given a multi-intruder localization solution, we first compute the L_{err} as the minimum-cost matching in the bi-partite graph over the ground-truth and the solution's locations, where the cost of each edge in the graph is the Euclidean distance. We use a simple greedy algorithm to compute the min-cost matching. The unmatched nodes are regarded as false alarms or misses. E.g., if there are 4 intruders in reality, but the algorithm predits 6 intruders then it is said to incur 0 misses and 2 false alarms and if it predicts 3 intruders then it incurs 1 miss and 0 false alarms. The M_r and F_r metrics are on a per-intruder basis, so in the above two examples: M_r is 0 and 1/4 and F_r is 2/4 and 0. In the plots, we stack miss rate and false alarm rate together to show the overall difference between the true number of intruders and predicted number of intruders. P_{err} is the average difference between the predicted power and the actual power of the matched pair in the above bi-partite graph.

Finally for interpolation schemes, we use the metric (5) interpolation error (I_{err}) defined as the estimated path-loss minus the ground-truth path-loss value.

2.5.3 Results

In this subsection, we evaluate the performance of our techniques for varying parameter values, viz., number of intruders and sensors in the field, and training cost. Here, the training cost is defined relative (specifically, as a percentage of) to the full training scenario wherein we construct each of the 1600×1600 PDs (one for each pair of transmitter and sensor locations) directly from observations. E.g., x% training cost indicates that we construct $1600 \times (16x)$ PDs directly, and interpolate the remaining $1600 \times (1600 - 16x)$ PDs; our proposed interpolation scheme only interpolates for sensor locations. In general, when we vary a specific parameter, the other parameters are set to their default values which are: 9% for training cost, 5 for number of intruders, and 240 for number of sensors. For each experiment, the said number of sensors and intruders are deployed randomly in the field, with the intruders deployed in the continuous location domain while the sensors deployed only at the centers of the grid cells. Each data point in the plots is an average of 50 experiments.

Varying Number of Intruders. First, we compare the localization accuracy of various algorithms for varying number of intruders. See Figure 2.7. We vary the number of intruders from 1 to 10. We observe that the localization error of `MAP`* is the minimum across the three algorithm. The localization error is 45% – 74% less than `SPLOT`. In terms of the M_r and F_r, `MAP`* also performs others which confirms the overall performance of `MAP`* to be the best among the algorithms compared. In terms of absolute performance, note that the localization error of 50-150m indicates an error of 1-2 grid cells, and thus is minimal in the context of the large area of 4km by 4km with 1600 cells and a sensor population of 240. Investigating further, we observe that misses in `MAP`* are

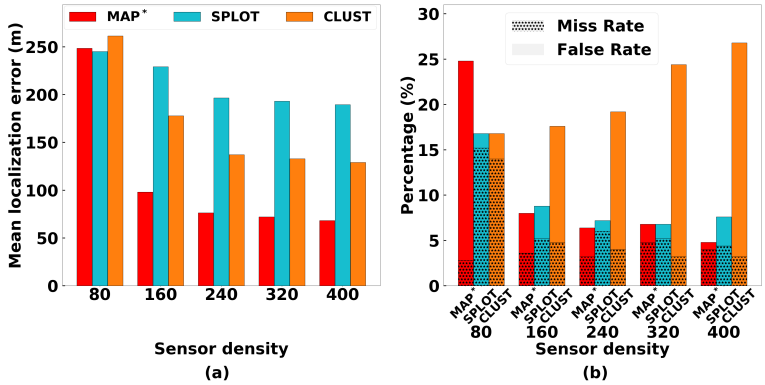


FIGURE 2.8: Localization performance of various algorithms in a large scale area, for varying sensor density

TABLE 2.2: MAP* Power Error (dB)

# Intru.	MAE	ME
1	0.56	-0.07
3	1.02	0.89
5	1.31	0.97
7	1.52	1.16
10	1.47	1.04

mostly due to the interpolated PDs (note that only 9% of the PDs are constructed from the actual sensor observations, and the remaining 91% are interpolated), while SPLOT's misses are mainly from the case of two or more intruders being close to each other. This demonstrates the superior ability of MAP* to localize intruders that are close-by via the designed sequence of Procedures 1 and 2.

Intruder Power Estimation, and Computation Time. Table 2.2 shows the mean absolute error (MAE) and mean error (ME) of the intruder's predicted power by MAP*. Note that CLUS and SPLOT do not predict intruder's power, and hence, not shown. We observe that MAP* is able to predict intuder's power quite accurately. The errors increase with the increase in number of intruders. Also, the mean error begins at near zero and then turns positive. Table 2.3 shows the running time of various algorithms over an Intel i7-8700 3.2 GHz processor. We see that CLUS is the fastest, and the running times of MAP* and SPLOT are comparable for small number of intruders, but for larger number of intruders, MAP* takes longer time than SPLOT mainly because of more number of iterations of the computationally-intensive Procedure 2.

Varying Sensor Density. We now vary the total number of sensors in the field, and

TABLE 2.3: Running time (s)

# Intru.	MAP*	SPLOT	CLUS
1	0.55	0.56	0.03
3	1.07	1.02	0.11
5	5.74	1.35	0.23
7	8.14	1.63	0.30
10	16.50	1.89	0.41

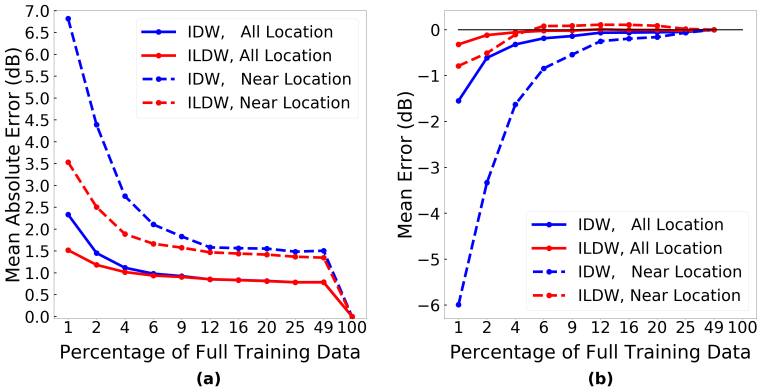


FIGURE 2.9: Estimation errors for interpolation schemes for varying training data

observe the impact on the performance of various algorithms. See Figure 2.8, where the number of sensors is varied from 80 to 400. We see that all algorithms perform better with increasing number of sensors as expected, with MAP* performance improving significantly (in both L_{err} as well as $f_r + m_r$) as number of sensors is increased from 80 to 160. More importantly, except for very low number of sensors (i.e., 80), MAP* handily outperforms the other two algorithms.

Varying Training Cost. Finally, we now investigate how the training cost (i.e., number of PDs constructed from raw observations) affects the performance of our MAP* algorithm. Note that the other algorithms do not depend on the training data, hence not shown. We first evaluate the interpolation error of our ILDW scheme for varying training cost (number of known PDs) by comparing with the traditional IDW scheme on which it is based. See Figure 2.9, which plots the mean absolute error (MAE) as well as mean error (ME). As the interpolation error is substantially higher for points that are closer to the transmitter, we plot MAE and ME as averaged over all interpolated points as well as over just the points close (less than 800m away) to the transmitter. Note that the PDs at sensor locations closer to the transmitter would have a stronger bearing on the localization accuracy, and thus, the MAE and ME values for points closer to the transmitter are of more significance. We observe here that as expected both MAE and (absolute value of) ME decrease with increase in the training cost for both IDW and ILDW, but MAE and ME of ILDW is significantly lower than that of IDW especially for low percentages of training cost and when the points are close to the transmitter.

We now plot the performance of MAP* for varying training data; see Figure 2.10. As expected, the performance metrics show general improvement with increase in amount of training. More importantly, we note that with 5-10% of training, MAP* achieves performance comparable to that with 100% training, suggesting that our interpolation scheme is largely effective as long as 5-10% of PDs are constructed from raw observations.

In Presence of Authorized Users (MAP).** We now evaluate the performance of our MAP** approach which is tailored to work in the presence of authorized users. To evaluate MAP**, we place 5 authorized users in the area—with 2 primary and 3 secondary users. The primary users are placed at fixed locations, while the secondaries are put at random locations. We assign each authorized user a random power in the range of 30 to 32dBm, while, as before, a random power between 28 and 32dBm to the intruders. To ensure that these 5 authorized users do not "interfere" with each other, we ensure that the distance between any two of these authorized users is at least 1000m. We compare MAP** with the simpler approach called MAP*+ that uses MAP* to localize all transmitters

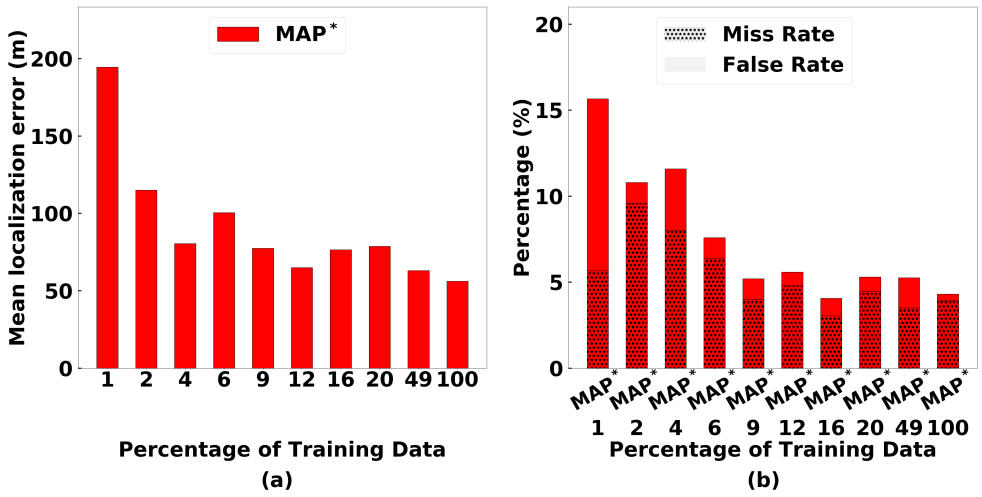


FIGURE 2.10: Localization performance of MAP* in a large scale area, for varying training data

(authorized as well as intruders) and then removes the predicted transmitters that are closest to the authorized users. See Figure 2.11, which shows that MAP** easily outperforms MAP*+ for varying number of intruders.

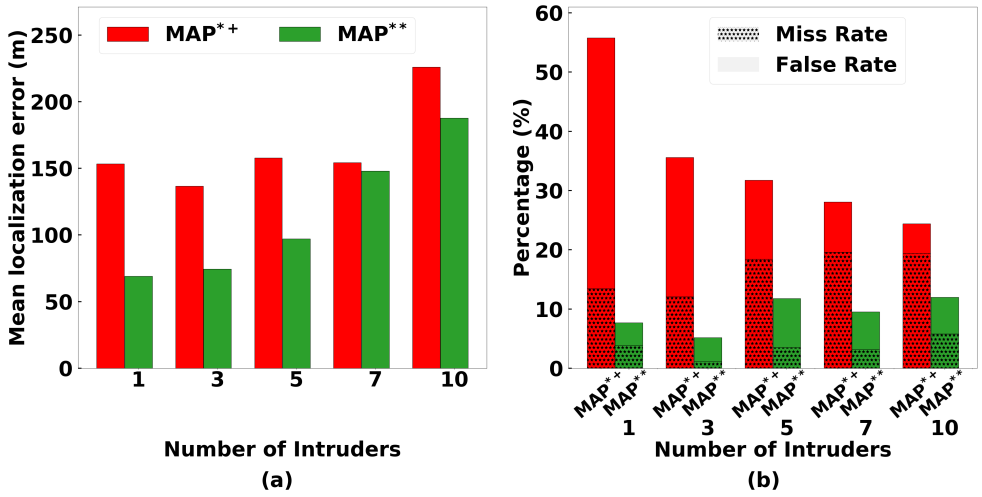
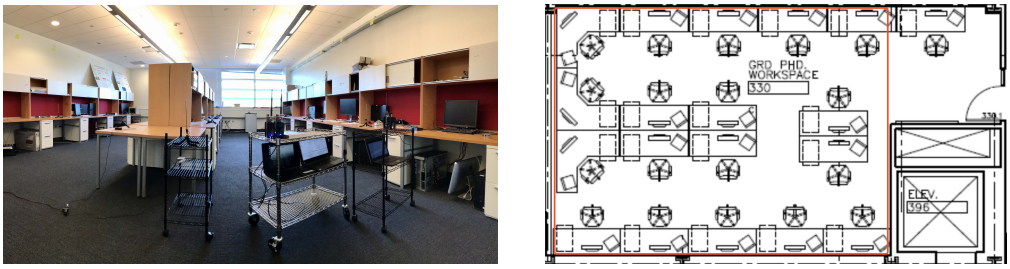


FIGURE 2.11: Localization performance of MAP*+ and MAP** in large-scale simulations with authorized users present, for varying number of intruders

2.6 Testbed Implementation

In this section, we implement our techniques over commodity devices and evaluate them over two small-scale testbeds—one indoor and one outdoor. Outdoor environment is a realistic setting for our target application of shared spectrum systems, while the indoor environment provides more challenging signal attenuation characteristics due to walls and other obstacles.

Sensor and Transmitters Used. Our low-cost (sub $100, see [34] for a measurement study of low-cost spectrum sensors) sensing device is composed of a single-board computer Odroid-C2 with an RTL-SDR dongle which connects to a dipole antenna. We



(A) Indoor lab environment (B) Floor plan

FIGURE 2.12: Indoor testbed. (a) Our lab used for the indoor testbed, (b) The lab's floor plan.

(A) Outdoor parking lot environment] (B) Satellite

FIGURE 2.13: Outdoor testbed. (a) Parking lot picture, (b) Satellite image of the parking lot; the red box is the area of the experiment, and the stars are the locations of sensing devices during evaluation.

deploy 18 of these sensing devices in our indoor and outdoor testbeds, and configure them for low gain. For transmitters/intruders, we use USRP B210 and HackRF devices powered by laptops; we place these on a cart for mobility. These transmiter devices are uncalibrated, and there is no way to assign a specific transmit power. However, they have a configurable parameter called *gain* which is almost perfectly correlated to power when the gain is in a specific range, i.e., when the transmitter's gain is increased by 1, the receiver's signal strength increases by 1dB. We thus use the gain parameter to adjust transmit power in the USRP devices. For indoor experiments, the location is manually derived, while for outdoor experiments, we use GPS dongles connected to the laptops. For collecting sensor observations, we implemented a Python repository in Linux that measures spectrum in real time at 915MHz ISM band and 2.4Msps sample rate. The repository collects I/Q samples fetched from the RTL-SDR dongle and computes the RSS value, then record the RSS along with timestamp and location. These three pieces of information are sent to a server that runs the localization algorithms.

Testbeds. The **indoor** testbed is built in a lab of our Computer Science building. Figure 2.12 depicts the lab with its floor plan. The red box in the floor plan is the area where experiments are conducted. The area is 9.6 $\times$ 7.2 m^2 (or 2177 square feet) large, with four rows of desks. The middle two rows are separated by a wooden board. The area is imagined to be divided into 48 grid cells each of size 1.2m $\times$ 1.2m, with the help of ceiling tiles each of which is 0.6m $\times$ 0.6 m. The **outdoor** testbed is over an open space parking lot. See Figure 2.13. The area is 32m $\times$ 32m. We divide the area into 100 grid cells with each cell representing an area of 3.2m $\times$ 3.2m. The GPS device returns

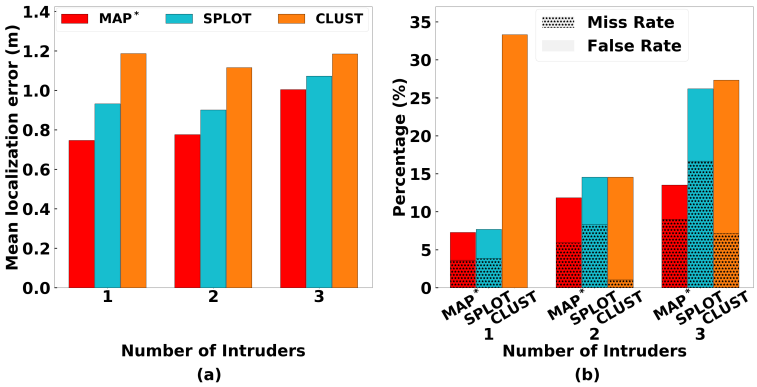


FIGURE 2.14: Localization performance of varies algorithms in an indoor testbed

location in (latitude, longitude) and the program converts it into coordinates. We use an outdoor WiFi router and long power cords for network and electrical connection respectively. During the evaluation, the 18 sensing devices are placed on the ground and are randomly spread out.

Training. In both the testbeds, for training (i.e., constructing non-interpolated PDs), we first pick 18 random grid cells and place sensors in their approximate centers. Then, we manually move the transmitter around in a cart through each of the grid cells. For the USRP transmitter, we use a gain value of 45 in the indoor environment and 58 in the outdoor testbed. We use a higher gain for outdoors to allow the transmitter to have a larger transmission range in a larger area. With each grid cell, the transmitter transmits from 3 to 4 different points within each grid cell, and for each such location of the transmitter, the sensors (at the 18 picked locations) gather tens of signal strength readings. From these readings, we construct a Gaussian probability distribution from each grid cell location of the transmitter. More specifically, for a particular grid cell location of the transmitter, we average over the readings from multiple TX positions within that particular grid cell—this process of averaging different positions of the TX inside a grid cell makes the Gaussian distributions more robust to multipath fading and shadowing. The overall training process takes an hour for indoors, and about two and a half hours for outdoors.

Evaluation. For evaluation, in both testbeds, we place the 18 sensors at centers of grid cells that are randomly chosen and are different from the cells chosen for training above. The chosen locations for the outdoor tested are shown in Fig. 2.13(b). We choose the intruder's gain/power to be in the range of $[p^* - 1, p^* + 1]$, where p^* is the gain/power used during the training phase as mentioned above. Roughly half of our experiments involve close-by (in the same or adjacent grid cells) intruders. Localization is done on a laptop which listens to HTTP requests containing the sensors' observations.

2.6.1 Results

Localization Metrics. Figure 2.14-2.15 show the localization results for the indoor and outdoor testbeds respectively. Overall, the results indicate that `MAP`* performs the best across all metrics, with the overall performance gap between `MAP`* and `SPLOT` increasing with the increase in number of intruders. When the number of intruders is 3, the performance of `SPLOT` is significantly worse than `MAP`* due to a significantly higher (84% for indoors and 53% for outdoors) sum of miss and false-alarm rates and

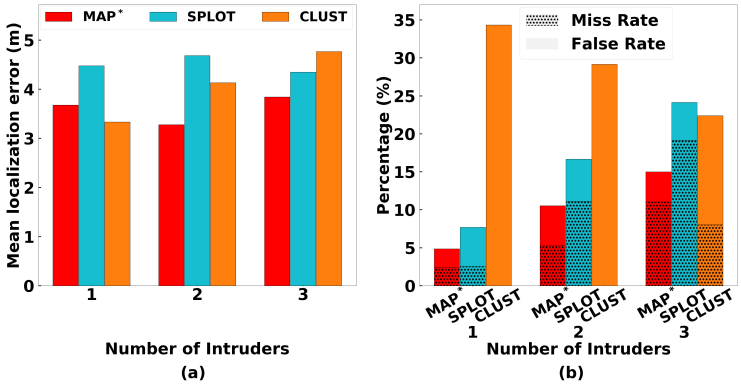


FIGURE 2.15: Localization performance of varies algorithms in an outdoor testbed

43% higher localization error. The `CLUS` algorithm generally performs the worst, but its performance doesn't have a strong correlation with the increase in the number of intruders; recall that `CLUS` is given the range of number of intruders as an extra piece of information compared to the other algorithms. In terms of absolute performance, we see that the localization error of `MAP`* is roughly around 1 or less grid cell, and the sum of miss-rate and false-alarm is between 5-15%.

TABLE 2.4: Interpolation Mean Absolute Error (MAE) and Mean Error (ME) in dB for `IDW` and `ILDW`

Environment	`IDW` (MAE)	`ILDW` (MAE)	`IDW` (ME)	`ILDW` (ME)
Indoor	2.6	1.7	1.7	0.25
Outdoor	6.2	2.7	5.8	0.48

Interpolation Error. Table 2.4 show the interpolation mean absolute error (MEA) as well as mean error (ME) of `IDW` and `ILDW` when the transmitter and receiver are close by (i.e., within a distance of 3 grid cells). When the transmitter and receiver are far away, the difference of `IDW` and `ILDW` is small and thus not shown. We see that when compared with `IDW`, our `ILDW` interpolation scheme decreased the mean absolute error by 35 percent in the indoor environment and 56 percent in the outdoor environment. In terms of mean error, `ILDW` reduced the error compared to `IDW` by as large as 86 percent and 92 percent respectively. This is because `IDW` mostly tends to estimate the value to be larger than the ground truth, while `ILDW`'s estimates are more even across the ground truth.

TABLE 2.5: Power Prediction Mean Absolute Error (MAE) and Mean Error (ME) in dB for indoor and outdoor testbed

# Intruder	Indoor (MAE)	Outdoor (MAE)	Indoor (ME)	Outdoor (ME)
1	0.34	0.50	-0.02	0.02
2	0.57	0.63	0.10	0.54
3	0.77	0.90	0.49	0.76

Intruder Power. Table 2.5 show the errors in the predicted powers of the intruders

in MAP*. We see that the outdoors have a slightly higher power prediction error, likely because of a larger number of grid cells. We also note that with the increase in the number of intruders, the error in predicted power increases.

2.7 Conclusion

In this chapter, we have developed an efficient Bayesian approach with a noval interpolation scheme to localize multiple transmitters in presence of authorized users, and demonstrate its superior power over large-scale simulations and smaller scale indoor and outdoor testbeds. In our future work, we wish to extend our techniques to allow a continuous location domain and design methods to further minimize training cost. In addition, we will consider alternate signal measurements such as angle-of-arrival (AoA).

Acknowledgments

The work is supported by NSF grants CNS-1642965 and CNS-1815306. Thanks to Salman Qavi and Jayesh Ranjan for their help during the testbed. Also thanks to Samir Das, Petar Djuric and Peter Milder for providing advice during group meetings.

Chapter 3

DeepMTL: Deep Learning Based Multiple Transmitter Localization and Power Estimation

In this chapter, we address the problem of Multiple Transmitter Localization (`MTL`). `MTL` is to determine the locations of potential multiple transmitters in a field, based on readings from a distributed set of sensors. In contrast to the widely studied single transmitter localization problem, the `MTL` problem has only been studied recently in a few works. `MTL` is of great significance in many applications wherein intruders may be present. E.g., in shared spectrum systems, detection of unauthorized transmitters and estimating their power are imperative to efficient utilization of the shared spectrum.

In this chapter, we present `DeepMTL`, a novel deep learning approach to address the `MTL` problem. In particular, we frame `MTL` as a sequence of two steps, each of which is a computer vision problem: image-to-image translation and object detection. The first step of image-to-image translation essentially maps an input image representing sensor readings to an image representing the distribution of transmitter locations, and the second object detection step derives precise locations of transmitters from the image of transmitter distributions. For the first step, we design our learning model `sen2peak`, while for the second step, we customize a state-of-the-art object detection model `YOLOv3-cust`. Using `DeepMTL` as a building block, we also develop techniques to estimate transmit power of the localized transmitters. We demonstrate the effectiveness of our approach via extensive large-scale simulations and show that our approach outperforms the previous approaches significantly (by 50% or more) in performance metrics including localization error, miss rate, and false alarm rate. Our method also incurs a very small latency. We evaluate our techniques over a small-scale area with real testbed data and the testbed results align with the simulation results.

3.1 Introduction

The RF spectrum is a limited natural resource in great demand due to the unabated increase in mobile (and hence, wireless) data consumption [4, 83]. In 2020, the U.S. FCC moves to free up 100 MHz of previously military occupied mid-band spectrum in the 3.45-3.55 GHz band for paving the way for 5G development. Also, the research and industry communities have been addressing this capacity crunch via the development of *shared spectrum*. Spectrum sharing is the simultaneous usage of a specific frequency band in a specific geographical area and time by a number of independent entities where harmful electromagnetic interference is mitigated through agreement (i.e., policy, protocol) [46]. Spectrum sharing techniques are also normally used in 5G networks to

enhance spectrum efficiency [68]. However, protection of spectrum from unauthorized users is important in maximizing spectrum utilization.

The increasing affordability of the software-defined radio (SDR) technologies makes the shared spectrum particularly prone to unauthorized usage or security attacks. With easy access to SDR devices (e.g. HackRF, USRP), it is easy for selfish users to transmit data on a shared spectrum without any authorization and potentially causing harmful interference to the incumbent users. Such illegal spectrum usage could also happen as a result of infiltration of computer viruses or malware on SDR devices. [68] depicts three cases of spectrum attack. As the fundamental objective behind such shared spectrum paradigms is to maximize spectrum utilization, the viability of such systems depends on the ability to effectively guard the shared spectrum against unauthorized usage. The current mechanisms however to locate such unauthorized users (intruders) are human-intensive and time-consuming, involving the FCC enforcement bureau which detects violations via complaints and manual investigation [63]. Motivated by the above, we seek an effective technique that is able to accurately localize multiple simultaneous intruders (transmitters). Below, we describe the multiple transmitter localization problem.

Multiple Transmitter Localization (`MTL`). The transmitter localization problem has been well studied, but most of the focus has been on localizing a *single* transmitter at a time. However, it is important to localize multiple transmitters simultaneously to effectively guard a shared spectrum system. E.g., a malware or virus-based attachment could simultaneously cause many devices to violate spectrum allocation rules; spectrum jamming attacks would typically involve multiple transmitters. More importantly, a technique limited by the localization of a single intruder could then be easily circumvented by an offender by using multiple devices. The key challenge in solving the multiple transmitter localization (`MTL`) problem comes from the fact that the deployed sensor would receive only a *sum* of the signals from multiple transmitters, and separating the signals may be impossible.

Prior Works. The `MTL` problem has been recently addressed in a few prior works, among which `SPLOT` [63], `MAP` [136], and `DeepTxFinder` [140] are the most prominent. `SPLOT` essentially decomposes the `MTL` problem to multiple single-transmitter localization problems based on the sensors with the highest power readings in a neighborhood. However, their technique implicitly assumes a propagation model, and thus, may not work effectively in areas with complex propagation characteristics, and it is not effective in the case of transmitters being located close by (a key challenging scenario for `MTL` problem). Our recent work `MAP` solves the `MTL` problem using a hypothesis-driven Bayesian approach; in particular, it uses prior training in the form of distributions of sensor readings for various transmitter locations, and uses the training data to determine the most likely configuration (i.e., transmitters' locations and powers) for a given vector of sensor readings. However, to circumvent the high computational cost of a pure Bayesian approach, `MAP` uses a divide and conquer heuristic which results in somewhat high number of misses and false alarms while still incurring high latency. `DeepTxFinder` uses a CNN-based learning model approach; however, they use a separate CNN model for a specific number of transmitters and thus may incur high model complexity and training cost while also limiting the number of transmitters that can be localized. In our evaluations, we compare our work with each of the above approaches.

`DeepMTL`: Our Two-Step Approach. As in prior works [63, 24], we assume a crowd-sourced sensing architecture (See Fig. 3.1) wherein relatively low-cost spectrum sensors are available for gathering signal strength in the form of received power. We use a

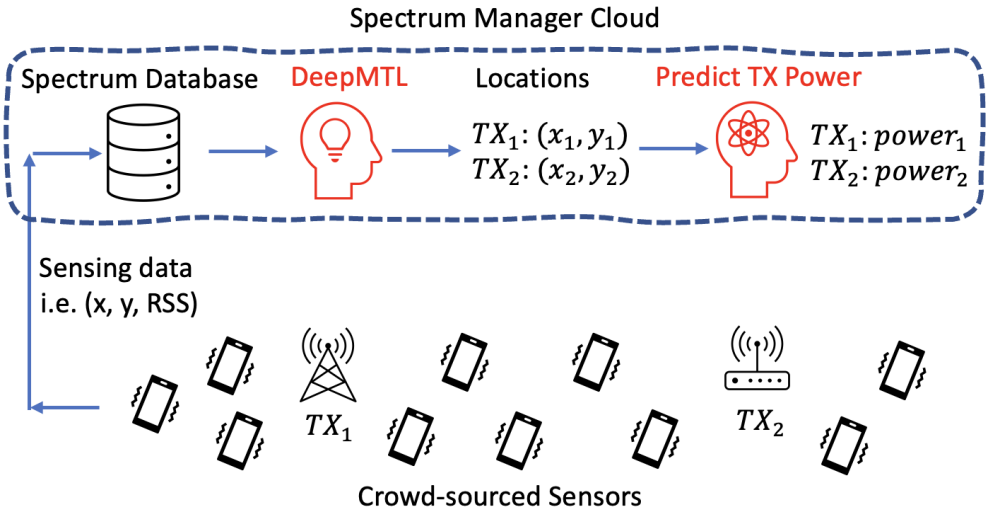


FIGURE 3.1: Multiple transmitter localization using a distributed set of sensors. Sensing data is uploaded to a spectrum manager server in the cloud. `DeepMTL` is a deep learning approach to multiple transmitter localization which helps protect spectrum against unauthorized usage. After that, prediction of transmission powers happens using `DeepMTL` as a building block.

convolutional neural network (CNN) based approach to solve the `MTL` problem. In particular, we frame `MTL` as a sequence of two steps: image-to-image translation and object detection, each of which is solved using a trained CNN model. The first step of image-to-image translation maps an input image representing sensor readings to an image representing the distribution of transmitter locations, and the second object detection step derives precise locations of transmitters from the image of transmitter distributions. We name our `MTL` approach as `DeepMTL`.

Motivation. Our overall approach and its various aspects are motivated by the following considerations. **First**, we use a learning-based strategy to preclude assuming a propagation model [63] or conducting surveys of sensors reading distributions [136]. Assumption of propagation model suffers from the fact that even sophisticated propagation models yield unsatisfactory accuracy and thus lead to degraded performance. Among all learning-based strategies, deep learning can implicitly capture the environment characteristics (e.g., objects, walls, landscape) in the neural network layers' weights learned through the training of the data [7]. Even though a learning-based approach incurs a one-time high training cost, it generally incurs minimal latency during inference, which is an important consideration for our `MTL` problem. The intruder detection should incur minimal latency to be effective. **Second**, the geographical nature of the `MTL` problem suggests that convolutional neural networks (CNNs) are well-suited for efficient learning of the desired function. In particular, the features of the `MTL` problem can be represented in an image (2D matrix) corresponding to their geographic locations, which can be fed as an input to an appropriate CNN model which can leverage the spatial correlation among the input features to facilitate efficient learning. **Lastly,** we use a two-step architecture to facilitate efficient training by essentially providing an additional intermediate image. In particular, we are able to map each step to well-studied standard computer vision problems, allowing us to build upon known techniques.

Overall Contributions. The goal of our work is to develop an efficient technique for accurate localization of simultaneously present multiple transmitters/intruders. We

also extend our technique to address various extensions such as power estimation and the presence of authorized users. Overall, we make the following contributions.

1. For the `MTL` problem, we develop a novel two-step CNN-based approach called `DeepMTL` approach. For the first step of image-to-image translation, we develop a CNN model that translates an image representing the sensor readings into an intermediate image that encodes distributions of transmitter locations (Section 3.3). For the second step of mapping transmitter distributions to precision locations via object detection, we customize the well-known object detection method YOLOv3 (Section 3.4).

2. For localization of transmitters in presence of authorized users, we augment the `DeepMTL` model by adding a pre-processing step based on a CNN-model that first reduces the sensor readings by the power received from the authorized users (Section 3.5).

3. To estimate transmit power of the intruders, we augment our `DeepMTL` model with a power-estimation CNN-model which iteratively estimates the power of transmitters in sub-areas (Section 3.6).

4. We evaluate our techniques via large-scale simulations as well as a small-scale testbed data and demonstrate their effectiveness and superior performance compared to the prior works (Section 3.7).

A preliminary version of this paper appeared at IEEE WoWMoM 2021 [134].

3.2 Background, `MTL` Problem and Our Approach

In this section, we describe the background of the shared spectrum systems, formulate the `MTL` problem, then describe our methodology.

Shared Spectrum System. In a shared spectrum paradigm, the spectrum is shared among licensed users (primary users, PUs) and unlicensed users (secondary users, SUs) in such a way that the transmission from secondaries does not interfere with that of the primaries (or secondaries from a higher-tier, in case of a multi-tier shared spectrum system). In some shared spectrum systems, the location and transmit power of the primary users may be unavailable, as is the case with military or navy radars in the CBRS band. Such sharing of spectrum is generally orchestrated by a centralized entity called *spectrum manager*, such as a spectrum database in TV white space [64] or a central spectrum access system in the CBRS 3.5GHz shared band [56]. The spectrum manager allocates spectrum to requesting secondaries (i.e., permission to transmit up to a certain transmit power at their location) appropriately so as to avoid interference to primaries. Users that transmit without explicit permission are referred to as unauthorized users or *intruders*; the `MTL` problem is to essentially localize such intruders.

`MTL` Problem. Consider a geographic area with a shared spectrum. Without loss of generality, we assume a single wireless frequency[1] throughout this paper[2]. For localization of intruders, we assume available crowdsourced sensors that can observe received

[1]To avoid confusion with image channels, we use *wireless frequency* instead of the perhaps more appropriate *wireless channel* term.

[2]Multiple wireless frequencies can be handled independently. Note that if we assume the wireless propagation characteristics to be similar for different frequencies, then we do not need to train different models for each of them. Our localization techniques would still work for scenarios wherein the intruders may change their transmit frequencies dynamically.

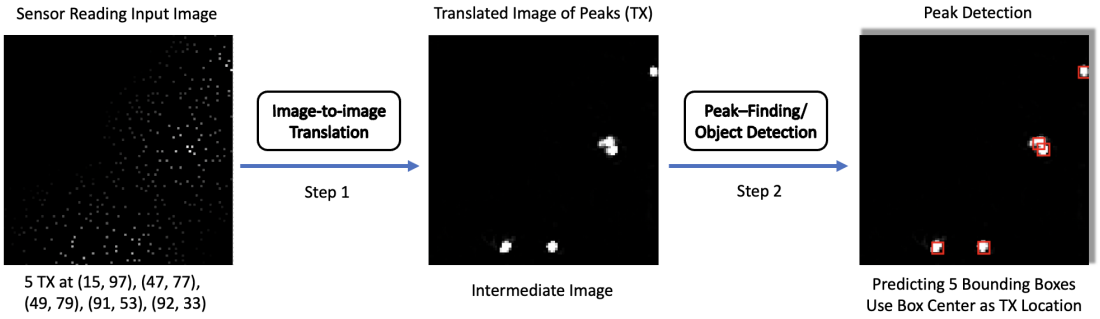


FIGURE 3.2: The overall two-step CNN architecture of the `DeepMTL` model. The first step is the `sen2peak`, whose higher idea is to translate the input image of sensor readings to the image of peaks where each peak implies a transmitter. The `sen2peak` architecture is illustrated in Fig. 3.4. The second step is `YOLOv3-cust`, a customized version of YOLOv3, to perform object/peak detection in the output image of the first step. This step returns the precise location coordinates of TX. The `YOLOv3-cust` architecture is illustrated in Fig. 3.5. A zoom-in of the peak detection result of the second step is in Fig. 3.6.

signal in the wireless frequency of interest, and compute (total) received signal strength (RSS). RSS can be measured using low-cost sensors and has been shown to achieve good accuracy for single-transmitter localization [10]. In the related work Section 3.8, we will discuss signal metrics other than RSS, such as AoA, ToA, etc. At any instant, there may be a set of intruders present in the area with each intruder at a certain location transmitting with a certain power which may be different for different intruders.

The `MTL` problem is to determine the set of intruders with their locations at each instant of time, based on the set of sensor observations at that instant. For the main `MTL` problem, we assume that there are no primary or authorized users, and thus, assume that the sensor readings represent aggregate received power from the transmitters we wish to localize. However, in Section 3.5, we investigate the more general `MTL` problem where the background primary and/or secondary users may also be present.

Our Approach. In our context, each sensor communicates its observation to a centralized spectrum manager which then runs localization algorithms to localize any potential (multiple) transmitters. We design and implement a novel two-step localization algorithm named `DeepMTL`, as illustrated in Fig. 3.2, based on CNN models. The first step (Section 3.3) is a four-layer image-to-image translation CNN model that is trained to translate an input image representing sensor readings to an image of transmitters' locations distributions. Each distribution of a transmitter can be visualized as a mountain with a peak, so we name this model `sen2peak`. The second step (Section 3.4), called `YOLOv3-cust`, is a customized object-detection method build upon YOLOv3[96] which localize the objects/peaks in the translated image. The high-level motivation behind our two-step design is to frame the overall `MTL` problem in terms of well-studied learning problem(s). The two steps facilitate efficient learning of the models by supplying an intermediate image with the training samples.

3.3 `DeepMTL` Step 1: Sensor Readings to TX Location Distributions

In this section, we present the first step of our overall approach to the `MTL` problem, i.e., the image-to-image translation step which translates/transforms the sensor reading

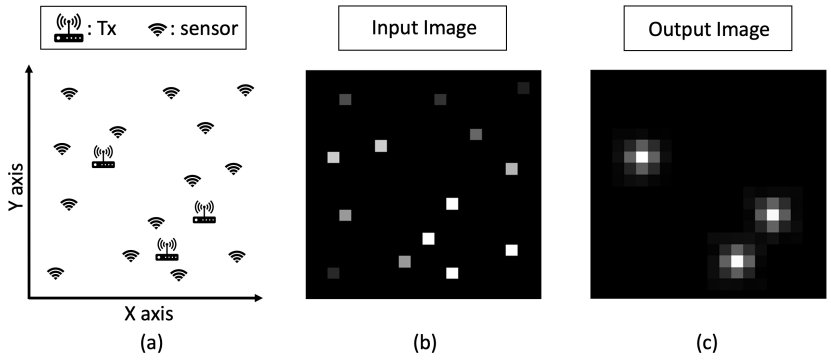


FIGURE 3.3: Illustration of DeepMTL first step's input and output images. (a) Area with distributed sensors and transmitters to be localized. (b) Input image representing the sensor readings (RSS) and locations. (c) Output Image, where we put a 2D Gaussian distribution with its "peak" at the transmitter's location.

to distributions of TX locations. Here, we first create a grayscale image to represent the input sensor readings; this image encodes both the sensors' RSS readings and the sensors' physical location. We then train and use a convolutional neural network (CNN) model to transform this input image to an output image which represents the distribution of TX locations. Pixels in the output image that have higher values will have a higher chance of having a TX being present at that location.

Input/Output Image Sizes and Tiling Approach for Large Areas. We need to represent data by images of certain sizes. Typically, an image should be a size of a few hundred pixels by a few hundred pixels, since a thousand pixels by thousand pixels images will consume too much GPU memory. In this chapter, we pick 100×100 as the size for both our input and output images in the first image-to-image translation step. Given an area that we want to monitor and a 100×100 size image, we will know how large an area a pixel will represent and we call it a pixel subarea. A large pixel subarea could certainly lead to high localization errors, due to very coarse granularity. We can address this by using a "tiling" technique, wherein we divide the given area into tiles, then represent each tile by 100×100 size image and use our localization techniques in the tile. We can do some post-processing to handle cross-tiling issues (e.g., [140] uses overlapping tiles and employs a voting scheme inside the overlapping tile area).

3.3.1 Input Image Representing Sensors' Readings

We localize transmitters based on observations from a set of sensors, i.e. solve the MTL problem assuming only intruders. The input of the localization method is sensor observations. Here, an *observation* at a sensor is the received power (RSS, in decibels) over a time window of a certain duration, in the frequency of interest (we assume only one wireless frequency). RSS is computed using FFT over the I/Q samples collected in a time window. More specifically, in our evaluations, we use a Python API [59] that computes the power spectral density from a sequence of signal data (I/Q samples), and then, we choose the RSS at the frequency of interest. Different than [63, 136], we represent the sensor information, i.e., their locations and observations, in a 2D input image. We use a 2D grayscale image, and let us denote it $\mathbf{X}$. The pixel $\mathbf{X_{i,j}}$ denotes the observation of the sensor at the grid cell whose index is (i, j). For example, $\mathbf{X}_{10,20} = -50$ denotes there is a sensor at coordinate $(10, 20)$ with an RSS reading of -50 dB. If there is no sensor at location (i, j), we assign the noise floor $\mathcal{N}$ (i.e. -80 dB) value to $\mathbf{X_{i,j}}$. Note

that the above pixel values (representing the sensor observations) are not the standard image pixel values that lie in the [0, 255] range. Also, since the pathloss computed by propagation models during simulations could be real numbers, the sensor observation values could be real numbers. So we use a 2D matrix with real numbers instead of an image object.

Before passing this sensor reading image as input to our CNN model, we do a normalization step; we first subtract the $\mathcal{N}$ from each value and then divide it by $-\mathcal{N}/2$. Let $\mathbf{X}^{'}$ denote the 2D matrix after the normalization of $\mathbf{X}$. The value $\mathbf{X}^{'}_{\mathbf{i,j}}$ will be zero at locations without sensors, and $\mathbf{X}^{'}_{\mathbf{i,j}}$ will be a positive real number (in most cases, less than two) for locations with sensors. E.g., if $\mathbf{X}_{10,20} = -50$, then the $\mathbf{X}^{'}{}_{10,20}$ equals to $(-50-(-80))/40) = 0.75$. Fig. 3.3 (b) shows how a matrix is used to represent the input information that contains both the RSS and the spatial location of the distributed sensors in an area that exists 14 sensors in Fig. 3.3(a).

3.3.2 Output Image Representing TX locations' Distributions

We now focus on designing the output image to represent the distribution of TX locations; the output image is essentially the "label" assigned to each input image that guides the training of the CNN model. Fig. 3.3(c) illustrates the output image of the image-to-image translation step in Fig. 3.3(a) that contains three transmitters.

A straightforward representation that represents the TXs with locations is to just use an array of (x, y) elements where each (x, y) element is the location of a transmitter, as in [140]. However, this simple representation is less conducive to efficient model learning, as the representation moves away from spatial representation (by representing locations as positions in the image) to direct representation of locations by coordinate values. E.g., in [140]'s CNN-based approach to `MTL` problem, the authors assume a maximum number N of transmitters and train as many as $N + 2$ different CNN models and thus, limiting the overall solution to the pre-defined maximum number of transmitters. Instead, in our approach, we facilitate the learning of the overall model, by solving the `MTL` problem in two steps, and in this step of translating sensors' reading to transmitter locations' distributions, we represent the output also as an image. This approach allows us to use a spatial learning model (e.g. CNN) for the second step too, and preclude use of regression or fully-connected layers in the first step.

Inspired by recent work on wireless localization problem [7] that represents the input and output as images, we represent our output of the first step as an image as well. The output image is a grayscale image implemented as a 2D matrix with real numbers. In the output image, we use 25 (5×5) pixel values to represent the presence of a transmitter. It is desirable to use an odd side length square (e.g., 3×3, 5×5, 7×7) for symmetry. For a 100×100 size input we use, while 3×3 gives too little information for a transmitter and 7×7 generates too many overlaps for close by transmitters, 5×5 is the sweet spot. Other pixels far away from any transmitter are zero-valued. Among multiple potential ways to represent a transmitter presence by a number of pixels, we found that using a 2D Gaussian distribution around the pixel of TX location, as shown in Fig. 3.3(c), yields the best model performance. Thus, a geographic area with multiple transmitters present is represented by a grayscale image with multiple Gaussian distributions, with each Gaussian distribution's peak inside the pixel corresponding to transmitter's location. Based on preliminary performance tests, we pick the amplitude of the 2D Gaussian peak to 10, the standard deviation to 0.9, and located the center of

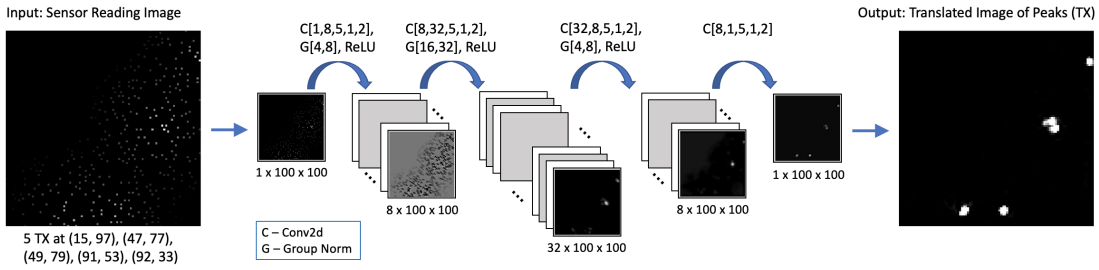


FIGURE 3.4: Architecture of the first step CNN, a four layer image-to-image translation model (`sen2peak`). The figure displays how the data volume flows through the various convolutional layers. C stands for Conv2d, and for each Conv2d layer, the five values shown are [number of input channels, number of output channels, kernel size, stride, padding]. G stands for group normalization, and, for each group normalization, the two values shown are [number of groups, number of channels]. See §3.3 for details.

the distribution at the location of each transmitter. Note that the location of the TX is in continuous domain and usually not at the center of the grid cell.

3.3.3 Image-to-Image Translation: `sen2peak` CNN Model

At a higher level, we use a deep and spatial neural network, in particular a CNN, to learn the approximation function that maps the input image (of sensor readings) to the output image (of Gaussian distributions for TX locations). We refer to this as the *image-to-image translation* model. Our approach is inspired by the recent work [7] that frames a different wireless localization problem as an image-to-image translation problem. We incorporate the idea into our multiple transmitter localization problem and utilize recent advances in the computer vision area. Encoder-decoder based CNN models like U-Net [101] with down-sampling and up-sampling convolutional layers have been successful in effectively learning image-to-image translation functions. However, in our setting, we observe that the usage of down-sampling layers (such as max-pooling) degrades the performance of the model, especially in the case when transmitters may be close to each other wherein the model is unable to distinguish the nearby transmitters and generate a single large distribution in the output image. To circumvent this, we avoid using any down-sampling layers in our model and redesign the image-to-image translation model as described below.

`sen2peak` CNN Model. We refer to our image-to-image translation CNN model as `sen2peak`, as it translates sensors' readings to "peaks" with Gaussian distributions corresponding to transmitter locations. It has four [3] convolutional layers, as shown in Fig. 3.2(a). We use an input size of 100×100. The number of convolutional filters are varying for different layers, with up to 32 in one of the layers. We tried doubling the filter numbers at each layer, but it does not lead to significant improvement (it does yield a lower error, but the output image does not improve significantly to impact the second step of our architecture). We use a kernel size of 5×5, a stride of 1, and a padding of 2. This ensures that the dimensions do not decrease and all the pixels are treated uniformly, including the ones at the edge of the image. With the above four convolutional layers, the receptive field [78] of each neuron in the output layer is 17×17.

[3] We observe that a four-layer lightweight and symmetric `sen2peak` model produces good results and adding more layers gives marginal improvement.

Normalization layers can improve the learning process. We chose group normalization [125] and put it after the first three convolutional layers. We compared group and batch normalization [60] methods in our context, and observed better performance with the group normalization. For the activation layers, we select rectified linear unit (ReLU) and put it after the group normalization layers.

The Loss Function. Our inputs (X) and output (Y) are images. We use L2 loss function which computes the mean squared error aggregated over individual pixels. More formally, our loss function is defined as:

$$\frac{1}{N}\sum_{i}^{N}||\texttt{sen2peak}(X_i) - Y_i||^2 \tag{3.1}$$

where N is the number of samples used in computing the loss, $||\cdot||^2$ is L2 loss function, X_i and Y_i are the i_{th} sample's input and output images respectively, and $\texttt{sen2peak}(X_i)$ is the predicted output image corresponding to the input X_i. During training, we use Adam [66] as the optimizer that minimizes the loss function. We set the learning rate to 0.001 and the number of epochs to 20 and the model converges well.

3.4 `DeepMTL` Step 2: TX Locations' Distributions to Precise Locations

In this section, we present the second step of our overall localization approach. We refer to this step as the *peak detection* step, as the goal is to detect the peaks within the Gaussian distributions in the input image (which is also the output image of the first step). The first step outputs an image that has multiple distributions (presumably, Gaussian), whose peaks need to be interpreted as precise locations of the transmitters/intruders. As, our end goal is to determine the precise locations of the present transmitters, we develop techniques to detect peaks within the output image of the first step. We propose two different strategies for the peak-detection task. The first strategy is a straightforward peak detection algorithm based on finding local maximal values, while the second strategy is based on framing the problem as an object detection task; for the second strategy, we utilize a widely used state-of-the-art computer vision model called YOLOv3 [96].

Simple Peak Detection Method. The simple and straightforward peak detection method is to designate pixels with locally maximal values as peaks, subject to certain thresholds. More formally, we use a threshold x for a peak value, and also use a parameter r to define a r-radius neighborhood of a pixel. Then, any pixel whose value is more than x and is the maximum among all pixels with a r-radius neighborhood, is designated as a peak (transmitter location). We use $x = 2$ and $r = 3$, in our evaluations. Note that each pixel represents a subarea; thus, a pixel designated as pixel only implies the transmitter location at the *center* of the corresponding subarea. To localize the transmission more precisely with the pixel's subarea, we use a scheme that localizes the transmitter within the subarea by computing a weighted average of the peak pixel's coordinate and the peak's neighbor pixels' coordinates. The weight of a pixel is the predicted pixel value itself from the first step `sen2peak`. We refer to the above simple approach for the second-step of `DeepMTL` as `simplePeak`.

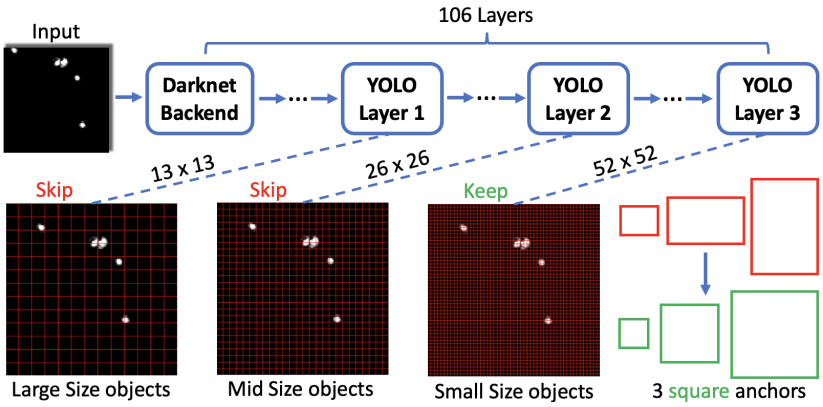


FIGURE 3.5: Our `YOLOv3-cust` in the second step of the `DeepMTL`. The two major customization are: (i) Use only the third YOLO layer that detects small size objects (the output of `YOLOv3-cust` is the bounding box predicted by the third YOLO layer and we use the center of the bounding box as the transmitter location), and (ii) change the rectangle anchors to square anchors.

3.4.1 Object-Detection Based Precise Localization: `YOLOv3-cust`

The simple hand-crafted method described in the previous subsection performs reasonable well in most cases in our simulations. However, its key drawback is that it needs appropriate threshold values that may vary from case to case; such thresholds can be difficult to determine, especially since the input images (with distributions) are not expected to be perfect as they are themselves output of a learning model. Inaccurate threshold values can lead to false alarms and misses. Also, the previous method is not sufficiently accurate at the sub-pixel level, where each pixel may represent a large area such as $10m \times 10m$ or even $100m \times 100m$. Thus, we propose a CNN-based learning method that overcomes the above shortcomings. CNN has been widely used for object detection in different areas [75, 3].

We frame this problem as an object detection task where the objective is to detect and localize known objects in a given image. We observe that our second-step peak detection problem is essentially an object detection problem where the "object" to detect is a "peak". Thus, we turn the `MTL` problem of localizing multiple transmitters into detecting peaks in the images output by `sen2peak` model. For object/peak detection, we design `YOLOv3-cust`, our customized version of YOLOv3 [96]. Fig. 3.6 is a zoom-in of localizing two close by transmitters (peaks) in Fig. 3.2(b).

Peak Detection Using `YOLOv3-cust`. Object detectors are usually comprised of two parts: (i) a backbone which is usually pre-trained on ImageNet, and (ii) a front part (head), which is used to predict bounding boxes of objects, probability of an object present, and the object class. For the front part, object detectors are usually classified into two categories, i.e., one-stage detectors such as the YOLO [97] series, and two-stage detectors such as the R-CNN [52] series. We choose the one-stage YOLO series because of its computational efficiency, high popularity and available ways to customize it for our specific context. We refer to the customized version as `YOLOv3-cust`, see Fig. 3.5. Implementing a 106-layer deep neural network with a complex design from scratch is out of scope of our work. Thus, we use a publicly available source repository [74] and made customization on top of it. We refer to the architecture that uses `sen2peak` and `YOLOv3-cust` in sequence as `DeepMTL`, our key product. In addition, we use `sen2peak` in combination with the uncustomized original YOLOv3, and refer to it as `DeepMTL-yolo`

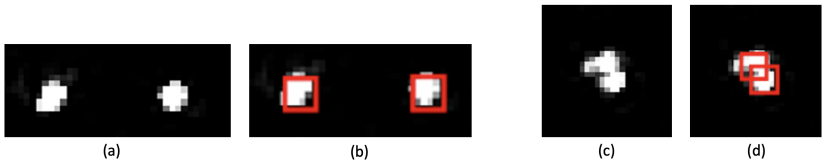


FIGURE 3.6: (a) is the zoom-in of two peaks at the bottom of the Fig. 3.2 example. (c) is the zoom-in of the two close by peaks in the middle right of the Fig. 3.2 example. (b) and (d) shows the bounding boxes that `YOLOv3-cust` outputs for (a) and (c) respectively.

TABLE 3.1: Differences between the original YOLOv3 and our `YOLOv3-cust`.

YOLOv3	`YOLOv3-cust`
Has three YOLO layers at 13×13, 26×26, and 52×52 for detection	Only use the last 52×52 YOLO layer for detection (skip the first two YOLO layers)
Has 3 different rectangle anchors for each YOLO layer	Has 3 square anchors
Every 10 batches, randomly chooses a new input image dimension size	Do not randomly choose new input dimension size
Has 80 different categories of object class	Only has one category for the peak class

(still change the class number to one).

Customization of YOLOv3. Overall, we incorporated four customization to YOLOv3, of which two are significant and the other two are relatively minor. See Table 3.1. YOLOv3 is designed to be a general object detector that can detect objects of various sizes, shapes, and classes within input images of various sizes. However, in our context, the input images are of a fixed size, with only a single class of objects which are relatively small and semi-circular. Based on the above observations, we make changes to the original YOLOv3 that both decrease the model complexity and improve its performance.

Customization Details. The first and second changes presented in Table 3.1 are major changes and we elaborate them in the following paragraphs. Making prediction at three different scales is one of the highlights of YOLOv3 and an improvement comparing to the previous version YOLOv2 which was prone to missing at detecting small objects. As shown in Fig. 3.5, the coarse-grain 13×13 YOLO layer-1 is designed for detecting large size objects, the 26×26 YOLO layer-2 is designed for detecting middle-sized objects, and the fine-grained 52×52 YOLO layer-3 is designed for detecting small-sized objects. Since the peaks in our translated images are always small objects, we only use the last 52×52 YOLO detection layer (and skip the first two YOLO layers). As shown in Fig. 3.5, by "skipping" the two YOLO layers means that we do not use them in computing the overall loss function and their outputs are not used in predicting the bounding boxes. In our `YOLOv3-cust`, the only YOLO layer predicts 8112 bounding boxes, since it has a dimension of 52×52 and each cell results in prediction of 3 bounding boxes; this is in contrast to the original YOLOv3, which predicts 10647 bounding boxes ($3 \times (13 \times 13 + 26 \times 26 + 52 \times 52) = 10647$).

The anchor box is one of the most important hyperparameters of YOLOv3 that can be tuned to improve its performance on a given dataset. The original YOLO's anchor boxes are 10×13, 16×30, and 33×23 (for the input image of size 416×416 pixels), which are essentially bounding boxes of a rectangular shape. These original YOLOv3

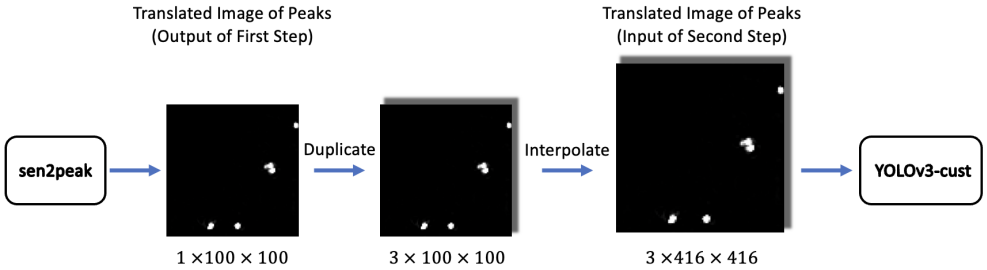


FIGURE 3.7: The data processing of `sen2peak`'s output to get `YOLOv3-cust`'s input of correct size.

anchors were designed for the Microsoft COCO [73] data set, and were chosen since they best describe the dimensions of the real world objects in the MS COCO data set. In our context, since the peaks are generally squares—we use the anchor boxes to be 15×15, 25×25, and 35×35.

Input Image for `YOLOv3-cust`. The first step `sen2peak`'s output image is 100×100, while the second step `YOLOv3-cust`'s input is required[4] to be a three-channel (RGB) image with each channel being size of 416×416. To feed the output of `sen2peak` to `YOLOv3-cust`, we do the following: (i) First, we duplicate the `sen2peak`'s output image to create two more copies and thus create a three-channel image of 100×100 size channels; (ii) Next, we resize the 100×100 channels to 416×416 channels using the PyTorch's default "nearest neighbor" interpolation. See Fig. 3.7.

Output of `YOLOv3-cust`. YOLO treats objected detection as a regression problem. The regression target (or "label") for an object is a five-value tuple $(x, y, \mathit{length}, \mathit{width}, \mathit{class})$. In our case, there is only one class. x and y are real number location coordinates of the center of the bounding box, which we use as the location of the transmitter. Width and height determine the size and shape of the object—which we consistently set to be 5 each to signify a 5×5 square. Note that the center of the bounding box is in the continuous domain. Thus, we are able to get sub-pixel level location of the transmitters.

3.5 Localization in the Presence of Authorized Users

Till now, we have assumed that the only transmitters present in the area are the intruders which need to be localized. In this section, we solve the more general `MTL` problem, where there may be a set of authorized users in the background. This is referred to as the multiple transmitter localization - shared spectrum (`MTL-SS`) problem [136].

In particular, in a shared spectrum paradigm, there are primary users and an evolving set of active secondary users transmitting in the background. Different than the intruders whose locations are unknown, the authorized users' locations are known and we wish to utilize this known information to better localize the unknown intruders. The key challenges come from the fact that the set of authorized users is not static and changes over time as allocation requests are granted and/or active secondary users become inactive over time. A straightforward way to handle background authorized users is to localize every transmitter, and then remove the authorized users. However,

[4]YOLOv3 was developed before our work and the YOLOv3 authors set the input size of the CNN model to $3 \times 416 \times 416$. Although we are customizing their YOLOv3 model, we cannot change the input size because changing it will change the convolutional layer structure, which will preclude us from using the pre-trained weights in the YOLOv3 backbone.

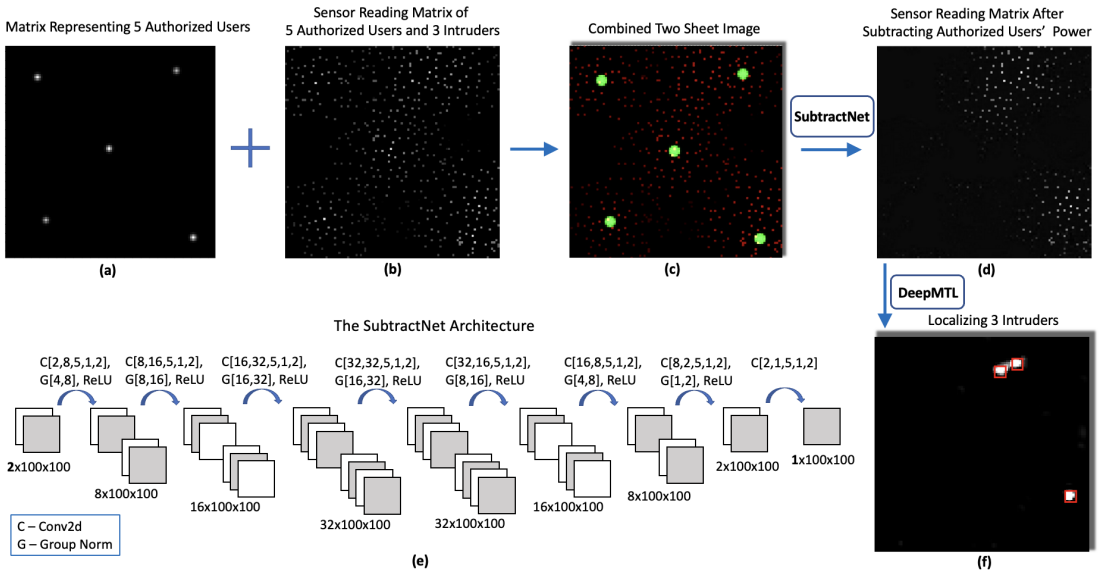


FIGURE 3.8: Overall architecture of second approach to localize 3 intruders in the presence of 5 authorized users. The input of the `SubtractNet` is (c), which is stacking authorized user matrix (a) and the sensor reading matrix (b). (d) is the output of `SubtractNet`, where the transmission power of the authorized users is subtracted from the area. The details of the `SubtractNet` model is in (e). (f) is the localization output after feeding (d) into `DeepMTL`.

any localization approach is susceptible to performance degradation with the increase in the number of transmitters to be localized. Thus, the straightforward approach of localizing every transmitter is likely to be error-prone. Therefore, we attempt to develop a new approach that uses `DeepMTL` as a building block that uses the information of the location of the authorized uses in a way other than removing them after localizing all. The new approach tries to subtract the received signal strength at the sensors by a value received from the authorized users. This subtraction is done by a novel CNN model; we refer to it as `SubtractNet`. Then we feed the image with subtracted powers to the `DeepMTL` and get the locations of the intruders. See Fig. 3.8(c)–(d)–(f). We describe `SubtractNet` in the following paragraphs.

`SubtractNet` Input Image. The sensor reading has two sources, one is the intruders and the other is the authorized users. We aim to subtract the power of the authorized users and remain the power from the intruders. So the input of the `SubtractNet` will contain two kinds of information: the authorized users' information (Fig. 3.8(a)), including both the location and the transmitter power, and the sensor reading matrix (Fig. 3.8(b)) that encode the power from all transmitters. To incorporate the two kinds of information, we first encode the authorized user information into a matrix that has the same dimension as the sensor reading matrix. Then stack the two matrices together. The combined stacked image is nothing but a two-channel image, which can be interpreted as Red and Green channels. The sensor reading matrix is the Red channel and the authorized user matrix is the Green channel. There is no Blue channel. To represent the authorized transmitter in the Green channel, we use a Gaussian peak similar to what we did in the `sen2peak` for representing transmitters (Section 3.3). The difference is that in `sen2peak`, all the peaks have a uniform height, whereas in `SubtractNet`, the height of the peak is the power of the authorized transmitter. So the higher the power of the authorized transmitter, the higher the peak in the Green channel. Another difference is

that the authorized transmitters are approximated at discrete locations instead of the continuous locations as in `sen2peak`.

`SubtractNet` Output Image. The `SubtractNet`'s output image is just a one-channel images and represents the sensor readings due to the intruders only.

`SubtractNet` CNN Architecture. We refer to the model that subtracts the power from the authorized users as the `SubtractNet`. It has a similar design philosophy with `sen2peak`. `SubtractNet` is also an image-to-image translation neural network. Compared to `sen2peak`, it doubled the number of layers, mainly because `SubtractNet` needs a bigger receptive field than `sen2peak`. A bigger receptive field can let the CNN model update sensors that are further away from the authorized user. For the loss function, we use the L2 loss function, similar to the loss function used in Equation 3.1, merely replacing the `sen2peak` with `SubtractNet` in Equation 3.1. The training details are also the same as in `sen2peak`.

3.6 Estimating the Transmit Power of Transmitters

In this section, we extend our techniques to estimate the transmit power of the intruders; we refer to the overall problem as Multiple Transmitter Power Estimation (`MTPE`). Estimation of the transmit power of transmitters can be very useful in the shared spectrum systems. In particular, estimated transmit powers of the primary users (if unknown, as in the case of military users or legacy systems) can be used to set a "protective" region around them—inside which secondary users can be disallowed [115]. Estimating transmit power of secondary users can also be useful. E.g., if the violation in a shared spectrum system is based on a certain minimum threshold, then it is important to estimate the transmit power to determine a violation. Also, the estimated transmit power of secondary users can also be used to "circumvent" their intrusion—i.e., for the primary users to appropriately increase their transmit power to overcome the harmful interference from the secondary users. In general, estimating the transmission power is beneficial to various operations such as node localization, event classification, jammer detection [132].

There are several works that estimates the transmission power of a single transmitter, often jointly with its location [132, 115, 65]. Our previous work [136] can estimate the power of multiple transmitters. The similarity among all four of these methods is that they are estimating the power and location jointly. In this chapter, we propose a new method that leverages the capabilities of `DeepMTL` by using it as a building block. We first localize the transmitters by `DeepMTL`. Then given the localized locations, estimate the transmitters' transmission power by a newly designed CNN model `PredPower`. Although `PredPower` is designed to only estimate the power of a single transmitter, we use it together with a machine learning-based error correction method that can mitigate the errors while applying `PredPower` to the multiple transmitter power estimation scenario.

In this section, we develop a technique to predict the transmission powers of the intruders. Here, for simplicity, we assume no background authorized users, though, the techniques in this section also work in the presence of authorized users. We leverage our accurate and robust localization solver that tolerates varying transmission power for different transmitters (the varying transmission power needs to be in a range). We propose an efficient approach and its overall methodology at a high-level is as follows. And then in the next subsection we describe our `PredPower` model.

1. We use `DeepMTL` to localize the multiple transmitters in a field.
2. We develop a CNN model `PredPower` to predict power of a single isolated (far away from other intruders) intruder.
3. For other (non-isolated) intruders, we still use `PredPower` to predict their powers but employ a post-processing "correction" technique to account for nearby intruders.

3.6.1 `PredPower`: Predicting Power of a Single Isolated TX

`PredPower` Input Image. Let us consider an "isolated" transmitter T. To predict T's power, we start with creating a smaller-size image by cropping the original sensor readings image with the area of a certain size around T. In our evaluations in Section 3.7, the transmitters have a transmit radius[5] of around 20 pixels, which is equivalent to 200 meters.[6] For this setting, we used an cropped area of 21×21 around the isolated transmitter T to predict its power, with T is at the center of this area; also, in this setting, we define a transmitter to be *isolated* if there is no other transmitter within a 20-pixel distance.[7] Note that the above cropping process requires the location of the transmitter to be known, and hence, we undertake the above power-estimation process after the localization of the transmitters using the `DeepMTL` model. We crop images from the same dataset where `DeepMTL` is trained on.

`PredPower` Output Power. The output of the `PredPower` is a single pixel whose value is the predicted power of the transmitter located at the center of the cropped image. Before coming into this single pixel output design, we tried using the height or radius of the peak from the output of `sen2peak` to indicate the power. But we figure out that the height or radius of the peak is hard to accurately predict and therefore is not an accurate indicator of the power. So we reduced the output complexity and designed the output as a simple single pixel whose value directly represents the power of the transmitter. By simplifying both the input side and output side, we can design and implement a novel CNN model that can accurately predict the power of a single transmitter, as described in the following paragraph.

`PredPower` CNN Architecture. We refer to our CNN model that estimates the power of a single transmitter as `PredPower`. See Fig. 3.9. It has a similar design to `sen2peak` as well, where it has no max-pooling layers and no fully connected layers. We do not use the fully connected layers and design a fully-convolutional network since the usage of fully connected layers will destroy the spatial relationships. `PredPower` has five CNN layers and each CNN layer has a kernel size 5×5, striding 1 and padding 0. With this setting, a pixel in the output layer has a receptive field of 21×21, which is exactly the size of the input cropped image. Also note that the pixel is exactly at the location where the transmitter is assumed to be located (recall that the transmitter is at the center

[5] I.e., sensors beyond a distance of 20 pixels away from a transmitter x receive only negligible power from x.

[6] Transmission ranges of a standard 2.4 GHz and 5 GHz WiFi at default transmission powers (100 mW) are roughly 45m and 15m respectively. In our simulations (Section 3.7), we use the 600 MHz frequency band. As the lower the signal frequency, the higher the transmission range, a transmission range of around 200m is reasonable.

[7] Ideally, transmitters with a transmit radius of 20 pixels should entail defining isolated transmitters as ones that have no other transmitters within a 40-pixel distance, and then use a 41×41 area around the isolated transmitter. However, in our evaluations, our chosen values yielded a more efficient technique with sufficient accuracy.

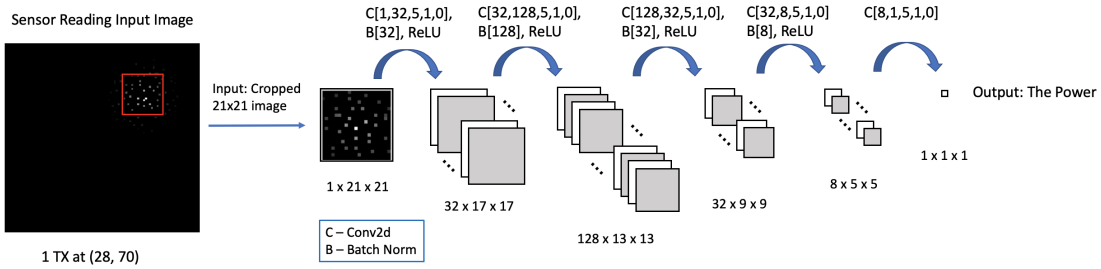


FIGURE 3.9: Architecture of the `PredPower`, a five-layer CNN model that takes in a cropped image from the original input image and outputs the predicted power of one transmitter. The figure displays how the data volume flows through the various convolutional layers. C stands for Conv2d, a 2D convolutional layer, and for each Conv2d layer, the five values shown are [number of input channel, number of output channel, kernel size, stride, padding]. B stands for batch normalization 2d, and for each batch normalization, the value shown is [number-of-features].

of the cropped image). We tried both batch normalization and group normalization and found that batch normalization is better than group normalization, which is the opposite to the `sen2peak` scenario. ReLU is used as the activation function.

Loss Function. The output of the last convolutional layer is technically a 3D cube, although $1 \times 1 \times 1$. So we flatten it in the end to get one scalar value. We use a L2 loss function, which is formally defined as:

$$\frac{1}{N}\sum_{i}^{N}(\texttt{PredPower}(X_i^c) - y_i)^2, \tag{3.2}$$

where N is the number of training samples, X_i^c is the cropped input image for the i^{th} sample and y_i is the ground truth power for the i^{th} sample. $\texttt{PredPower}(X_i^c)$ is the predicted power. We use Adam as the optimizer, and set the learning rate to 0.001 and the number of epochs to 20, which is sufficient for the model convergence.

3.6.2 Estimating Powers of Multiple Transmitters

Our end goal is to estimate the power of multiple transmitters at the same time. When the multiple transmitters are far away and isolated from each other, the problem reduces to single transmitter power estimation, which `PredPower` handles well. The hard part is to estimate transmit powers of multiple transmitters that are close by. In this case, a sensor will receive an aggregated power from multiple transmitters. We assume that blind source power separation is not viable.

Overall High-Level Approach. For each localized intruder by using `DeepMTL` (whether isolated or not), we crop the 21×21 size area around it and feed it to `PredPower`, and estimate its power. If it is actually isolated, then the predicted power is final. If it is not isolated, then we apply a post-processing correction phase to account for the overestimation of the powers, as described below.

Correction Method for Close by Transmitters. Let us first consider the case where there are two close by transmitters T_0 and T_1. We use `PredPower` to estimate the power of two transmitters and get $p_0^{'}$ and $p_1^{'}$ respectively. Let us say the ground truth are p_0 and p_1 respectively. The estimated power will most likely be higher than the ground true power, i.e., $p_0^{'} > p_0$ and $p_1^{'} > p_1$. Because `PredPower` can only "see" one transmitter, and it will view two transmitters in the areas as a combined single one. Let us focus on

T_0 and assume $\delta_0 = p_0^{'} - p_0$. The intuition is that δ_0 has some underlying patterns that we are able to recognize. We model δ_0 as a function of some features related to T_0 and T_1. We model δ_0 as follows,

$$\delta_0 = \theta_0 \cdot p_0^{'} + \theta_{(1,1)} \cdot d_{01} + \theta_{(1,2)} \cdot p_1^{'} + \theta_{(1,3)} \cdot \frac{p_1^{'}}{d_{01}} \tag{3.3}$$

where d_{01} is the distance between T_0 and T_1, and the four θs are the coefficients for the four terms respectively. The first term is related to T_0 itself, and the other three terms are related to T_1. We observe that the smaller the d_{01}, the larger the value of δ_0. And the bigger the $p_1^{'}$, the larger the value of δ_0. So d_{01} has a negative correlation with δ_0 while $p_1^{'}$ has a positive correlation. $\frac{p_1^{'}}{d_{01}}$ is a combination of two terms to increase the number of features. We also tried a few other features, but we decided to use only these three features for a close by transmitter as a balance of model accuracy and model complexity.

Equation 3.3 is for the case of one close by transmitter, we then extend the equation to handle multiple close by transmitters in the following Equation 3.4,

$$\delta_0 = \theta_0 \cdot p_0^{'} + \sum_{i=1}^{m} (\theta_{(i,1)} \cdot d_{0i} + \theta_{(i,2)} \cdot p_i^{'} + \theta_{(i,3)} \cdot \frac{p_i^{'}}{d_{0i}}) \tag{3.4}$$

where m is the number of close by transmitters for T_0, the transmitter of interest, d_{0i} is the distance between T_0 and close by T_i, and $p_i^{'}$ is the uncorrected power predicted by `PredPower`. For the ith close by transmitter, we introduce three terms $d_{0i}, p_i^{'}, \frac{p_i^{'}}{d_{0i}}$, and assign three coefficients $\theta_{(i,1)}, \theta_{(i,2)}, \theta_{(i,3)}$ to the three terms respectively. So for m close by transmitters, there are $1 + 3m$ number of terms in the Equation 3.4.

After modeling δ_0, in Equation 3.5, we "correct" $p_0^{'}$ by subtracting δ_0 from $p_0^{'}$ to get more an accurate estimation of the power of transmitter T_0.

$$p_0^{correct} = p_0^{'} - \delta_0 \tag{3.5}$$

Estimating the parameter θ. Equation 3.4 is essentially a linear model and we can train it by using either linear, ridge, or LASSO regression models [90]. We perform experiments using ridge regression (alpha=0.01). We set a distance threshold for a neighbor transmitter to be classified as a close by transmitter. Note that the transmitters will have a different number of close by transmitters. So, let us denote M as the maximum number of close by transmitters we see in the dataset. When training the linear model in Equation 3.4, we train a model that assumes a maximum M number of close by transmitters, i.e., the linear model has $1 + 3M$ terms. The $3M$ terms are organized in a group of three (i.e., three features) and the groups are sorted by distance in an ascending order. Then, for a transmitter with a smaller than M number of close by transmitters, let us say m, only the first $1 + 3m$ terms will have a meaningful value. And for the rest $3(M - m)$ terms, we set the value to zero, i.e., impute missing value with zero.

3.7 Evaluation

To evaluate the performance of our proposed techniques, we conduct large-scale simulations over two settings based on two different propagation models. In particular,

we consider the log-distance-based propagation model and the Longley–Rice model obtained from SPLAT! [79]. We evaluate various algorithms, using multiple performance metrics as described below.

Performance Metrics. We use the following metrics 1, 2, and 3 to evaluate the localization methods and use the 4th metric to evaluate the power estimation methods.

1. Localization Error (L_{err})
2. Miss rate (M_r)
3. False Alarm rate (F_r)
4. Power Error (P_{err})

Given a multi-transmitter localization solution, we first compute the L_{err} as the minimum-cost matching in the bi-partite graph over the ground truth and the solution's locations, where the cost of each edge in the graph is the Euclidean distance between the matched ground truth node location and the solution's node location. We use a simple greedy algorithm to compute the min-cost matching. The unmatched nodes are regarded as false alarms or misses. We also put an upper threshold on the cost (L_{err}) of an eligible match. E.g., if there are four intruders in reality, but the algorithm predicts six intruders then it is said to incur zero misses and two false alarms, so the M_r is zero and the F_r is one-third. If the algorithm predicts three intruders then it incurs one miss and zero false alarms, so the M_r is one-fourth and the F_r is zero. In the plots, we stack the miss rate and false alarm rate to reflect the overall performance.

Algorithms Compared. We implement[8] and compare six algorithms in two stages. In stage one, we compare three versions of our techniques, viz., `DeepMTL`, `DeepMTL-yolo`, and `DeepMTL-peak`. Recall that `DeepMTL`, `DeepMTL-yolo`, and `DeepMTL-peak` use `sen2peak` in the first step, and `YOLOv3-cust`, original YOLOv3, and `simplePeak` respectively in the second step. In the first stage of our evaluations, we will show that `DeepMTL` outperforms `DeepMTL-yolo` and `DeepMTL-peak` in almost all performance metrics. Thus, in the second stage, we only compare `DeepMTL` with schemes from three prior works, viz., `SPLOT` [63], `DeepTxFinder` [140], and `MAP` [136] and show that `DeepMTL` outperforms the prior works.

Training and Testing Dataset. We consider an area of $1km \times 1km$, and use grid cells (pixels) of $10m \times 10m$, so the grid is 100×100. The transmitters may be deployed anywhere within a cell (i.e., their location is in the continuous domain), while the sensors are deployed at the centers of the grid cells (i.e. their location is in the discrete domain). For each instance (training or test sample), the said number of sensors and transmitters are deployed in the field randomly. For each of the two settings (propagation models described below), we create a 100,000 sample training dataset to train our models and create another 20,000 sample testing dataset to evaluate the trained model.

We will evaluate the performance of various techniques for varying number of transmitters/intruders and sensor density. When we vary a specific parameter, the other parameter is set to its *default* value; the number of transmitters varies from 1 to 10 and the default value is 5; the sensor density varies from 1% to 10% and the default value is 6% (600 sensors in a 100×100 grid). The two default numbers 5 and 6% are chosen because they are in the middle of their ranges. When not mentioned, the default values are used. The transmitter power varies from 0 to 5 dBm and is randomly picked. To

[8]Source code at: https://github.com/caitaozhan/deeplearning-localization.

minimize overfitting, the training dataset and testing dataset have sensors placed at completely different locations.

We train the `DeepMTL` model using the 100,000 sample dataset. To train the `DeepTxFinder` [140], we partition the 100,000 sample training dataset into ten datasets based on the number of transmitters in the samples which varies from 1 to 10. These ten datasets are used to train the ten "localization" CNN models in `DeepTxFinder`, and the full dataset of 100,000 samples is used to train the `DeepTxFinder` model that determines the number of transmitters. For the `MAP` scheme [136], we assume the availability of all required probability distributions. We note that using a simple cost model (number of samples need to be gathered), the overall training cost for `MAP` is an order of magnitude higher than `DeepMTL` and `DeepTxFinder`. Lastly, `SPLOT` [63] does not require any training.

Two Propagation Models and Settings. The sensor readings (i.e. the dataset) are simulated based on a propagation model. To demonstrate the generality of our techniques, we consider two propagation models as described below.

Log-Distance Propagation Model and Setting. Log-Distance propagation model is a generic model that extends Friis Free space model which is used to predict the path loss for a wide range of environments. As per this model, the path loss (in dB) between two points x and y at a distance d is given by: $PL_d = 10\alpha \log d + \mathcal{X}$, where α (we use 3.5) is the path-loss exponent and $\mathcal{X}$ represents the shadowing effect that can be represented by a zero-mean Gaussian distribution with a certain (we use 1) standard deviation. Power received (in dBm) at point y due to a transmitter at point x with a transmit power of P_x is thus: $P_x - PL_d$. Power received at point y due to multiple sources is assumed to be just an aggregate of the powers (in linear) received from each of the sources.

SPLAT! Model and Setting. This is a complex model of wireless propagation based on many parameters including locations, terrain data, obstructions, soil conditions, etc. We use SPLAT! [79] to generate path-loss values. SPLAT! is an open-source software implementing the Longley-Rice [27] Irregular Terrain With Obstruction Model (ITWOM) model. We consider a random area in Long Island, New York of $1km \times 1km$ large and use the 600 MHz band to generate path losses.

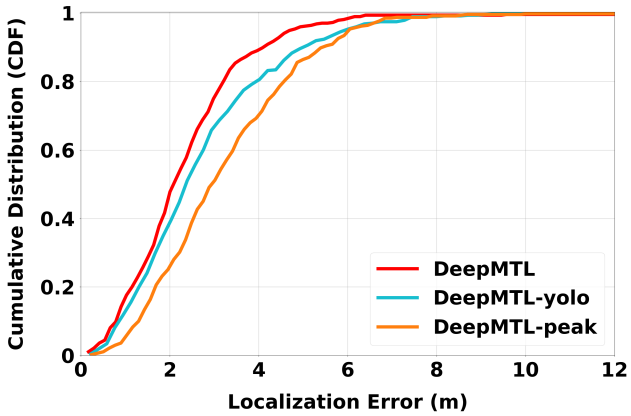


FIGURE 3.10: Cumulative probability of localization error of `DeepMTL`, `DeepMTL-yolo` and `DeepMTL-peak`, for the special case of single transmitter localization with 6% sensor density.

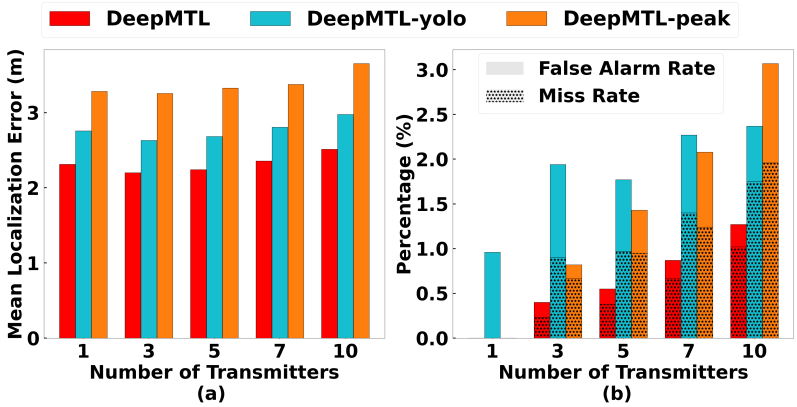


FIGURE 3.11: (a) Localization error and (b) miss and false alarm rates, of `DeepMTL`, `DeepMTL-yolo` and `DeepMTL-peak` variants for varying number of transmitters in log-distance dataset (propagation) model.

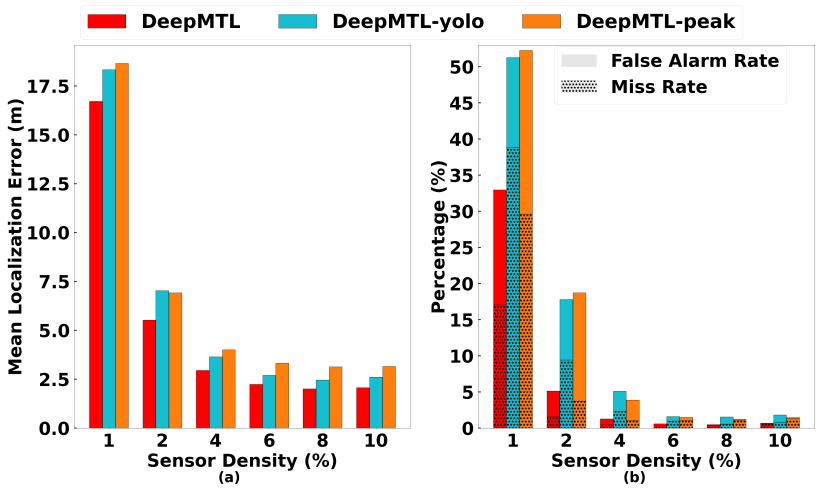


FIGURE 3.12: (a) Localization error and (b) miss and false alarm rates, of `DeepMTL`, `DeepMTL-yolo` and `DeepMTL-peak` variants for varying sensor density in log-distance dataset (propagation) model.

3.7.1 `DeepMTL` vs. `DeepMTL-yolo` vs. `DeepMTL-peak`

In this subsection, we compare the three variants of our technique, viz., `DeepMTL`, `DeepMTL-yolo`, and `DeepMTL-peak`. For simplicity, we only show plots for the log-distance propagation model setting in this subsection (we observed similar performance trends for the Longley-Rice propagation model too).

Performance Results. In Fig. 3.10, we plot the cumulative density function (CDF) of the localization error, for the simple case of a single transmitter. We observe that `DeepMTL` outperforms the other variants, as it yields a higher cumulative probability for a lower range of errors. In addition, we evaluate the three variants for varying number of transmitters (Fig. 3.11) and sensor density (Fig. 3.12), and evaluate the localization error as well as the false alarm and miss rates. We observe that `DeepMTL` consistently outperforms the other two variants across all plots and performance metrics. As expected, the performance of all algorithms degrades with an increase in the number of transmitters (in terms of false alarms and miss rates) or with a decrease in sensor

density. In general, the localization error of `DeepMTL` is around 15-30% lower than the other variants. Impressively, the total cardinality error (i.e., false alarms plus miss rates) is fewer than 1% for the `DeepMTL` technique, when the sensor density is 6% or above.

When the sensor density is as low as 1%, the performance of all methods significantly decreases. Because when the sensor density is 1% or lower, the input image will be very sparse and contain only a few pixels. `DeepMTL`'s first part `sen2peak` has a receptive field of 17×17. This area will contain an average of less than three sensors when the sensor density is 1% ($17 \times 17 \times 0.01 = 2.89$). This number is considered too low and note that 2.89 sensors are not enough for the trilateration localization method, which needs three sensors. Our CNN models need to function well with enough pixels that contain useful information. So we suggest the sensor density to be at least 2% to achieve reasonable results.

TABLE 3.2: Compare Localization Running Time (s) for 1 to 10 Number of Intruders

Intru.	`DeepMTL-peak`	`DeepMTL-yolo`	`DeepMTL`	`MAP`	`SPLOT`	`DeepTxFinder`
1	0.0013	0.0180	0.0180	8.78	1.53	0.0015
3	0.0014	0.0183	0.0186	15.1	1.79	0.0016
5	0.0016	0.0192	0.0189	19.3	2.06	0.0017
7	0.0018	0.0196	0.0194	24.1	2.32	0.0019
10	0.0023	0.0205	0.0206	28.5	2.72	0.0022

Running Time Comparison. For the running time comparison of the variants, see Table 3.2. Our hardware is an Intel i7-8700 CPU and an Nvidia RTX 2070 GPU. We observe that, as expected, `DeepMTL` and `DeepMTL-yolo` which use a sophisticated object-detection method do incur higher latency (around 20 milliseconds) than `DeepMTL-peak` (around two milliseconds). As our key performance criteria is accuracy and the run time of `DeepMTL` is still quite low, we choose `DeepMTL` for comparison with the prior works in §3.7.2.

Localizing Transmitters Close By. Localizing two or more transmitters close by is a hard part of the `MTL` problem. Fig. 3.6(c) and (d) gives an example of when an advanced object detection algorithm will work while a simple local maximal peak detection might not. Fig. 3.6(c) and (d) shows `DeepMTL` can successfully localize two transmitters as close as three pixels apart. When a pixel represents a $10m \times 10m$ area, then it is 30 meters apart. If a pixel represents a smaller area, such as $1m \times 1m$, it has the potential to localize two transmitters as close as three meters apart.

Two YOLO Thresholds. YOLO has two important thresholds to tune that can affect the miss rate and false alarm rate. One is the confidence threshold (`conf`) and the other is the non-maximum suppression threshold (`nms`). An object will be recognized as a peak only if its confidence level is larger than `conf`. If two recognized peaks' bounding boxes have a large overlap, and their intersection of union is higher than `nms`, then the two peaks will be considered as one peak. The peak with a higher confidence level keeps while the other peak with a lower confidence level discards. A higher `conf` will bring a lower false alarm rate but a higher miss rate, and a higher `nms` will bring a lower miss rate but a higher false alarm rate. We pick `conf` $= 0.8$ and `nms` $= 0.5$ for `DeepMTL` as we observe these values bring a good balance between false alarm rate and miss rate. In particular, a high `conf` of 0.8 precludes "fake peaks" at locations with no transmitters. Also, a low `nms` weakens `DeepMTL`'s ability to localize two close by transmitters, while a high *nms* yields a high false alarm rate (by incorrectly interpreting a single transmitter as multiple close by transmitters); thus, we chose `nms` of 0.5.

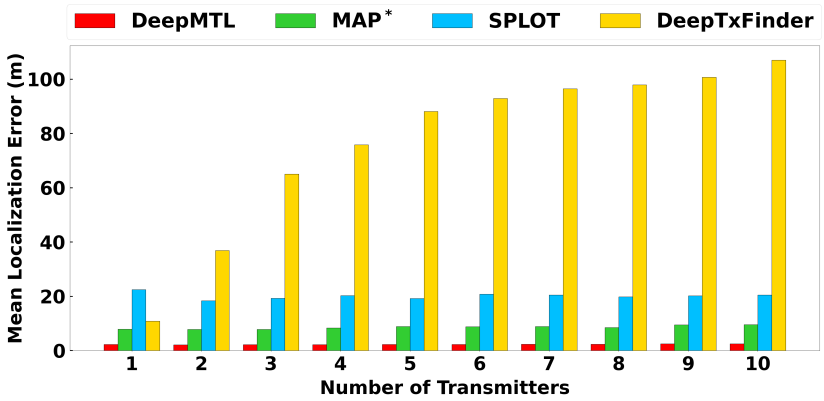


FIGURE 3.13: Localization error of `DeepMTL`, `MAP`, `SPLOT`, and `DeepTxFinder` for varying number of transmitters in the log-distance dataset.

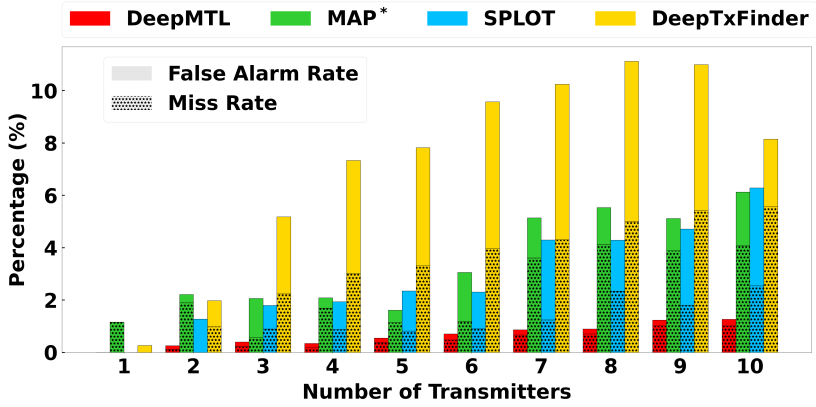


FIGURE 3.14: Miss and false alarm rates of `DeepMTL`, `MAP`, `SPLOT`, and `DeepTxFinder` for varying number of transmitters in the log-distance dataset.

3.7.2 `DeepMTL` vs. Prior Works

In this subsection, we compare `DeepMTL` with `SPLOT`, `MAP`, `DeepTxFinder` in both log-distance (Fig. 3.13, 3.14, 3.15) and SPLAT (Fig. 3.16, 3.17, 3.18) propagation models and thus, datasets. We observe similar performance trends for both datasets, i.e., `DeepMTL` significantly outperforms the other approaches by a large margin (in many cases, by more than 50% in localization errors, false alarms, and miss rates). For all techniques, as expected, the performance is generally worse in the SPLAT dataset compared to the log-distance dataset.

Varying Number of Transmitters. Fig. 3.13 and Fig. 3.16 show the localization error with varying number of transmitters, in the two datasets. We see that `DeepMTL` has a mean localization error of only 2 to 2.5 meters (roughly, one-fourth of the side length of a pixel/grid cell) in the log-distance dataset and about 5 to 6 meters in the SPLAT dataset. In comparison, the localization errors of `MAP`, `SPLOT`, `DeepTxFinder` are two to three times, eight to nine times, and few tens of times respectively more than that of `DeepMTL`. Fig. 3.14 and Fig. 3.17 show the miss and false alarm rates with varying number of transmitters in the two datasets. We observe that `DeepMTL`'s summation of miss and false alarm rate is only 1% even at ten transmitters in the log-distance dataset, and about 4% for the case of SPLAT! dataset. In comparison, the summation of miss and false alarm rates for other schemes is at least 6% and 10% respectively for the two

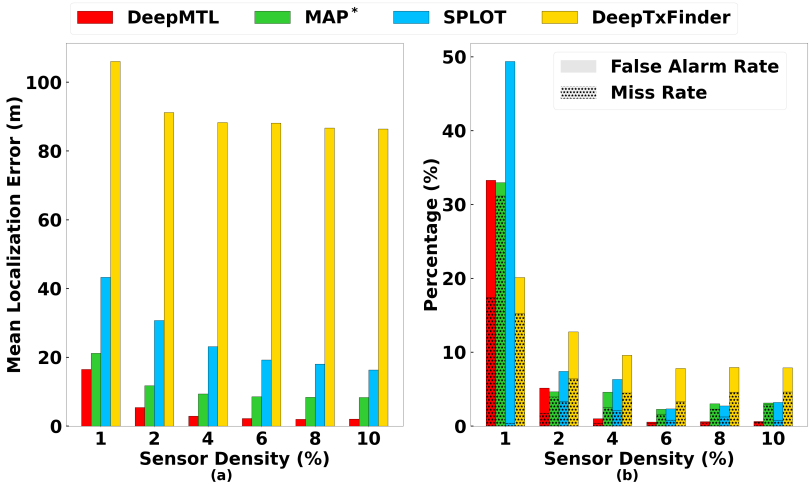


FIGURE 3.15: (a) Localization error, and (b) miss and false alarm rates, of `DeepMTL`, `MAP`, `SPLOT`, and `DeepTxFinder` for varying sensor densities in the log-distance dataset.

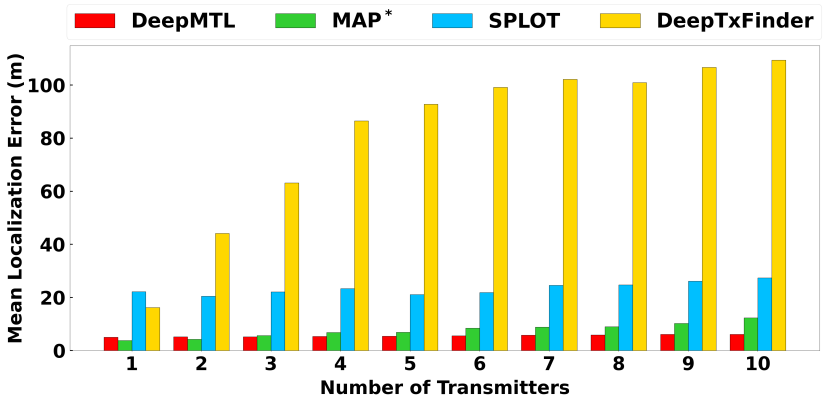


FIGURE 3.16: Localization error of `DeepMTL`, `MAP`, `DeepTxFinder` and `SPLOT` for varying number of transmitters in the SPLAT! Dataset.

datasets, when there are ten transmitters.

Varying Sensor Density. Fig. 3.15 and Fig. 3.18 plot the performance of various algorithms for varying sensor density in the two datasets. For very low sensor density of 1%, all algorithms perform badly (in comparison with higher sensor densities), but `DeepMTL` still performs the best except that `MAP` performs best at 1% in terms of false alarm rate and miss rate. For higher sensor densities, we observe a similar performance trend as above—i.e., `DeepMTL` easily outperforms the other schemes by a large margin. For the SPLAT! dataset at the 6% sensor density, the summation of false alarm rate and miss rate is 2%, which is higher than the 1% summation for the log-distance dataset.

Running Times. The run time of `DeepMTL` (in tens of milliseconds) is orders of magnitude faster than `MAP` and `SPLOT` (both in seconds). See Table 3.2. The `DeepMTL` run time is an order of magnitude slower than `DeepTxFinder` (in a few milliseconds), due to the deep `YOLOv3-cust` taking up over 90% of the run time.

Summary and Analysis. In summary, our approach significantly outperforms the other approaches in all the accuracy performance metrics, as well as in terms of latency. In particular, our approach also significantly outperforms the other CNN-based approach `DeepTxFinder`. The main reason for `DeepTxFinder`'s inferior performance is its inability

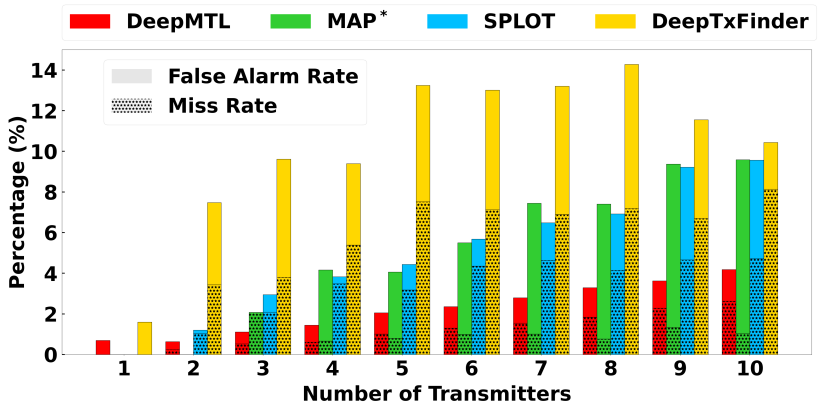


FIGURE 3.17: Miss and false alarm rates of `DeepMTL`, `MAP`, `SPLOT`, and `DeepTxFinder` for varying number of transmitters in the SPLAT! Dataset.

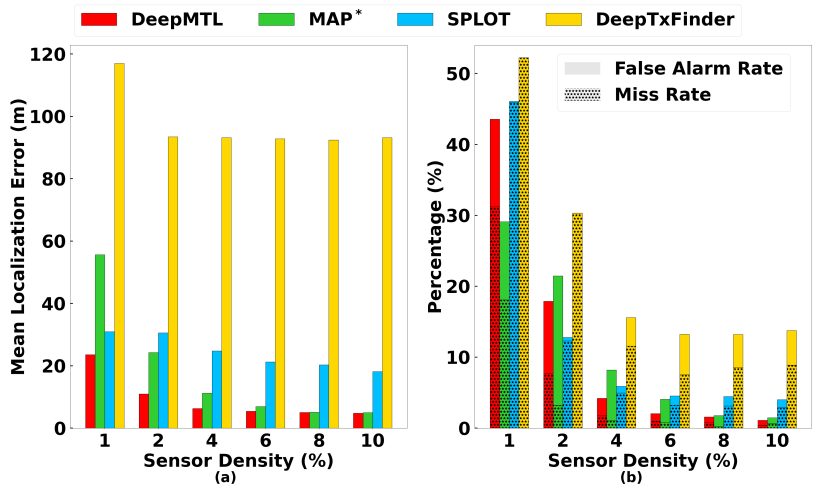


FIGURE 3.18: (a) Localization error, and (b) miss and false alarm rates, of `DeepMTL`, `MAP`, `SPLOT`, and `DeepTxFinder` for varying sensor densities in the SPLAT! Dataset.

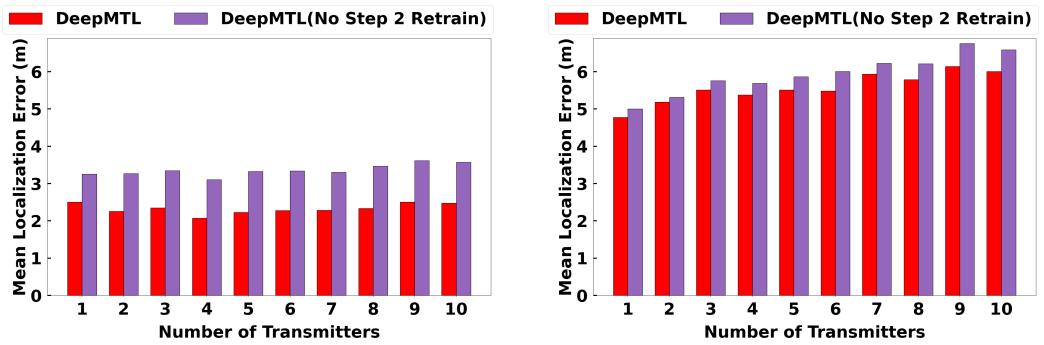
to accurately predict the number of TXs—which forms a fundamental component of their technique. In contrast, `DeepMTL` can circumvent explicit pre-prediction of number of transmitters by using a well-developed object-detection technique which works well for multiple objects especially in our context of simple objects.

3.7.3 Transfer Learning

We demonstrate transfer learning (generalizability) by showing that the second step in `DeepMTL` does not need to be retrained for different radio frequency propagation models and terrains. In the previous experiments, the two steps of `DeepMTL` are both trained in the same setting, either log-distance or SPLAT!. We do the following two combinations to show that the second step does not need to retrain:

1. The first step is trained in the log-distance setting and the second step is trained in the SPLAT! setting. Tested on the log-distance data.

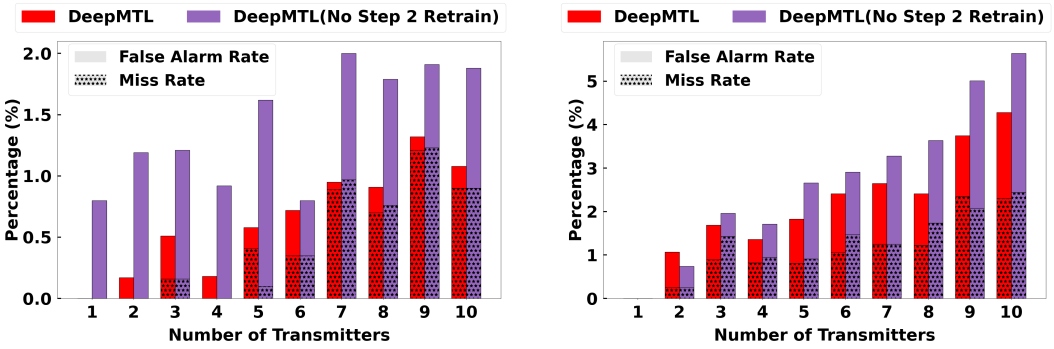
2. The first step is trained in the SPLAT! setting and the second step is trained in the log-distance setting. Tested on the SPLAT! data.



(A) First step trained in log-distance data, while the second step trained in SPLAT! data. Tested on the log-distance data.

(B) First step trained in SPLAT! data, while second the step trained in log-distance data. Tested on the SPLAT! data.

FIGURE 3.19: Localization error for varying number of transmitters when the first and second step of `DeepMTL` are trained on different training dataset.



(A) First step trained in log-distance data, while second step trained in SPLAT! data. Tested on the log-distance data.

(B) First step trained in SPLAT! data, while second step trained in log-distance! data. Tested on the SPLAT! data.

FIGURE 3.20: The miss rate and false alarm rate for varying number of transmitters when the first and second step of `DeepMTL` are trained on different training dataset.

In both combinations, the second step `YOLOv3-cust` is trained on a different dataset compared to the first step `sen2peak`. Fig. 3.19a shows that the localization error increases one-third in the first combination compared to the case where both the first and second steps are trained on log-distance dataset. Fig. 3.19b shows that the localization error increases only five percent in the second combination compared to the case where both the first and second steps are trained on SPLAT! dataset. The miss rate and false alarm rate for both combinations also increase minimally, i.e. the summation of miss rate and false alarm rate only increases around 1% in absolute value. See Fig. 3.20. This implies that the second step of `DeepMTL`, `YOLOv3-cust`, is general and does not need to retrain for different radio frequency propagation models and terrains. This is because the first step `sen2peak` is translating sensor readings images from different geographical areas to the same Gaussian peaks. The first step `sen2peak` still needs to be retrained for different situations to translate the sensor readings to the peaks.

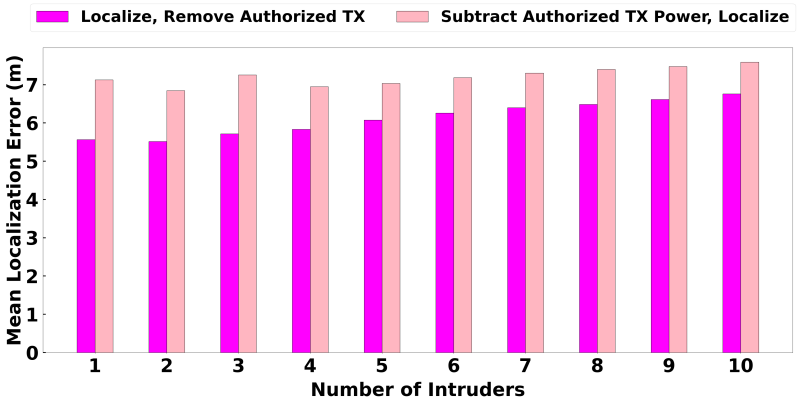


FIGURE 3.21: The localization error of two approaches in the presence of five authorized users with varying number of intruders.

3.7.4 Localize Intruders in the Presence of Authorized Users

The previous experiment setting is based on the assumption that all transmitters we are localizing are intruders. Different than the previous setting, here, we put five authorized users and they are spread out in the field, so those five will not interfere with each other. This is the more general version of the `MTL` problem, where there are some authorized users in the background. Fig. 3.21 shows the localization error of two approaches localizing intruders in the presence of five authorized users with a varying number of intruders. It is observed that the first approach, localize then remove authorized users, has a ten to twenty percent smaller localization error compared to the second approach, subtract authorized user power then localize. This is due to the inaccuracy of power subtraction from the `SubtractNet`. Fig. 3.22 shows the miss and false alarm of two approaches localizing intruders in the presence of five authorized users with a varying number of intruders. It is observed that the second approach, subtract authorized TX power then localize, is having a high false alarm when the number of intruders is three or less. So for `SubtractNet`, subtracting the power of five background authorized users from six transmitters (five out of six transmitters are authorized users, one intruder) is relatively more difficult than subtracting the power of five authorized users from nine users (five out of nine transmitters are authorized users, four intruders). Also statistically, getting one false alarm when there are one intruder and five authorized users is 100% false alarm rate, while getting one false alarm when there are two intruders and five authorized users is only 50% false alarm rate (the denominator is the number of intruders). Thus, the false alarm rate for one and two number of intruders looks to differ a lot, but in reality, the false alarm cases do not differ a lot). When the number of intruders is three or four, the two approaches are comparable. But when the number of intruders is larger than four, the second approach is having a lower miss and false alarm rate. In summary, the two approaches both have their strengths. The main advantage for the second approach is that the sum of miss rate and false alarm rate is lower when the number of intruders is large.

3.7.5 Power Estimation Evaluation

In this subsection, we evaluate the transmitter power estimation performance. In all experiments, the power range is 5 dB. The power error is presented in absolute value. A power error of 0.5 dB implies a relative power error of 10%. First, we compare the

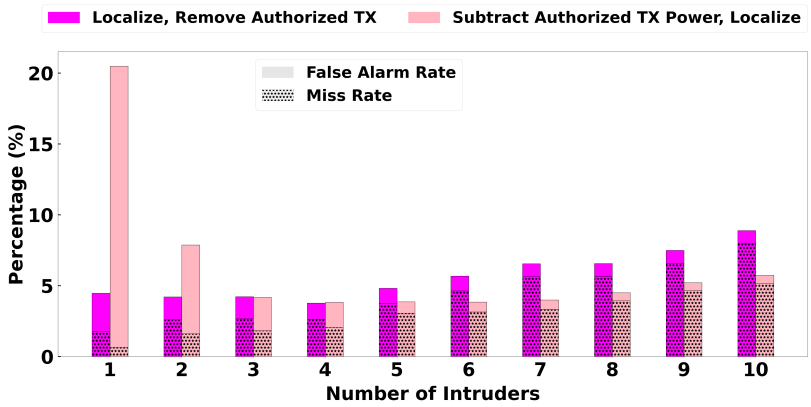


FIGURE 3.22: The miss and false alarm of two localization approaches in the presence of 5 authorized users with varying number of intruders.

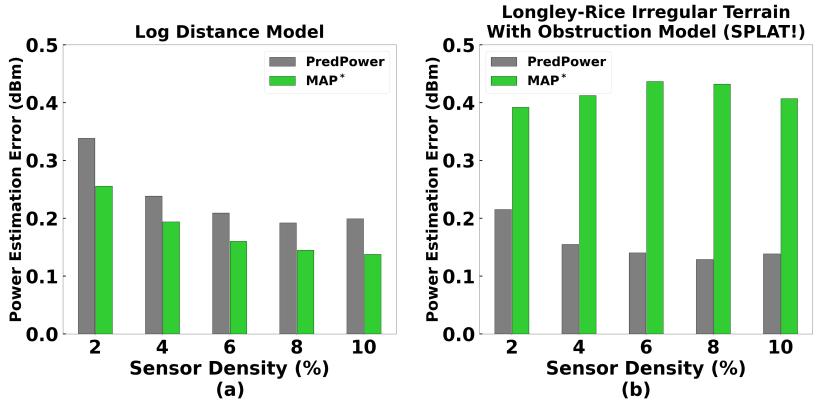


FIGURE 3.23: The single transmitter power estimation error of `PredPower` and `MAP` in two propagation models, (a) Log-distance model and (b) Longley–Rice Irregular Terrain with Obstruction Model (SPLAT!), for varying sensor densities.

single transmitter power estimation between `MAP` and `PredPower`, and then compare the multiple transmitter power estimation between `MAP`, `PredPower` with error correction, and `PredPower` with error correction.

Figure 3.23(a) shows the performance of single transmitter power estimation in the log-distance propagation model scenario with varying sensor density. In this case, `MAP` has a 10 to 20 percent smaller power estimation error. Figure 3.23(b) shows the performance of single transmitter power estimation in the SPLAT! model with varying sensor density. In this case, `PredPower` is significantly lower in power error. So in average, `PredPower` outperforms `MAP` in single transmitter power estimation. We can also conclude that for `PredPower`, a higher sensor density will decrease the power estimation error. While a 2% of sensor density will lead to a higher error, a sensor density of 6% is enough to give relatively good results.

For multiple transmitter power estimation, we compare three methods in two propagation models and show that `PredPower` with error correction has the best performance among the three methods. `PredPower` without error correction is expected to perform the worst and it suggests that the post-processing error correction stage for `PredPower` is important and works well. Figure 3.24 shows the power estimation error of three methods with a varying number of transmitters while the sensor density is 6%. In this

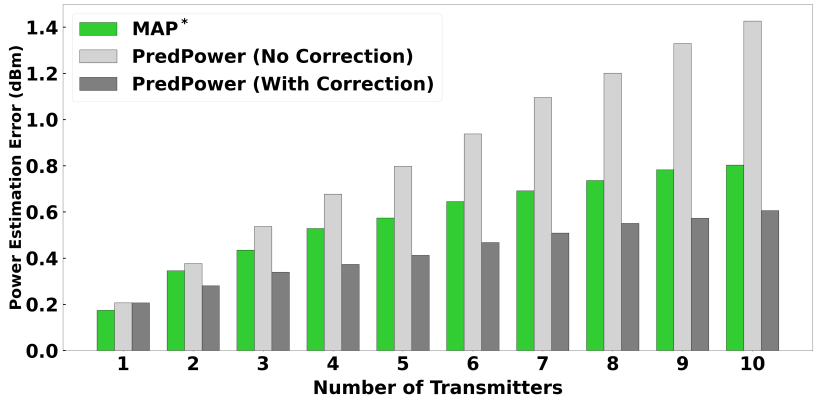


FIGURE 3.24: The transmitter power estimation error of `MAP`, `PredPower` with and without correction in Log-distance model for varying number of intruders

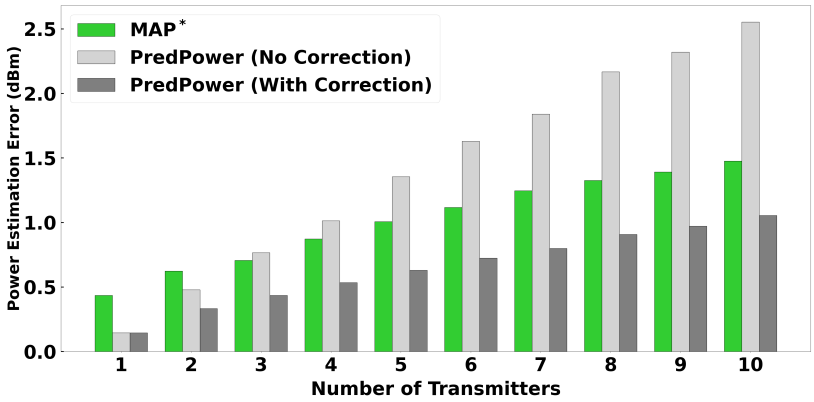


FIGURE 3.25: The transmitter power estimation error of `MAP`, `PredPower` with and without correction in Longley–Rice Irregular Terrain with Obstruction Model (SPLAT!) for varying number of intruders.

figure, `MAP` is the best only when the number of transmitters is one (which is consistent with Fig 3.23(a)). Also the number of transmitters is one is the only case when `PredPower` with correction and without correction has the same performance. This is also expected because there is no need to error correction when there is only one transmitter in the area. In all other cases, we see that `PredPower` with error correction is the best, `PredPower` without error correction is the worst, and `MAP` is in the middle. In Figure 3.25, which shows experiment results running in the SPLAT! propagation model, we see a similar pattern compared to Figure 3.24. The difference is that `PredPower` with error correction is always the best and the power error is larger than the log-distance model scenario. For example in Figure 3.24, the power estimation error for `PredPower` with error correction goes up to 0.6 dB, where as in Figure 3.25, the error goes up to 1 dB.

3.7.6 Evaluation over Testbed Data

In this subsection, we show that our `DeepMTL` performs well in real-world collected data. For this, we repurpose our testbed data from [136] as described below. We start with describing our testbed data from [136].

Testbed Data. In [136], we conducted a testbed in an outdoor parking area of $32m \times 32m$ large.[9] Each grid cell has a size of $3.2m \times 3.2m$, with the grid size being 10×10. We place a total of 18 sensors on the ground. The sensors consist of Odroid-C2 (a single-board computer) connected to an RTL-SDR dongle and the RTL-SDR connects to dipole antennas. The transmitters are USRP or HackRF connecting to a laptop. We collect raw Inphase-Quadrature (I/Q) samples from the RTL-SDR at the 915 MHz ISM band. We perform FFT on the I/Q samples with a bin size of 256 samples to get the signal power values, and then utilize the mean and standard deviation of the power at frequency 915 MHz reported from each of the sensors.

Transforming the Data from 10×10 to 100×100 Grid. Note that `DeepMTL`'s input requires a 100×100 input, while the above data is over a 10×10 grid. Also, the sensor density in the above data is 18%, while we desire a sensor density of around 4-6% to have a fair comparison with our simulation based evaluations in previous subsections. To achieve these objectives, we transform the above 10×10 data to a 100×100 grid data in two steps as follows.

1. Increase the data granularity from 10×10 to 20×20, by dividing each cell into 2×2 cells; we randomly pick one of these four smaller cells to represent the original cell (i.e., to place the sensor if it existed in the original cell). See the red-bordered boxes in Fig. 3.26(a)-(b). We refer to the full 20×20 grid as a *tile*.

2. Now, we duplicate the 20×20 tile 25 times using a 5×5 pattern to generate a 100×100 grid. See Fig. 3.26(b)-(c).

The above steps effectively increase the area from the original $32m \times 32m$ to $160m \times 160m$. Note that the first step above only splits each original cell into four smaller cells without increasing the whole area size. The 100×100 grid will have a sensor density of 4.5% and each grid cell represents an area of $1.6m \times 1.6m$.

We note that the second duplication step can introduce inaccurate sensor readings at the tile's "edges", due to any transmitters from adjoining tiles. To circumvent this issue, we place *transmitters* only within the internal 10×10 cells of each 20×20 tile (i.e., avoid placing a transmitter on the five-cell edge of each tile). This yields a total of 2500 potential positions to place a transmitter in the final 100×100 grid. With the above setting, we generate training and testing datasets consisting of 25,000 and 12,500 samples respectively.

Testbed Results. The performance of `DeepMTL` on this real world based data is shown in Fig. 3.27. Compared to `DeepTxFinder`, `DeepMTL` is significantly better in localization error and false alarm rate and miss rate in almost all cases, which aligns to the results in the previous subsections based on data generated from either log-distance model or SPLAT!. The localization error of `DeepMTL` in Fig. 3.27(a) is around 1.3 meters. The error increases mildly with the increase in the number of transmitters. The localization error in the testbed data is smaller compared to both log-distance data results (Fig. 3.13) and SPLAT! data results (Fig. 3.16). This is because a grid cell here is representing a smaller area. In the log-distance data, the localization error is roughly one-fourth the side length of the grid cell. In the SPLAT! data result, the localization error is roughly half the side length of its grid length. In the testbed data, the localization is roughly eighty percent the side length of a grid cell. So the localization error in the testbed data is the highest relative to the length of a grid cell it represents. The sum of false

[9] Dataset publicly available at: https://github.com/Wings-Lab/IPSN-2020-data

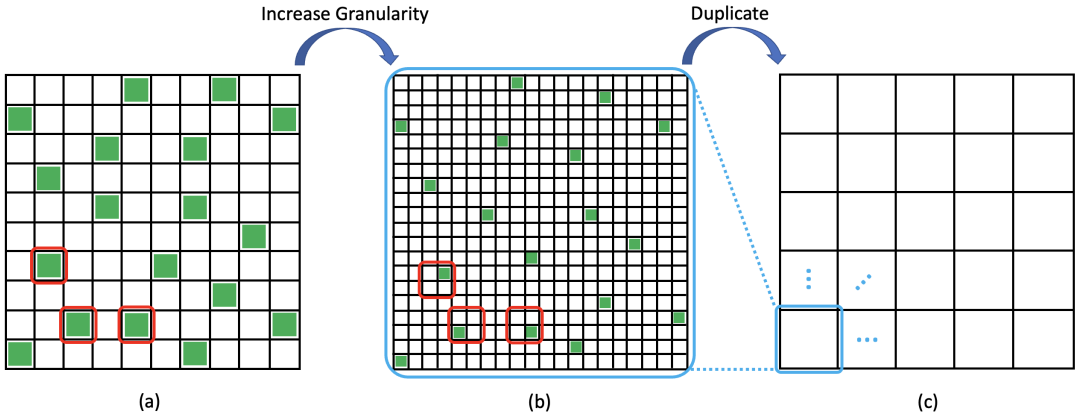


FIGURE 3.26: (a). The original 10×10 testbed grid with 18 sensors (green cells) representing a $32m \times 32m$ area. (b). The 20×20 grid (a tile) obtained by replacing each original cell by 2×2 smaller cells; a sensor, if present in the original cell, is placed in a random cell within the 2×2 grid (see the green cells). (c). The final 100×100 grid obtained by duplicating the 20×20 tile 25 times using a 5×5 pattern. The final geographic area is $160m \times 160m$.

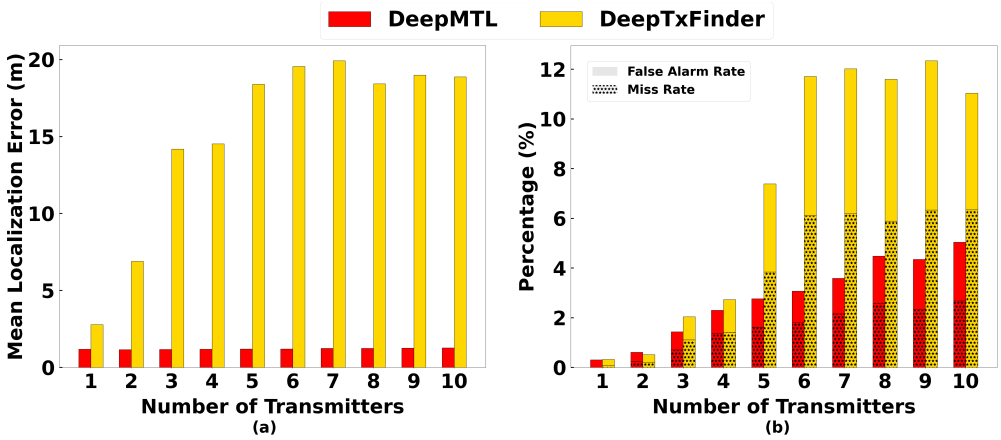


FIGURE 3.27: The localization error (a), false alarm rate and miss rate (b) of `DeepMTL` and `DeepTxFinder` in a real world collected data for varying number of intruders.

alarm rate and miss rate is 3% when the number of transmitters is five and is 5% when the number of transmitters is ten. The results are a little bit worse than the results in the SPLAT! data (Fig. 3.17), where the sum is 2% for five transmitters and 4% for ten transmitters.

3.8 Related Work

Spectrum sensing is usually being realized by some distributed crowdsourced low-cost sensors. Electrosense [94] and its follow up work Skysense [98] are typical work of spectrum sensing. In this crowdsourced sensing paradigm [24], sensors collect I/Q samples (in-phase and quadrature components of raw signals) and compute PSD (power spectral density), which is RSS. Crowdsourced low-cost sensors do not have the capability to collect AoA (angle of arrival) data because it requires more expensive antenna arrays. They also do not have the capability to collect ToA (time of arrival) data because it requires the transmission of a wide-band known sequence [35], which is impossible in

the case of localizing (blind) intruders. Spectrum sensing platforms serve as the foundation of the spectrum applications, and transmitter localization is one of the main applications. Other applications include signal classification [93], spectrum anomaly detection [72], sensor selection [17, 16], spectral occupancy estimation [105], etc.

Transmitter localization. Localization of an intruder in a field using sensor observations has been widely studied, but most of the works have focused on localization of a single intruder [23, 43]. In general, to localize multiple intruders, the main challenge comes from the need to "separate" powers at the sensors [88], i.e., to divide the total received power into power received from individual intruders. Blind source separation is a very challenging problem; only very limited settings allow for known techniques using sophisticated receivers [107, 72]. We note that (indoor) localization of a device [10] based on signals received from multiple reference points (e.g, WiFi access points) is a quite different problem (see [131] for a recent survey), as the signals from reference points remain separate, and localization or tracking of multiple devices can be done independently. Recent works on multi-target localization/tracking such as [62] are different in the way that targets are passive, instead of active transmitters in the `MTL` problem. Among other related works, [49] addresses the challenge of handling time-skewed sensors observations in the MTL problem.

Wireless localization techniques mainly fall into two categories: geometry mapping and fingerprinting-based. Geometry mapping mainly has two subcategories: ranging-based such as trilateration (via RSS/RSSI, ToA, TDoA) and direction-based such as triangulation (via AoA). Fingerprinting-based methods can use all signal physical measurements including but not limited to amplitude, RSS/RSSI, ToA, TDoA, and AoA. Whenever deep learning is used for localization, it can be considered as a fingerprinting-based method, since it requires a training phase to survey the area of interest and a testing phase to search for (predict) the most likely location.

Deep learning for wireless localization. Quite a few recent works have harnessed the power of deep learning in the general topic of localization. E.g., DeepFi in [121] designs a restricted Bolzmann machine that localizes a single target using WiFi CSI amplitude data. DLoc in [7] uses WiFi CSI data as well. Its novelty is to transform CSI data into an image and then uses an image-to-image translation method to localize a single target. MonoDCell in [99] designs an LSTM that localizes a single target in indoor environment using cellular RSS data. [35] designs a three-layer neural network that locations a single transmitter. `DeepTxFinder` in [140] uses CNN to address the same `MTL` problem using RSS data in this chapter.

Transmitter Power Estimation. There are several works that estimate the transmission power of a single transmitter. [132] studies the "blind" estimation of transmission power based on received-power measurements at multiple cooperative sensor nodes using maximum likelihood estimation. Blind means there is no prior knowledge of the location of the transmitter or transmit power. [115] propose an iterative technique that jointly estimate the location and power of a single primary transmitter. In [65], the primary transmitter location and power is jointly estimated by a constrained optimization method. [136] uses the maximum likelihood estimation method to estimate the power of an isolated single transmitter and adopts an online learning method to estimate the power of multiple close by transmitters simultaneously.

3.9 Conclusion

In this chapter, we have designed and developed some novel deep-learning based scheme (`DeepMTL`) for the multiple transmitter localization (`MTL`) problem. We extended this problem to localizing the intruders in the presence of authorized users and developed a novel technique to solve it. We also developed a novel technique that can solve the multiple transmitter power estimation (`MTPE`) problem. Solving the general `MTL` and `MTPE` are both achieved by utilizing our robust `DeepMTL` as a building block. We evaluated all our methods extensively through data simulated from two propagation models as well as a small-scale data collected from a real world testbed. Our developed technique outperforms prior approaches by a significant margin in all performance metrics.

Acknowledgements

This work is supported by National Science Foundation grants CNS-1642965, CNS-1815306, and CNS-2128187. The authors would like to thank the reviewers for the valuable feedback and advice.

Chapter 4

Localize Transmitter in Quantum Sensor Networks

In this chapter, we present the first radio-frequency (RF) transmitter localization via a quantum sensor network and a quantum-based localization method. Localization of RF signals is a widely studied area in the past three decades. It has a great impact on human lives as it serves as the basis of a variety of location-based services. Meanwhile, quantum sensing is a new and interdisciplinary area that brings new opportunities to well-established problems.

We assume distributing some quantum sensors in an area that contains a single RF transmitter to be localized. We pose our transmitter localization problem as a quantum state discrimination problem and use the positive operator-valued measurement (POVM) as a tool for localization in a novel way. Evaluation results show that our proposed method achieves a high localization accuracy in the discrete case and a low localization error in the continuous case.

4.1 Introduction

Quantum sensors, being strongly sensitive to external disturbances, are able to measure various physical phenomena with extreme sensitivity. These quantum sensors interact with the environment and have the environment phenomenon or parameters encoded in their state [36]. A group of distributed quantum sensors, if prepared in an appropriate entangled state, can further enhance the estimation of a single continuous parameter, improving the standard deviation of measurement by a factor of $1/\sqrt{N}$ for N sensors [51]. Recently, experimental physicists successfully demonstrated a reconfigurable entangled radio-frequency photonic sensor network [127, 126]. The experiments establish a connection between the entanglement structure and the achievable quantum advantage in different distributed sensing problems.

In the classical world, a network of wireless sensors is well-known to facilitate spatially distributed sensing. For example, in the task of indoor RF localization, a transmitter signal's angle-of-arrival observed from different locations facilitates localization via triangulation [128]. RF localization is a key technology for location-based services. An improvement in RF localization will be very beneficial to an array of mobile applications and thus generate huge economic effects. For example, getting a fine-grained location in supermarkets, libraries, or museums will be very useful. Precise location is very important in augmented reality applications. For outdoor spectrum patrolling, we may want to detect and locate the intruders that illegally use unauthorized spectrum [23].

Motivation for Quantum Sensors. Albeit classical sensors are omnipresent, there are big motivations to explore quantum sensors. Quantum sensing is an emerging

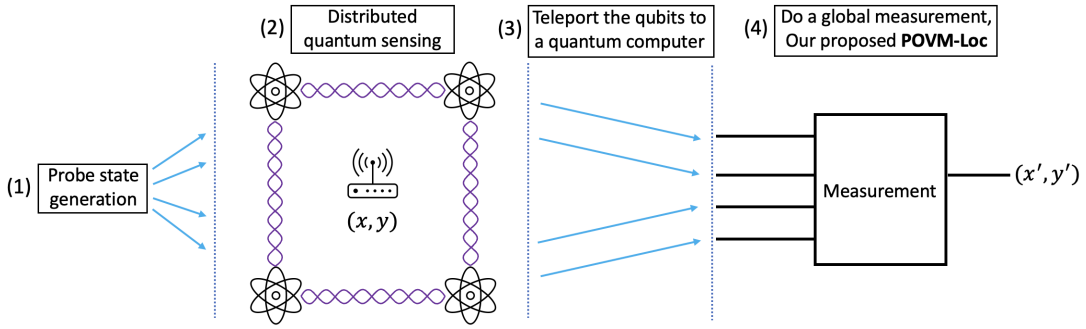


FIGURE 4.1: Overall architecture of localizing an RF transmitter in a quantum sensor network.

field that leverages the quantum properties of light and matter at atomic/subatomic scales and has the potential to sense signals at an unprecedented level of precision. For example, physicists in the year 2016 used squeezed quantum states to improve the sensitivity of the Laser Interferometer Gravitational-wave Observatory (LIGO) detector and successfully detected gravitational waves. In [127], researchers use some distributed quantum RF-photonic sensors to estimate the amplitude and phase of a radio signal. They showed the performance of sensing a global property of the RF wave is enhanced by leveraging a shared multipartite entangled state produced by squeezed light. In their experiments, the estimation variance of RF amplitude and phase both beat the standard quantum limit by over 3 dB. The precision improvement factor of $1/\sqrt{N}$ for N sensors is known as reaching the Heisenberg limit. Motivated by the above, we aim to leverage quantum sensors to perform some canonical tasks and thus open a new avenue of research.

Quantum localization on quantum data. The canonical task we picked is RF transmitter localization [128, 135]. We pose the localization problem as a quantum state discrimination problem [12]. The overall architecture is illustrated in Fig. 4.1. First, a probe state is generated and distributed to a network of quantum sensors. Second, the quantum sensors make interactions with the signal sent from a transmitter. The location of the transmitter is encoded in the quantum state at the quantum sensors. Third, the quantum states will be teleported [50] to a centralized node. Finally, at the central node, we consider global measurement where all the teleported qubits are measured simultaneously. The measurement outcome indicates the location of the transmitter. Quantum measurement via positive operator-valued measure (POVM) is the core part of quantum state discrimination. POVM is the general quantum measurement defined as a set of positive semi-definite Hermitian matrices $\{E_i\}$ and each Hermitian matrix E_i is associated with the measurement outcome i [86]. In our context, we further associate an outcome i to a two-dimensional location (x, y). Thus, we discriminate the quantum state via POVM and the POVM's output indicates the transmitter location.

Contributions. To the best of our knowledge, we are the first to explore using a distributed set of quantum sensors to localize an RF transmitter via a quantum localization approach. Our implementation is open source at [1]. In this context, we make the following three contributions.

1. Mathematically models a quantum sensor and introduces the problem of transmitter localization via a quantum sensor network. Section 4.2.

[1] https://github.com/caitaozhan/QuantumLocalization

2. Design and implement `POVM-Loc` and its improved version `POVM-Loc Pro`, two quantum localization methods using a quantum sensor network. Section 4.3.

3. Evaluate our techniques via simulations and demonstrate the effectiveness of our developed techniques. Section 4.4.

4.2 Sensor Model, Problem, Related Work

In this section, we describe our mathematical model of a quantum sensor and formulate the quantum transmitter localization problem. We also briefly talk about related work.

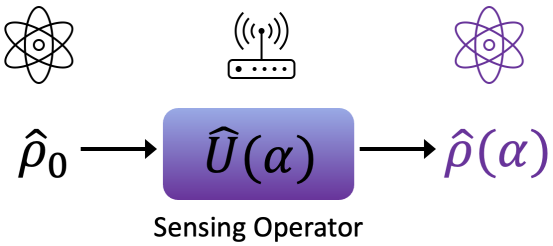


FIGURE 4.2: Sensing modeled as a single parameter estimation problem. The initial state $\hat{\rho}_0$ is used to probe an unknown parameter α embedded in the unitary operator $\hat{U}(\alpha)$, yielding output state $\hat{\rho}(\alpha)$.

The mathematical model of quantum sensors. We first formulate quantum sensing as a single parameter estimation problem, shown in Fig. 4.2. Consider a parameter α embedded in a unitary operator. To sense α, one prepares an initial quantum state $\hat{\rho}_0$ and passes it through unitary operator $\hat{U}(\alpha)$, obtaining the output state: $\hat{\rho}(\alpha) = \hat{U}(\alpha)\hat{\rho}_0\hat{U}^\dagger(\alpha)$. $\hat{U}(\alpha)$ describes the interaction between a qubit and the environment [126]

$$\hat{U}(\alpha) = e^{-i\alpha\hat{G}} \tag{4.1}$$

where $\hat{G}$ is called the generator, which is a Hermitian operator. $\hat{G}$ is determined by the type of α. The type includes quadrature displacement [127], phase shift [76] and qubit phase rotation, etc, [138]. In this chapter, we assume α is the *phase shift*, and our generator follows the expression in [76]

$$\hat{G} = \sigma_z/2 = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} \end{bmatrix} \tag{4.2}$$

We further assume that the phase shift α is a function of the distance between the transmitter and the sensor. This assumption is based on the observation that the closer the sensor is to the transmitter, the more affected the sensor is by the transmitter. If we can build a connection between the transmitter-receiver distance and some properties of the RF wave, we can model the $\hat{U}(\alpha)$ as a function of the RF wave. In this chapter, we pick received signal strength (RSS) as the bridge that connects the α and the RF wave, because RSS is easily accessible and is widely used in the fingerprinting-based wireless localization field. We have

$$\alpha = \frac{2\pi(RSS - NF)}{MP - NF} \tag{4.3}$$

where NF is the noise floor, MP is the maximum power a sensor can receive. We have $\alpha \in [0, 2\pi]$, since the received signal strength $RSS \in [NF, MP]$. Derived from Equation 4.1, 4.2, 4.3, the unitary operator at the sensor can be described as

$$\hat{U}(\alpha) = \begin{bmatrix} e^{-i\frac{\pi(RSS-NF)}{MP-NF}} & 0 \\ 0 & e^{i\frac{\pi(RSS-NF)}{MP-NF}} \end{bmatrix} \tag{4.4}$$

The Equation 4.4 is interpreted as follows: A quantum sensor is affected by the signal sent by a transmitter. The transmitter's signal triggers a phase shift in the quantum sensor and a certain value of phase shift is a function of RSS. For the RSS model, we use the log-distance model

$$RSS(d) = P_0 - 10\beta log_{10}(d) + \mathcal{X} \tag{4.5}$$

where P_0 is the reference power a sensor receives at 1 meter away from the transmitter, d is the transmitter-receiver distance, β is the path-loss exponent, and $\mathcal{X}$ is the shadowing effect that can be represented by a zero-mean Gaussian distribution.

After modeling a single sensor, let us model multiple sensors in a distributed sensing scenario [138]. We assume N unknown parameters at N quantum sensors respectively. The combined unitary operator is a *tensor product* of N individual unitary operators, $\hat{U}(\boldsymbol{\alpha}) = \bigotimes_{n=1}^{N} \hat{U}(\alpha_n)$. To carry out sensing, one inputs N probes in a quantum state $\hat{\boldsymbol{\rho}}_{\mathbf{0}}$ and obtains the output quantum state, $\hat{\boldsymbol{\rho}}_N(\boldsymbol{\alpha}) = \hat{U}(\boldsymbol{\alpha})\hat{\boldsymbol{\rho}}_0\hat{U}^{\dagger}(\boldsymbol{\alpha})$, where the unknown parameters are $\boldsymbol{\alpha} = \{\alpha_n\}_{n=1}^{N}$, and each α_n represents an unknown parameter at the nth sensor.

Problem Setting and Formal Definition. Consider a network of quantum sensors distributed in a geographic area and a single transmitter active in the area. We assume each quantum sensor holds one qubit. For simplicity, we assume all qubits are in pure states, i.e., $\hat{\rho}_0 = |\psi_0\rangle\langle\psi_0|$ and $|\psi_\alpha\rangle = \hat{U}(\alpha)|\psi_0\rangle$. Assume the initial state of the quantum sensor network is $|\boldsymbol{\psi_0}\rangle$. After the interaction with the environment, the quantum state evolves to $|\boldsymbol{\psi_\alpha}\rangle = \hat{U}(\boldsymbol{\alpha})|\boldsymbol{\psi_0}\rangle$. Our problem is to determine the location of the transmitter, given the quantum state $|\boldsymbol{\psi_\alpha}\rangle$ reported by the quantum sensor network.

Related Work. RF localization has been an active area of research for three decades. One of the pioneering works is RADAR [10]. RADAR is a fingerprinting-based method that has two phases. In the training phase, a person holds a receiver and travels in an area, meanwhile RADAR records information about the radio signal as a function of the location, i.e., constructs a received signal strength (RSS) fingerprinting map. In the localization phase, RADAR conducts the nearest neighbor search and reports the location of the closest signal on the fingerprinting map. The premise is that RSS information provides a means of inferring RF location. Horus [130] improves the localization accuracy by using a probabilistic method. The target to be localized can either be a transmitter [58], a sensor [10], or neither both (device-free localization [89]). In this work, the target we localize is a single transmitter. For the simultaneous localization of multiple transmitters, see [135].

All the localization works above are classical, i.e., they use classical wireless sensors and classical localization methods. The first localization work that brings in quantum is [110]. It uses quantum amplitude encoding [108] that embeds an RSS vector of length 2^N into the amplitudes of an N qubit quantum state. Thus, in theory, it brings exponential enhancement in space complexity. Then, it uses a quantum fingerprinting method based on the quantum cosine similarity algorithm [20] to match a signal to be

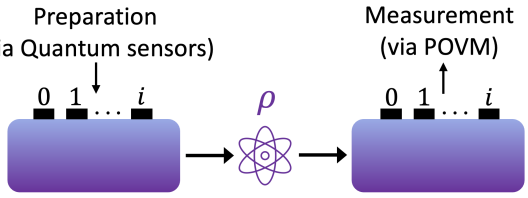


FIGURE 4.3: Illustration of Quantum state discrimination. There are two parties: Alice for preparing and sending a quantum state, and Bob for measuring the received quantum state from Alice.

localized to a signal recorded in the fingerprinting map. The work leverages quantum computing and the power of quantum superposition, but they still use classical wireless sensors that collect classical data.

4.3 Methodology and Our Approach

Quantum State Discrimination for Localization. Our methodology is to pose the localization problem as a quantum state discrimination problem. Quantum state discrimination underlies various applications in quantum information processing tasks [12]. It essentially describes the distinguishability of quantum systems in different states, and the general process of extracting classical information from quantum states [9]. Fig. 4.3 illustrates the process of quantum state discrimination, where one party (Alice) prepares a quantum state, and another party (Bob) measures the received quantum state sent by Alice. In our context of localization, the *quantum sensor network plays the role of Alice*. And a *central node that performs global measurement plays the role of Bob*. The problem of discriminating among non-orthogonal quantum states boils down to a measurement optimization problem, i.e., optimizing the POVM. Mathematically, a POVM is a set of positive semi-definite Hermitian matrices $\{E_i\}$ on a Hilbert space $\mathcal{H}$ that sum to the identity matrix, i.e., $\sum_i E_i = I$. In quantum mechanics, the POVM element $\{E_i\}$ is associated with the measurement outcome i, such that the probability of obtaining it when making a measurement on the quantum state ρ is

$$Prob(i) = tr(\rho E_i) = tr(|\psi\rangle \langle\psi| E_i) \tag{4.6}$$

where tr is the trace operator, and $\rho = |\psi\rangle \langle\psi|$ for pure states. If we further associate an outcome i with a 2D location (x, y), then we have a location estimator. For example, we can associate measurement outcomes $i = \{0, 1, 2, 3\}$ to locations $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$ respectively. So a POVM outputting 1 indicates the transmitter is at location $(0, 1)$.

Optimizing the POVM is governed by some metrics, such as the minimum error discrimination metric and the unambiguous discrimination metric [13]. Semi-definite programming [44] is a general numerical method that can compute the optimal POVM under various metrics. In our paper, we choose a simple but good enough method to compute a POVM, the pretty good measurement (PGM) [57]

$$E_i = p_i \hat{\rho}^{-1/2} \hat{\rho}_i \hat{\rho}^{-1/2} \tag{4.7}$$

where p_i is the prior probability (we assume equal prior for simplicity) for the ith quantum state $\hat{\rho}_i$, and $\hat{\rho} = \sum_i p_i \hat{\rho}_i$. We simulate $\hat{\rho}_i$ through our $\hat{U}(\alpha)$ model as if the

transmitter is indeed at the location associated with i and putting the initial probe state $\hat{\boldsymbol{\rho}}_{\mathbf{0}}$ in a uniform superposition state.

Challenges. If we have an appropriate POVM $\{E_i\}$, the localization problem is solved. But designing the appropriate $\{E_i\}$ has challenges. 1) The first challenge arrives in localization accuracy. Consider 256 discrete locations in a 16×16 grid. We can design a POVM that has 256 elements, indicating 256 output locations. But we observe that a POVM having too many outputs will lead to lower accuracy. Because when the number of quantum states is high, the difference between each state is relatively smaller (i.e., the overlap between quantum states is relatively larger). Thus being more difficult to discriminate. So we want to avoid a POVM that has hundreds of elements. 2) The second challenge is that if we do a global measurement on a relatively high number of sensors, the dimension of the Hilbert space will be large. It results in POVM elements being too costly in RAM during computation. Take an example of doing a global measurement on 12 sensors. Each POVM element will be a complex number matrix of dimension $2^{12} \times 2^{12}$. Each complex number needs 16 bytes. A POVM of 256 elements will cost 69 GB of RAM. In reality, the physical implementation of POVM is costly as well [129]. So we want to avoid doing a global POVM on tens of sensors. But if we actually do have tens of quantum sensors, not being able to use all of them is a waste of resources.

Our approach `POVM-Loc`. A quantum counterpart of the fingerprinting-based approach has a training phase and a localization phase. `POVM-Loc` is our approach for the localization phase. The names come from the critical role POVM plays in the approach. The two challenges mentioned in the previous paragraph can be summarized as the scalability challenge. In this section, `POVM-Loc` solves the scalability challenge that 1) it has two levels of POVM and each level's POVM needs fewer elements, and 2) utilizes all the given quantum sensors in a novel way that each POVM does a global measurement on fewer number of sensors. Before describing `POVM-Loc`, we define some concepts.

- Grid, block, and cell. We discretize the area into a grid; a cell is the smallest unit in a grid. A block is a group of cells. For example, a block in Fig. 4.4 (a) is comprised of 2×2 cells in Fig. 4.4 (b).

- Coarse level and fine level. A coarse level is a level comprised of blocks and a fine level is a level comprised of cells. Fig. 4.4 (a) shows a coarse level with 4 blocks and Fig. 4.4 (b) shows a fine level with 16 cells.

- Coarse-level sensors and fine-level sensors. A set of coarse-level sensors collects quantum data ρ_i to determine the block where the transmitter is in. A set of fine-level sensors collects quantum data ρ_i to determine the cell where the transmitter is in. A set of coarse-level sensors is associated with the grid since it covers the whole grid. A set of fine-level sensors is associated with a block since it only covers a block. Fig. 4.4 (a) has one set of coarse-level sensors, and Fig. 4.4 (b) has four sets of fine-level sensors that cover four blocks (in four colors). A sensor at the borders of two blocks covers both blocks.

- Coarse-level POVM and fine-level POVM. Take $\{\rho_i\}$ with equal priors and compute the POVM $\{E_i\}$ using PGM. During the training phase, if $\{\rho_i\}$ are collected by a set of coarse-level sensors and the transmitter fingerprints at the center of the blocks, we get a coarse-level POVM $\{cE_i\}$. Likewise, if the $\{\rho_i\}$ are collected by a set of fine-level sensors and the transmitter fingerprints at the center of the

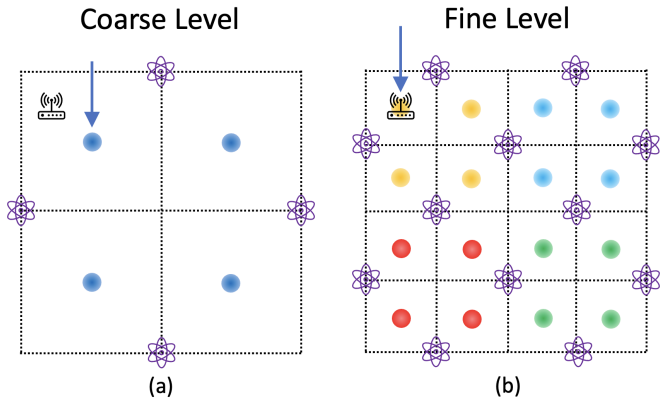


FIGURE 4.4: Illustration of `POVM-Loc`, a multi-level POVM scheme for transmitter localization. The coarse level POVM determines the block (a), and the fine level POVM determines the cell (b).

cell of the associated block, we get fine-level POVM $\{fE_i\}$. Fig. 4.4 (a) shows one coarse-level POVM and Fig. 4.4 (b) shows four fine-level POVMs. The dots in Fig. 4.4 indicates the center of a block or cell.

The training phase. We discretize an area into a $N \times N$ grid. Without loss of generality, we assume $\sqrt{N}$ is an integer. We divide the grid into $\sqrt{N}$ row $\sqrt{N}$ column of blocks whose size is $\sqrt{N} \times \sqrt{N}$. The sensors will be randomly spread out, ideally close to uniform to better cover the whole area. For the grid, there will be a set of coarse-level sensors associated with the grid, and the coarse-level POVM is computed by quantum data collected from the coarse-level sensors. For each block, there will be a small number of sensors associated with the block. Each block's fine-level POVM is computed by quantum data collected from the block's associated fine-level sensors.

Algorithm 1: `POVM-Loc`

Input: $\{cE_i\}$ – one coarse-level POVM
Input: $[\{fE_i^0\}, \{fE_i^1\}, \cdots]$ – an array of fine-level POVMs
Output: location (x, y)

$repeat \leftarrow 1000$;
$j \leftarrow \text{SenseMeasure}(\{cE_i\}, repeat)$;
$block_j \leftarrow$ the block associated with j ;
$\{fE_i\} \leftarrow$ the fine-level POVM associated with $block_j$;
$j \leftarrow \text{SenseMeasure}(\{fE_i\}, repeat)$;
$cell_j \leftarrow$ the cell associated with j ;
return the location (x, y) of $cell_j$;

The localization phase. At a higher level, `POVM-Loc` localizes a transmitter from a coarser level to a finer level. Without any prior information, a transmitter could be anywhere in the area. The coarse-level POVM's output can decrease the scope of the area from the whole grid to a block. Then inside a block, a fine-level POVM's output will determine where the cell the transmitter is in. Algorithm 1 is the pseudo-code of `POVM-Loc`. Algorithm 1 relies on Procedure 1 that repeats the process of state preparation via a quantum sensor network and POVM measurement and returns the

Procedure 1: SenseMeasure($\{E_i\}$, K)

Input: $\{E_i\}$ – a POVM
Input: K – number of repetition
Output: j – the most frequent measurement outcome

$count \leftarrow$ a key-value pair dictionary;
$qsensors \leftarrow$ a set of quantum sensors associated with the POVM $\{E_i\}$;
for $k = 1 \cdots K$ **do**
    $\rho \leftarrow$ a quantum state sensed by $qsensors$;
    $i \leftarrow$ outcome of measuring ρ via POVM $\{E_i\}$;
    $count[i] = count[i] + 1$;
end
$j \leftarrow arg\,max_{\{i\}}\ count[i]$;
return j ;

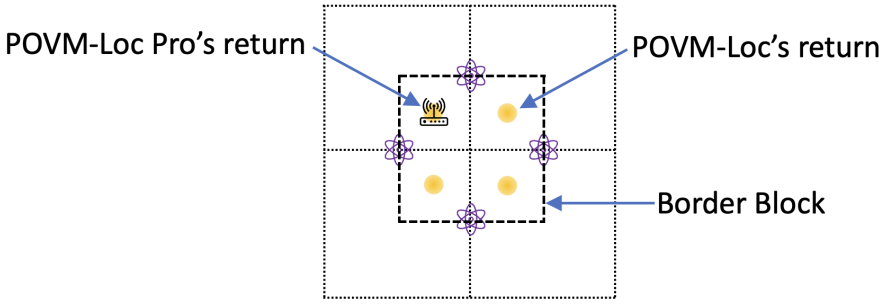


FIGURE 4.5: Illustration the idea of `POVM-Loc Pro`. There are four blocks and one border block in the middle. If `POVM-Loc` returns a cell that is at the edge of a block, do another fine-level POVM associated with the border block.

most frequent measurement output. Procedure 1 can be viewed as the quantum sensing protocol [36] in our localization context.

A further improvement `POVM-Loc Pro`**.** Through the two-level scheme, `POVM-Loc` avoids doing a global measurement on a large number of quantum sensors and having a large number of elements in a POVM, thus being scalable. Through evaluation, the scheme works quite well. However, we observe that the cells at the edge of the blocks have a higher chance of error in localization. Another observation is that the wrong output cell of `POVM-Loc` is usually nearby to the correct cell, like the other side of the border. To mitigate this kind of error, we put a new block that covers the border of the blocks. These new blocks are called border blocks and each border block will be associated with a set of fine-level sensors (previous fine-level sensors can be reused). So, whenever `POVM-Loc` returns a cell at the border of blocks, we find the border block where the cell is inside and do a second fine-level POVM associated with the border block. Then the outcome of the second fine-level POVM is the final return. This scheme of `POVM-Loc` plus an optional second fine-level POVM is named `POVM-Loc Pro`. See Fig. 4.5.

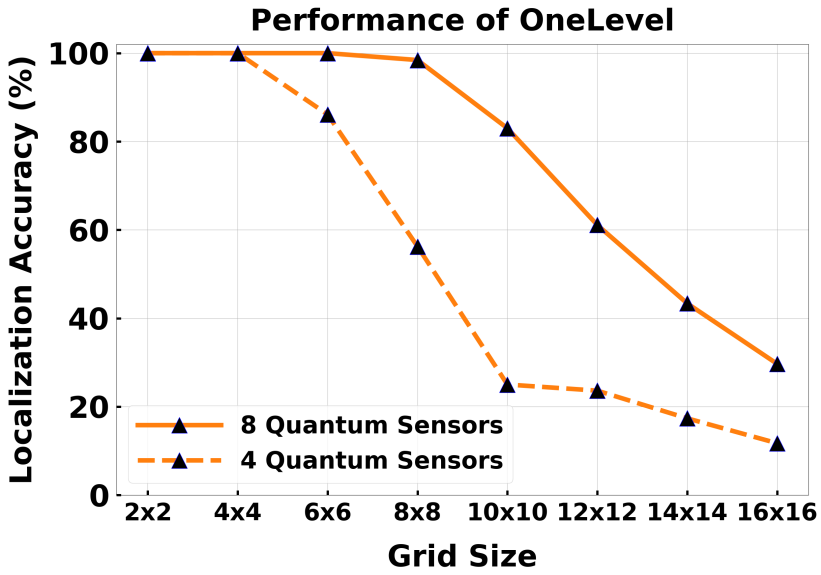


FIGURE 4.6: The performance of `OneLevel` against varying grid size.

4.4 Evaluation

In this section, first, we show the limitation of our baseline method, i.e., a single-level POVM-based method named `OneLevel`. Then we evaluate the effectiveness of `POVM-Loc` and `POVM-Loc Pro` and show that their performance is superior to `OneLevel`.

Performance Metrics. We use the following two metrics to evaluate the localization methods.

1. Localization accuracy (L_{acc})
2. Localization error (L_{err})

L_{acc} is used for the discrete location setting, where we confine the location of the transmitters during both the training and localization phases to the center of the cells. L_{err} is used for the general continuous location setting, where the transmitters during the localization phase can be anywhere (the transmitters during the training phase are still at the cell center). Therefore, L_{acc} is a percentage similar to the classification accuracy, and L_{err} is the distance between the ground truth location and the method's output location measured in meters.

Experiment Settings. We perform experiments in different grid sizes, while the area of a cell represents stays at $10m \times 10m$. Our wireless propagation model is the log-distance model in Equation 4.5. Specifically, the reference power $P_0 = -10dBm$, and path-loss exponent $\beta = 3.5$. For the zero-mean normal distribution shadowing effect, the default standard deviation is 1. The shadowing effect is considered the source of noise in our evaluation. In the training phase, the POVM $\{E_i\}$ is computed with data that doesn't contain noise. In the localization phase, each testing sample (a quantum state reported from quantum sensors) contains noise. For the location of sensors, we assume the sensors are uniformly spread out to cover the whole area better.

The limitation of `OneLevel`. Fig. 4.6 shows that the localization accuracy of `OneLevel` decreases significantly when the grid size increases. A larger grid size leads to a larger

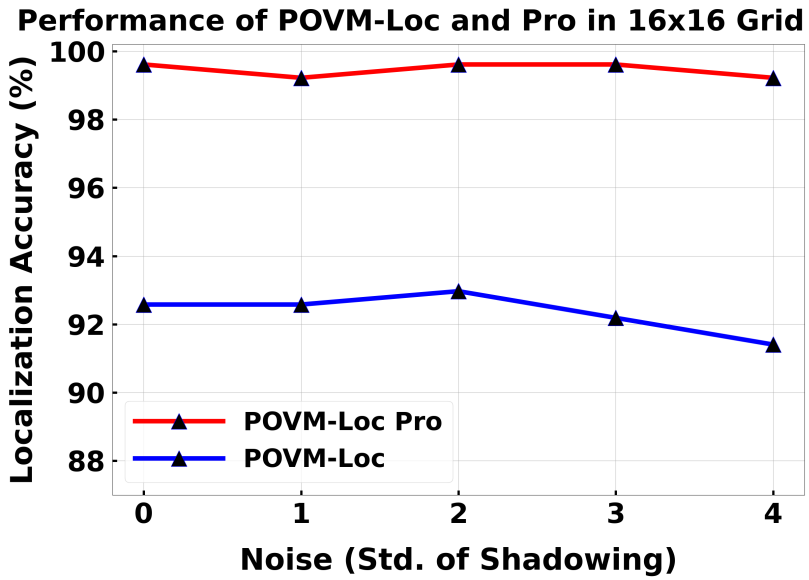


FIGURE 4.7: The performance of `POVM-Loc` and `POVM-Loc Pro` in 16×16 grid against varying noise represented by the standard deviation in the shadowing effect (see Equation 4.5).

number of quantum states being discriminated. This leads to a larger overlap between the quantum states, making the quantum states harder to discriminate. Fig. 4.6 also shows that by using 8 quantum sensors, the localization accuracy is higher compared with 4 quantum sensors. This is because more quantum sensors cover the area better so that the quantum state is able to encode more information about the environment. More information is encoded in a larger Hilbert space, so the overlap between the quantum states becomes smaller, making the quantum states easier to discriminate. But the high accuracy brought by a higher number of sensors comes at a cost of higher RAM and runtime, see Table 4.1. `OneLevel`'s limitation is exactly the challenges mentioned in Section 4.3.

TABLE 4.1: Runtime (s) for `OneLevel` using 4 and 8 quantum sensors against varying grid size

	2x2	4x4	6x6	8x8	12x12	16x16
4 Sensors	1.0	1.1	1.2	1.4	2.0	2.6
8 Sensors	4.8	9.1	27.4	53.1	113.4	193.8

TABLE 4.2: Runtime (s) for three methods in 16×16 grid

`OneLevel`	`POVM-Loc`	`POVM-Loc Pro`
193.8	10.3	11.3

The Improvement of `POVM-Loc` and `POVM-Loc Pro`. `POVM-Loc` and `POVM-Loc Pro` are evaluated in a 16×16 grid that is divided into 4 rows and 4 columns of blocks where each block's size is 4×4. There are another 21 border blocks for `POVM-Loc Pro`. There are 8 coarse-level quantum sensors and 40 fine-level quantum sensors. At the fine level, each block will be covered by 4 fine-level sensors. A sensor at the border of blocks can cover either block to reduce the total sensor used. `POVM-Loc` and `POVM-Loc Pro`'s improvement of localization accuracy compared with `OneLevel` is major. Fig. 4.7

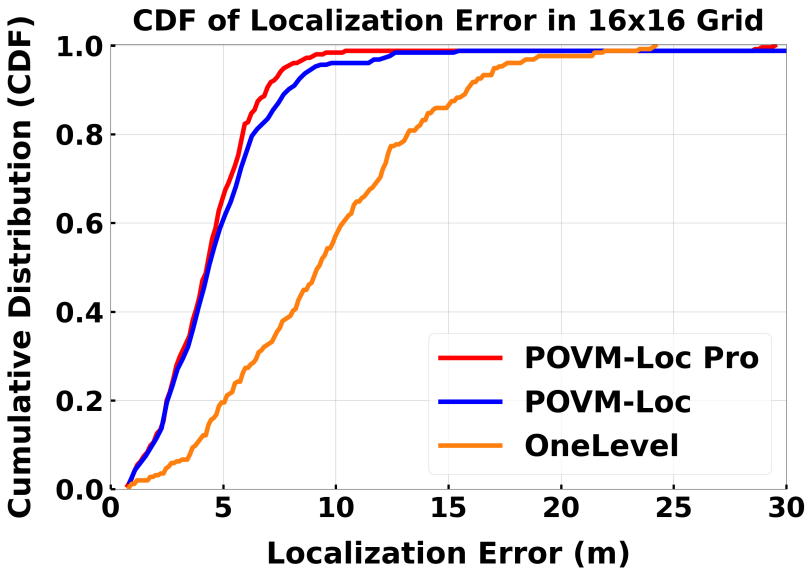


FIGURE 4.8: The cumulative probability of localization of `POVM-Loc`, `POVM-Loc Pro`, `OneLevel` in a continuous and 16×16 grid setting.

shows the localization accuracy of `POVM-Loc` and `POVM-Loc Pro` at noise equals 1 is 92.5% and 99.2% respectively. At the same noise level, the localization accuracy of `OneLevel` is only 11.7% and 29.7% using 4 and 8 quantum sensors respectively. Although the total number of quantum sensors deployed for `POVM-Loc` and `POVM-Loc Pro` is a lot higher than `OneLevel`, localizing a single instance costs only a small number of extra sensors. `POVM-Loc` uses 12 sensors (8 coarse, 4 fine) and `POVM-Loc Pro` uses a couple more. Table 4.2 shows that in simulation, `POVM-Loc` and `POVM-Loc Pro` are a lot faster than `OneLevel` in the 16×16 grid test case. `POVM-Loc Pro`'s additional step costs 1 second more compared with `POVM-Loc`. Fig. 4.7 also shows that `POVM-Loc` and `POVM-Loc Pro` are robust against noise, i.e., when the noise level increases, the performance barely drops. This robustness is due to the one thousand repetitions in Procedure 1. For the general case of transmitter location in the continuous domain, the cumulative distribution of localization error in Fig. 4.8 shows `POVM-Loc` and `POVM-Loc Pro` also outperforms `OneLevel` by a large margin. While the error under 60% cumulative is the same, `POVM-Loc Pro` is better than `POVM-Loc` in the above 60% cumulative range.

4.5 Conclusion and Future Work

We introduce `POVM-Loc` and an improved version `POVM-Loc Pro` that localizes an RF transmitter in a sensor network effectively and efficiently. The approach utilizes quantum data obtained from a quantum sensor network and performs POVM measurements in a creative way to determine the transmitter location. For future work, we would like to explore performing quantum measurements on a computational basis combined with quantum machine learning. We would also like to explore more unitary operator models, as well as whether an entangled initial state's can enhance the localization performance.

Acknowledgment

Thanks to the grant NSF-2106447 for supporting this paper.

Chapter 5

Efficient Quantum Network Communication using Balenced Entanglement Swapping Trees

Quantum network communication is challenging, as the No-cloning theorem in quantum regime makes many classical techniques inapplicable; in particular, direct transmission of qubit states over long distances is infeasible due to unrecoverable errors. For long-distance communication of unknown quantum states, the only viable communication approach (assuming local operations and classical communications) is teleportation of quantum states, which requires a prior distribution of entangled pairs (EPs) of qubits. Establishment of EPs across remote nodes can incur significant latency due to the low probability of success of the underlying physical processes. The focus of this chapter is to develop efficient techniques that minimize EP generation latency. Prior works have focused on selecting entanglement *paths*; in contrast, we select *entanglement swapping trees*—a more accurate representation of the entanglement generation structure. Prior work [50] developed a dynamic programming algorithm to select an optimal swapping-tree for a single pair of nodes, under the given capacity and fidelity constraints. For the general setting, [50] also developed an efficient iterative algorithm to compute a set of swapping trees. However, the dynamic programming algorithm has a high time complexity, and thus, may not be practical for real-time route finding in large networks. In this chapter, we focus on developing an *almost linear time* heuristic for the `QNR-SP` problem, based on the classic Dijkstra shorted path algorithm. The designed heuristic performs close to the DP-based algorithms in our empirical studies.

5.1 Introduction

Fundamental advances in physical sciences and engineering have led to the realization of working quantum computers (QCs) [6, 48]. However, there are significant limitations to the capacity of individual QC [22]. Quantum networks (QNs) enable the construction of large, robust, and more capable quantum computing platforms by connecting smaller QCs. Quantum networks [111] also enable various important applications [45, 106, 67, 29, 80]. However, quantum network communication is challenging — e.g., physical transmission of quantum states across nodes can incur irreparable communication errors, as the No-cloning Theorem [40] proscribes making independent copies of arbitrary qubits. At the same time, certain aspects unique to the quantum regime, such as entangled states, enables novel mechanisms for communication. In particular, teleportation [11] transfers quantum states with just classical communication, but requires an

a priori establishment of entangled pairs (EPs). This chpater presents techniques for efficient establishment of EPs in a network.

Establishment of EPs over long distances is challenging. Coordinated entanglement swapping (e.g. DLCZ protocol [41]) using quantum repeaters can be used to establish long-distance entanglements from short-distance entanglements. However, due to low probability of success of the underlying physical processes (short-distance entanglements and swappings), EP generation can incur significant latency— of the order of 10s to 100s of seconds between nodes 100s of kms away [104]. Thus, we need to develop techniques that can facilitate fast generation of long-distance EPs. [50] sovles the `QNR-SP` Problem: Given a single (s, d) pair, select a minimum-latency swapping tree under given constraints. In this chapter, we select near-optimal swapping trees by a heuristic at a much lower time complexity.

To the best of our knowledge, no prior work has addressed the problem of selecting an efficient swapping-tree for entanglement routing; they all consider selecting routing *paths* ([21] selects a path using a metric based on balanced trees; see §5.3.2). Almost all prior works have considered the "waitless" model, wherein all underlying physical processes much succeed *near-simultaneously* for an EP to be generated; this model incurs minimal decoherence, but yields very low EP generation rates. In contrast, we consider the "waiting" protocol, wherein, at each swap operation, the earlier arriving EP waits for a limited time for the other EP to be generated. Such an approach with efficient swapping trees yields high entanglement rates; the potential decoherence risk can be handled by discarding qubits that "age" beyond a certain threshold.

Our Contributions. We formulate the entanglement routing problem (§5.3) in QNs in terms of selecting optimal swapping *trees* in the "waiting" protocol, under fidelity constraints. In this context, we make the following contribution:

1. For the `QNR-SP` problem, the optimal algorithm in [50] has high time complexity; we aim to improve the time-complexity of the algorithm without degrading its empirical performance. We thus design a near-linear time heuristic for the `QNR-SP` problem based on an appropriate metric which essentially restricts the solutions to balanced swapping trees (§5.4).

5.2 QC Background

Qubit States. Quantum computation manipulates *qubits* analogous to how classical computation manipulates *bits*. At any given time, a bit may be in one of two states, traditionally represented by 0 and 1. A quantum state represented by a *qubit* is a *superposition* of classical states, and is usually written as $\alpha_0 |0\rangle + \alpha_1 |1\rangle$, where α_0 and α_1 are *amplitudes* represented by complex numbers and such that $\|\alpha_0\|^2 + \|\alpha_1\|^2 = 1$. Here, $|0\rangle$ and $|1\rangle$ are the standard (orthonormal) *basis* states; concretely, they may represent physical properties such as spin (down/up), polarization, charge direction, etc. When a qubit such as above is *measured*, it collapses to a $|0\rangle$ state with a probability of $\|\alpha_0\|^2$ and to a $|1\rangle$ state with a probability of $\|\alpha_1\|^2$. In general, a state of an n qubit system can be represented as $\Sigma_{i=0}^{2^n-1} \alpha_i |i\rangle$ where "i" in $|i\rangle$ is i's bit representation.

Entanglement. Entangled pure[1] states are multi-qubit states that cannot be "factorized" into independent single-qubit states. E.g., the 2-qubit state $\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$; this

[1] In this chapter, we largely deal with only pure qubit states. Entanglement of general mixed states is defined in terms of separation of density matrices [55].

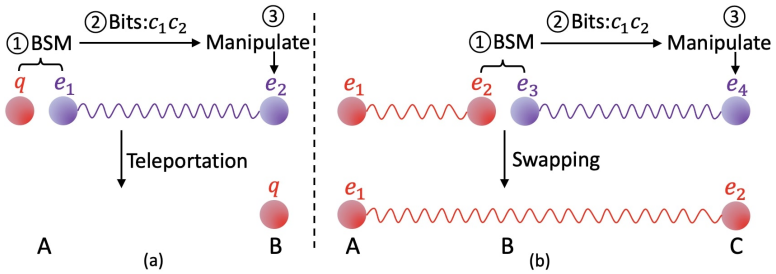


FIGURE 5.1: (a) Teleportation of $|q\rangle$ from A to B, while consuming an entangled pair (e_1, e_2). (b) Entanglement swapping over the triplet of nodes (A, B, C), which results in A's qubit entangled with C's qubit. This can be viewed as a teleportation of e_2 from node B to C.

particular system is a *maximally-entangled* state. We refer to maximally-entangled pairs of qubits as EPs. The surprising aspect of entangled states is that the combined system continues to stay entangled, even when the individual qubits are physically separated by large distances. This facilitates many applications, e.g., teleportation of qubit states by local operations and classical information exchange, as described next.

Teleportation. Direct transmission of quantum data is subject to unrecoverable errors, as classical procedures such as amplified signals or re-transmission cannot be applied due to quantum no-cloning [123, 40].[2] An alternative mechanism for quantum communication is *teleportation*, Fig. 5.1 (a), where a qubit q from a node A is recreated in another node B (*while "destroying" the original qubit* q) using only classical communication. However, this process requires that an EP already established over the nodes A and B. Teleportation can thus be used to reliably transfer quantum information. At a high-level, the process of teleporting an arbitrary qubit, say qubit q, from node A to node B can be summarized as follows:

1. an EP pair (e_1, e_2) is generated over A and B, with e_1 stored at A and e_2 stored at B;

2. at A, a *Bell-state measurement* (BSM) operation over e_1 and q is performed, and the 2 classical bits measurement output (c_1c_2) is sent to B through the classical communication channel; at this point, the qubits q and e_1 at A are destroyed.

3. manipulating the EP-pair qubit e_2 at B based on received (c_1, c_2) changes its state to q's initial state.

Depending on the physical realization of qubits and the BSM operation, it may not always be possible to successfully generate the 2 classical bits, as the BSM operation is stochastic.

Entanglement Swapping (ES). Entanglement swapping is an application of teleportation to generate EPs over remote nodes. See Fig. 5.1 (b). If A and B share an EP and B teleports its qubit to C, then A and C end up sharing an EP. More elaborately, let us assume that A and B share an EP, and B and C share a separate EP. Now, B performs a BSM on its two qubits and communicates the result to C (teleporting its qubit that is entangled with A to C). When C finishes the protocol, it has a qubit that is entangled with A's qubit. Thus, an entanglement swapping (ES) operation can

[2] Quantum error correction mechanisms [82, 38] can be used to mitigate the transmission errors, but their implementation is very challenging and is not expected to be used until later generations of quantum networks.

be looked up as being performed over a triplet of nodes (A, B, C) with EP available at the two pairs of adjacent nodes (A, B) and (B, C); it results in an EP over the pair of nodes (A, C).

Fidelity: Decoherence and Operations-Driven. Fidelity is a measure of how close a realized state is to the ideal. Fidelity of qubit decreases with time, due to interaction with the environment, as well as gate operations (e.g., in ES). Time-driven fidelity degradation is called *decoherence*. To bound decoherence, we limit the aggregate time a qubit spends in a quantum memory before being consumed. With regards to operation-driven fidelity degradation, Briegel et al. [18] give an expression that relates the fidelity of an EP generated by ES to the fidelities of the operands, in terms of the noise introduced by swap operations and the number of link EPs used. The order of the swap operations (i.e., the structure of the swapping tree) does not affect the fidelity. Thus, the operation-driven fidelity degradation of the final EP generated by a swapping-tree T can be controlled by limiting the number of leaves of T, assuming that the link EPs have uniform fidelity (as in [25]).

Entanglement Purification [18, e.g.] and Quantum Error Correction [100, e.g.] have been widely used to combat fidelity degradation. Our work focuses on optimally scheduling ES operations with constraints on fidelity degradation, without purification or error correction.

Quantum Memories. Multiple quantum memories have been recently proposed to bring quantum networks into realization. Types of quantum memories that support BSM measurements and gate unitary operations, and probably have a long decoherence time can be used in quantum communications. Most of them are matter-based which have photonic interface to produce matter-matter entanglement over two neighboring nodes (see below). At a high-level, there are three different quantum memory platforms: quantum dots, trapped atoms or ions, and colour centers in diamond. Each has its own physical characteristics and applications. While quantum dots have the ability to process quantum information very fast, they exhibit a very low decoherence time among others [92, 118]. To overcome the low efficiency of single atom-photon coupling process, atomic ensemble schemes have been proposed [41] where along with dynamic decoupling and cooling techniques, decoherence times of a few seconds have been achieved [103, 117, 37]. For trapped ion memories, decoherence times from several minutes to few hours have been demonstrated [70, 119]. To further increase the entanglement generation rate, [14] proposes a way to use a single silicon–vacancy (SiV) colour center in diamond to perform asynchronous photonic BSM at the node located in the middle of two adjacent quantum nodes.

5.2.1 Generating Entanglement Pairs (EPs)

As described above, teleportation, which is the only viable means of transferring quantum states over long distances, requires an a priori distribution of EPs. Thus, we need efficient mechanisms to establish EPs across remote QN nodes; this is the goal of our work. Below, we start with describing how EPs are generated between adjacent (i.e., one-hop away) nodes, and then discuss how EPs across a pair of remote nodes can be established via ESs.

Generating EP over Adjacent Nodes. The physical realization of qubits determines the technique used for sharing EPs between adjacent nodes. The *heralded entanglement* process [109, 21] to generate an atom-atom EP between adjacent nodes A and B is as follows:

1. Generate an entangled pair of atom and a telecom-wavelength photon at node A and B. Qubits at each node are generally realized in an atomic form for longer-term storage, while photonic qubits are used for transmission.

2. Once an atom-photon entanglement is locally generated at each node (at the same time), the telecom-photons are then transmitted over an optical fiber to a photon-photon/optical BSM device C located in the middle of A and B so that the photons arrive at C at the same time.

3. The device C performs a BSM over the photons, and transmits the classical result to A or B to complete ES.

Other entanglement generation processes have been proposed [81]; our techniques themselves are independent of how the link EP are generated.

Generating EP between Remote Nodes. Now, EP between non-adjacent nodes connected by a path in the network can be established by performing a sequence of ESs at intermediate nodes; this requires an a priori EP over each of the adjacent pairs of nodes in the path. For example, consider a path of nodes $x_0, x_1, x_2, x_3, x_4, x_5$, with an EP between every pair of adjacent nodes (x_i, x_{i+1}). Thus, each node x_i $(1 \leq i \leq 4)$ has two qubits, one of which is entangled with x_{i-1} and the other with x_{i+1}. Nodes x_0 and x_5 have only one qubit each. To establish an EP between x_0 and x_5, we can perform a sequence of entanglement swappings (ESs) as shown in Fig. 5.2. Similarly, the sequence of ES over the following triplets would also work: (x_2, x_3, x_4), (x_2, x_4, x_5), (x_0, x_1, x_2), (x_0, x_2, x_5).

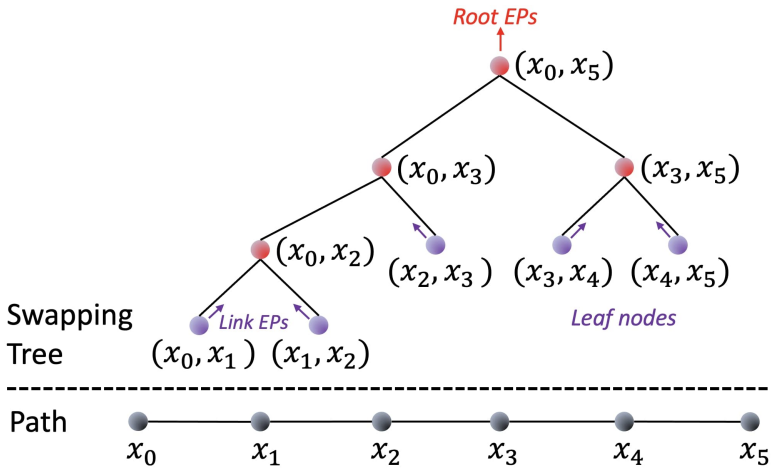


FIGURE 5.2: A swapping tree over a path. The leaves of the tree are the path-links, which generate link-EPs continuously.

Swapping Trees. In general, given a path $P = s \rightsquigarrow d$ from s to d, any complete binary tree (called a *swapping tree*) over the ordered links in P gives a way to generate an EP over (s, d). Each vertex in the tree corresponds to a pair of network nodes in P, with each leaf representing a link in P. Every pair of siblings (A, B) and (B, C) perform an ES over (A, B, C) to yield an EP over (A, C)—their parent. See Fig. 5.2. *Note that subtrees of a swapping tree execute in parallel.* Different swapping trees over the same path P can have different performance characteristics, as discussed later (see Fig. 5.4).

Expected Generation Latency/Rate of EPs. In general, our goal is to continuously generate EPs at some rate using a swapping tree, using continuously generated EPs at the leaves. The stochastic nature of ES operations means that an EP at the tree's root will be successfully generated only after many failed attempts and hence significant latency. We refer to this latency as the *generation latency* of the EP at the root, and in short, just the generation latency of the tree. EP generation rate is the inverse of its generation latency. Whenever we refer to generation latency/rate, we implicitly mean *expected* generation latency/rate.

Two Generation Protocols: `WaitLess` and `Waiting` When a swapping tree is used to (continuously) generate EPs, there are two fundamentally different generation protocols [104, 114].

- `WaitLess` *Protocol.* In this model, all the underlying processes, including link EP generations and atomic BSMs are synchronized. If all of them succeed then the end-to-end EP is generated. If *any* of the underlying processes fail, then all the generated EPs are discarded and the whole process starts again from scratch (from generation of EP at links). In the `WaitLess` protocol, all swapping trees over a given path P incur the same generation latency—thus, here, the goal is to select an optimal path P (as in [109, 25]).

- `Waiting` *Protocol.* In `Waiting` protocol, a qubit of an EP may wait (in a quantum memory) for its counterpart to become available so that an ES operation can be performed. Using such storage, we preclude discarding successfully generated EPs, and thus, reduce the overall latency in generation of a root-level EP. E.g., let (A, B) and (B, C) be two siblings in a swapping tree and EP for (A, B) is generated first. Then, EP (A, B) may wait for the EP (B, C) to be successfully generated. Once the EP (B, C) is generated, the ES operation is done over the triplet (A, B, C) to generate the EP (A, C). If the EP (A, B) waits beyond a certain threshold, then it may decohere.

Hardware Requirement Differences. `WaitLess` protocols can generate EPs without quantum memories in a relay fashion if the EP generation among adjacent nodes can be tightly synchronized. In contrast, `Waiting` protocols benefit from memories with good coherence times (§5.5).

Why `Waiting`'s Entanglement Generation Rate is Never Worse. The focus of the `WaitLess` protocol is to avoid qubit decoherence due to storage. But it results in very low generation rates due to a very-low probability of *all* the underlying processes succeeding at the same time. However, since qubit coherence times are typically higher than the link-generation latencies[3], an appropriately designed `Waiting` protocol will always yield better generation rates *without significantly compromising the fidelity* The key is to bound the waiting time to limit decoherence as desired; e.g., in our protocol, we restrict to trees with high expected fidelities (§5.3), and discard qubits that "age" beyond a threshold. Both protocols use the same number of quantum memories (2 per node), though the `Waiting` protocols will benefit from low-decoherence memories; other hardware requirements also remain the same.

[3]Link generation latencies for 5 to 100km links range from about 3 to 350 milliseconds for typical network parameters [21], while coherence times of few seconds is very realistic (coherence times of several seconds [77, 47] have been shown long ago, and more recently, even coherence times of several minutes [113, 102] to a few hours [139, 120] have been demonstrated.

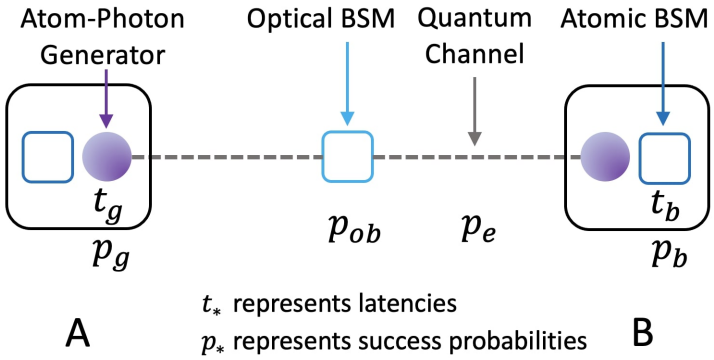


FIGURE 5.3: Key notations used.

5.3 Model, Problem, and Related Works

In this section, we discuss our network model, formulate the problem addressed, and discuss related work.

Network Model. We denote a quantum network (QN) with a graph $G = (V, E)$, with $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{(v_i, v_j)\}$ denoting the set of nodes and links respectively. Pairs of nodes connected by a link are defined as *adjacent* nodes. We follow the network model in [21] closely. Thus, each node has an atom-photon EP generator with generation latency (t_g) and probability of success (p_g). Generation latency is the time between successive attempts by the node to excite the atom to generate an atom-photon EP; this implicitly includes the times for photon transmission, optical-BSM latency, and classical acknowledgement. *For clarity of presentation* and without loss of generality, we assume homogeneous network nodes with same parameter values. The generation rate is the inverse of generation latency, as before. A node's atom-photon generation capacity/rate is its aggregate capacity, and may be split across its incident links (i.e., in generation of EPs over its incident links/nodes). Each node is also equipped with a certain number of atomic memories to store the qubits of the atom-atom EPs.

A network link is a quantum channel (e.g., using an optical fiber or a free-space link), and, in our context, is used only for establishment of link EP. In particular, a link $e = (A, B)$ is used to transmit telecom-photons from A and B to the photon-photon BSM device in the middle of e. Thus, each link is composed of two half-links with a probability of transmission success (p_e) that decreases exponentially with the link distance (see §5.5). The optical-BSM operation has a certain probability of success (p_{ob}). To facilitate atom-atom ES operations, each network node is also equipped with an atomic-BSM device with an operation latency (t_b) and probability of success (p_b). Finally, there is an independent classical network with a transmission latency (t_c); we assume classical transmission always succeeds.

Single vs. Multiple Links Between Nodes. For our techniques multiple links between a pair of adjacent nodes can be replaced by a single link of aggregated rate/capacity. Hence we assume only a single link between every pair of nodes. However, distinct multiple links between nodes have been used creatively in [109] (which refers to them as multiple channels); thus, we will discuss multiple links further in §5.5 when we evaluate various techniques. We note that the all-photonic protocol in [8] is essentially a more sophisticated version of the multi-link `WaitLess` protocol in [109] to further minimize memory requirements, but it uses multipartite cluster states which are challenging to create. In either case, in terms of selection of paths/trees, the path-selection techniques

from [109] should also apply to the all-photonic protocol with certain modifications to account for how the cluster states are generated.

EP Generation Latency of a Swapping Tree. Given a swapping tree and EP generation rates at the leaves (network links), we wish to estimate the generation latency of the EPs over the remote pair corresponding to the tree's root with the `Waiting` protocol. Below, we develop a recursive equation. Consider a node (A, C) in the tree, with (A, B) and (B, C) as its two children. Let T_{AB}, T_{BC}, and T_{AC} be the corresponding (expected) generation latencies of the EPs over the three pairs of nodes. Below, we derive an expression for T_{AC} in terms of T_{AB} and T_{BC}; this expression will be sufficient to determine the expected latency of the overall swapping tree by applying the expression iteratively. We start with an observation. If two EP arrival processes X_1 and X_2 are exponentially distributed with a mean inter-arrival latency of λ each, then the expected inter-arrival latency of $\max(X, Y)$ is $(3/2)\lambda$. From above, if assume T_{AB} and T_{BC} to be exponentially distributed with the same expected generation latency of T, then the expected latency of both EPs arriving is $(3/2)T$. Thus, we have:

$$T_{AC} = (\frac{3}{2}T + t_b + t_c)/p_b, \tag{5.1}$$

Remarks. We make the following remarks regarding the above expression. First, when $T_{AB} \neq T_{BC}$, we are able to only derive an upper-bound on T_{AC} which is given by the above equation but with T replaced by $\max(T_{AB}, T_{BC})$.[4] Second, our motivation for the exponential distribution assumption stems from the fact that the EP generation latency at the *link level* is certainly exponentially distributed if we assume the underlying probabilistic events to have a Poisson distribution. Third, note that the resulting distribution is not exponential. Despite this, we apply the above equation recursively to compute the tree's generation latency. Finally, Eqn. 5.1 is conservative in the sense that each round of an EP generation of any subtree's root starts from scratch (i.e., with no link EPs from prior round) and ends with either a EP generation at the *whole swapping tree*'s root or an atomic-BSM failure at the subtree's root. We do not "pipeline" any operations across rounds within a subtree, which may lower latency; this is beyond this work's scope.

5.3.1 Problem Formulation

We now formulate the central problem of selecting a *single* swapping trees for a single source-destination pair.

`QNR` Single Path (`QNR-SP`) Problem. Given a quantum network and a source-destination pair (s, d), the `QNR-SP` problem is to determine a single swapping tree that maximizes the expected generation rate (i.e., minimizes the expected generation latency) of EPs over (s, d), under the following capacity and fidelity constraints:

1. *Node Constraints.* For each node, the aggregate resources used by $\bigcup_i \mathbf{T}_i$ is less than the available resources; we formulate this formally below.

2. *Fidelity Constraints.* Each swapping tree in $\bigcup_i \mathbf{T}_i$ satisfies the following: (a) Number of leaves is less than a given threshold τ_l; this is to limit fidelity degradation

[4] The 3-over-2 formula as an upper bound has also been corroborated in a recent work [30] which derives analytical bounds on EP latency times in more general contexts.

due to gate operations. (b) Total memory storage time of any qubit is less[5] than a given *decoherence threshold* τ_d.

Informally, the swapping-trees may also satisfy some fairness constraint across the given source-destination pairs. A special case of the above `QNR` problem is to select a single tree for a source-destination pair; we address this in the next section.

Formulating Node Constraints. Consider a swapping tree $\mathbf{T} \in \bigcup_i \mathbf{T}_i$ over a path P. For each link $e \in P$, let $R(e, \mathbf{T})$ be the EP rate being used by $\mathbf{T}$over the link e in P. Let us define $R_e = \sum_{\mathbf{T}} R(e, \mathbf{T})$, and let $E(i)$ be the set of edges incident on i. Then, the node capacity constraint is formulated as follows.

$$1/t_g \quad \geq \quad \sum_{e \in E(i)} R_e/({p_g}^2 {p_e}^2 p_{ob}) \qquad \forall i \in V. \tag{5.2}$$

The above comes from the fact that to generate a single link EP over e, each end-node of e needs to generate $1/({p_g}^2 {p_e}^2 p_{ob})$ photons successfully, since each photon (from each end-node) has a generation success of p_g and a transmission success rate of p_e, and the optical-BSM's success probability over the two successfully arriving photons is p_{ob}. Note that $1/t_g$ is a node's total generation capacity. Also, the memory constraint is that for any node i, the memory available in i should be more than $2x + y$ where x is the number of swapping trees that use i as an intermediate node and y is the number of trees that use i as an end node.

For homogeneous nodes and link parameters, it is easy to see that the best swapping-tree is the balanced or almost-balanced tree over the shortest path. As described in §5.3.2, the `QNR-SP` problem has been addressed before in [109, 21] under different models. The problem of selecting *multiple* swapping trees for *multiple* source-destination pairs is solved in [50].

5.3.2 Related Works

There have been a few works in the recent years that have addressed generating long-distance EPs efficiently. All of these works have focused on selecting an efficient routing path for the swapping process ([21] also selects a path, but using a metric based on balanced trees). In addition, all except [21] have looked at the `WaitLess` protocol of generating the EPs. Recall that in the `WaitLess` model, selection of paths suffice, while in the `Waiting` model, one needs to consider selection of efficient swapping trees with high fidelity. Selection of optimal swapping trees is a fundamentally more challenging problem than selection of paths—and has not been addressed before, to the best of our knowledge. We start with discussing how the `WaitLess` model works.

`WaitLess` Approaches. The most recent works to address the above problem are [109] and [25], both of which consider the `WaitLess` model. In particular, Shi and Qian [109] design a Dijkstra-like algorithm to construct an optimal path between a pair of nodes, when there are multiple links (channels) between adjacent nodes. Then, they use the algorithm iteratively to select multiple paths over multiple pairs of nodes. Chakraborty

[5] We note that, in our context, the storage time as well as the memory coherence time are statistical quantities due to the underlying statistical mechanisms. However, for the purposes of *selecting* a swapping tree, we use a fixed decoherence threshold τ_d value equal to the mean of the distribution of the coherence time (recent work [71] computes optimal cut-offs/thresholds, and their techniques can be used to pick τ_d). When simulating a selected tree for generation of EPs, we can implement coherence time as a statistical measure.

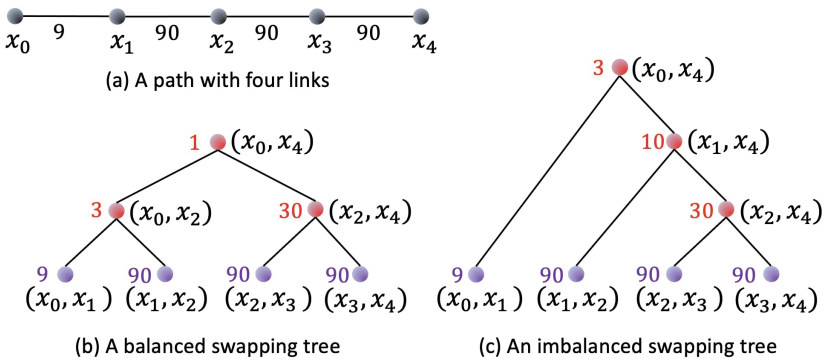


FIGURE 5.4: Consider the path in (a). The imbalanced tree of (b) has a higher EP generation rate than that of the balanced tree of (c). Here, the numbers represent the EP generation rates over adjacent links or node-pairs.

et al. [25] design a multi-commodity-flow like LP formulation to select routing paths for a set of source destination pairs. They map the operation-based fidelity constraint to the path length (as in [18]), and use node copies to model the constraint in the LP. However, they explicitly assume that the link EP generation is deterministic—i.e., always succeeds. Among earlier relevant works, [87] proposes a greedy solution for grid networks, and [26] proposes virtual-path based routing in ring/grid networks.

`Waiting` **Approach.** Due to photon loss, establishing long-distance entanglement between remote nodes at L distance by *direct* transmission yields EP rates that decay exponentially with L. DLCZ protocol [41, 104] broke this exponential barrier using 2^k equidistant intermediate nodes to perform entanglement-swapping operations, implicitly over a balanced binary tree, with a `Waiting` protocol; this makes the EP generation rate decay only polynomially in L. More recently, Caleffi [21] formulated the entanglement generation rate on a given path between two nodes, under the more realistic condition where the intermediate nodes in the path may not all be equidistant, but still considered only balanced trees. Their path-based metric was then used to select the optimal path by enumerating over the exponentially many paths in the network.

Our Approach (vs. [21]). Though [21] considers only balanced trees, its brute-force algorithm is literally impossible to run for networks more than a few tens of nodes. In our work, we observe that a path has many swapping trees, and, in general, imbalanced trees may even be better; see Figure 5.4. Thus, [50] design a polynomial-time dynamic programming (DP) algorithm that delivers an *optimal* high-fidelity swapping-tree; the approach effectively considers all possible swapping trees, not just balanced ones (note that, even over a single path, there are exponentially many trees). Our `Balanced-Tree` Heuristic (§5.4) is closer to [21]'s work, in that both consider only balanced trees; however, we use a heuristic metric that facilitates a polynomial-time Dijkstra-like heuristic to select the optimal path, while their recursive metric [6] (albeit more accurate than ours) is not amenable to an efficient (polynomial-time) search algorithm.

Other Works. In [61], Jiang et al. address a related problem; given a path with uniform link-lengths, they give an algorithm for selecting an optimal sequence of swapping and purification operations to produce an EP with fidelity constraints. In other recent works, Dahlberg et al [33] design physical and link layer protocols of a quantum network stack,

[6] We note that their formula (Eqn. 10 in [21]) is incorrect as it either ignores the 3/2 factor or assumes the EP generations to be synchronized **across all** links. In addition, their expression for "qubit age" ignores the "waiting for ES " time completely.

and [69] proposes a data plane protocol to generate EPs within decoherence thresholds along a *given* routing path. More recently, Bugalho et al. [19] propose an algorithm to efficiently distribute multipartite entanglement across over than two nodes.

5.4 `Balanced-Tree` Heuristic for `QNR-SP`

The DP-based algorithms presented in [50] for the `QNR-SP` problem have high time complexity, and thus, may not be practical for real-time route finding in large networks. In this section, we develop an almost-linear time heuristic for the `QNR-SP` problem, based on the classic Dijkstra shortest path algorithm; the designed heuristic performs close to the DP-based algorithms in our empirical studies.

Basic Idea. The main reason for the high-complexity of our DP-based algorithms in [50] is that the goal of the `QNR-SP` problem is to select an optimal swapping *tree* rather than a path. One way to circumvent this challenge efficiently while still selecting near-optimal swapping tree, is to restrict ourselves to only "balanced" swapping trees. This restriction allows us to think in terms of selection of paths—rather than trees—since each path has a unique[7] balanced swapping tree. We can then develop an appropriate path metric based on above, and design a Dijkstra-like algorithm to select an (s, d) path that has the optimal metric value. We note that Caleffi [21] also proposed a path metric based on balanced swapping trees, but their metric, though accurate, only had a recursive formulation without a closed-form expression—and hence, was ultimately not useful in designing an efficient algorithm. In contrast, we develop an approximate metric with a closed-form expression, based on the "bottleneck" link, as follows.

Path Metric M. Consider a path $P = (s, x_1, x_2, \ldots, x_n, d)$ from s to d, with links $(s, x1), (x1, x2), \ldots, (x_n, d)$ with given EP latencies. We define the path metric for path P, $M(P)$, as the EP generation latency of a balanced swapping over P, which can be estimated as follows. Let L be the link in P with maximum generation latency. If L's depth (distance from the root) is the maximum in a throttled swapping tree, then we can easily determine the accurate generation latency of the tree. However, in general, L may not have the maximum depth, in which case we can still estimate the tree's latency approximately, if the tree is balanced, as follows. In balanced swapping trees, *assuming* the maximum latency link L to be at the maximum depth, gives us a constant-factor approximation of the tree's generation latency. Thus, let us assume L to be at the maximum depth of a balanced tree over P; this maximum depth is $d = \lceil(\log_2 |P|)\rceil$. Let the generation latency of L be T_L. If we ignore the $t_b + t_c$ term in Eqn. 5.1 , then, the generation latency of a throttled swapping tree can be easily estimated to $T(\frac{3}{2p_b})^d$. The term $t_b + t_c$ can also be incorporated as follows. Let $T(i)$ denote the expected latency of the ancestor of L at a distance i from L. Then, we get the recursive equation: $T(i) = (\frac{3}{2}T(i-1) + t_b + t_c)/p_b$. Then, the path metric value $M(P)$ for path P is given by $T(d)$, the generation latency of the tree's root at a distance of d from L, and is equal to:

$$M(P) = T(d) = \bar{p}^d T_L + [(\bar{p}^d - 1)/(\bar{p} - 1)](t_b + t_c)/p_b$$

where $\bar{p} = 3/(2p_b)$ and $d = \lceil(\log_2 |P|)\rceil$. The above is a $(1+3/(2p_b))$-factor approximation latency of a balanced and throttled swapping tree over P.

[7] In fact, there can be multiple balanced trees over a path whose length is not a power of 2, but, since they differ minimally in our context, we can pick a unique way of constructing a balanced tree over a path.

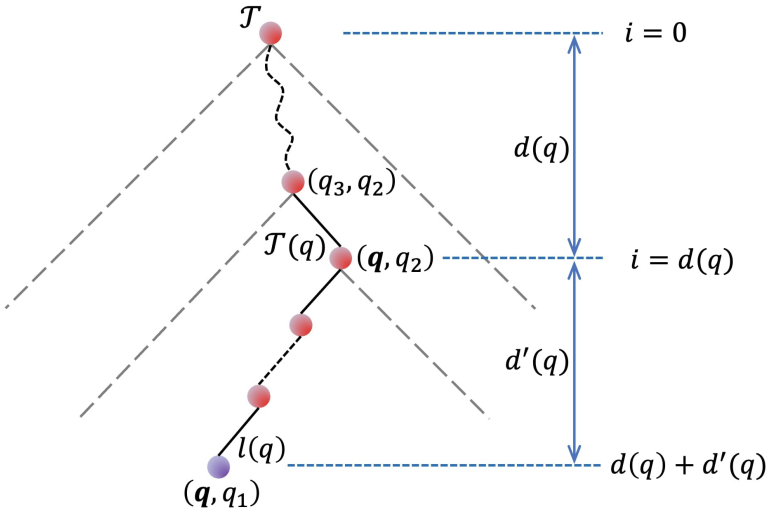


FIGURE 5.5: Qubit parameters in a swapping tree used to compute the *age* of a qubit q at a leaf node $l(q)$. Here, $l(q)$ is the left-most leaf of the subtree $\mathbf{T}(q)$.

Optimal Balanced-Tree Selection. The above path-metric $M()$ is a monotonically increasing function over paths, i.e., if a path P_1 is a sub-sequence of another path P_2, then $M(P_1) \leq M(P_2)$. Thus, we can tailor the classical Dijkstra's shortest path algorithm to select a (s, d) path with minimum $M(P)$ value, using the link's EP generation latencies as their weights. We refer to this algorithm as `Balanced-Tree`, and it can be implemented with a time complexity of $O(m + n \log n)$ using Fibonacci heaps, where m is the number of edges and n is the number nodes in the network.

Incorporating Fidelity Constraints.

Definition 1. (G)iven a swapping tree, the total time spent by a qubit in a swapping tree is the time spent from its "birth" via an atom-photon EP generation at a node till its consumption in a swapping operation or in generation of the tree's root EP. We refer to this as a qubit's *age*. The maximum age over all qubits in a swapping tree is called the tree's (expected) *age*. See Fig. 5.5. □

Fidelity constraints in our path-metric based setting can be handled by essentially computing the optimal path for each path-length (number of hops in the path) up to τ_l, and then pick the best path among them that satisfies the fidelity constraints. This obviously limits the number of leaves to τ_l and addresses the operations-based fidelity degradation. The above also address the decoherence/age constraint, since it is easy to see that the age of a balanced swapping tree can be very closely approximated in terms of the latency and the number of leaves. Now, to compute the optimal path for each path-length, we can use a simple dynamic programming approach that run in $O(m\tau_l)$ time where m is the number of edges and τ_l is the constraint on number of leaves.

5.5 Evaluations

The goal of our evaluations is to compare the EP generation rates, evaluate the fidelity of generated EPs, and validate our analytical models. We implement the various schemes over a discrete event simulator for QNs called NetSquid [31]. The NetSquid simulator accurately models various QN components/aspects, and in particular, we are able to

define various QN components and simulate swapping-trees protocols by implementing gate operations in entanglement swapping.

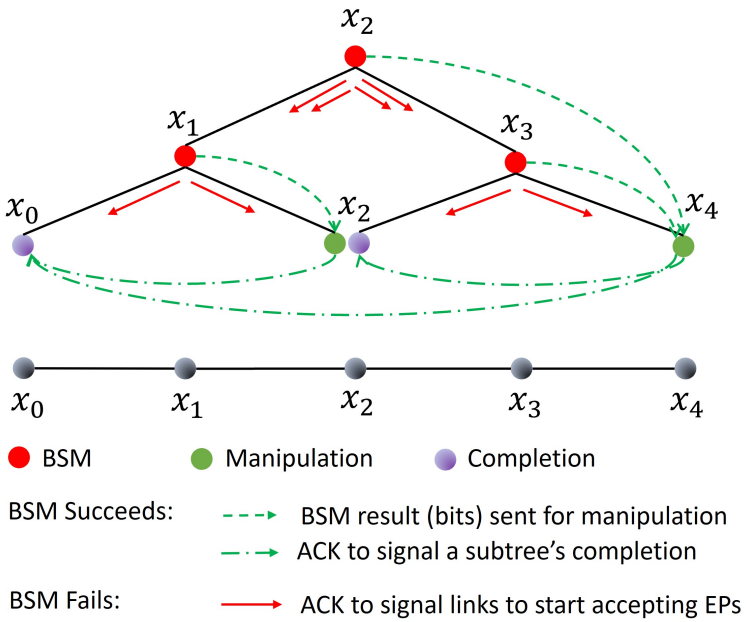


FIGURE 5.6: Swapping Tree Protocol Illustration. The shown tree is a certain hierarchy of nodes to illustrate the BSM operation in the swapping-tree protocol. A link-layer protocol continuously generates EPs over links (x_0, x_2) and (x_2, x_4). On receiving EP on links on either side, x_1 (x_3) attempts a BSM operation on the stored qubit atoms. If the BSM succeeds, x_1 (x_3) sends two classical bits (solid green arrows) to x_2 (x_4) for desired manipulation/correction after which x_2 (x_4) sends an ACK (dashed green arrows) to the other end-node x_0 (x_2) to complete the EP generation. If BSM at x_1 and x_3 are both successful, then x_2 attempts the BSM as above. If a BSM at say x_1 fails, that x_1 failure signals (red arrows) to all the descendant nodes of the subtree rooted at x_1 so that they can start accept new EPs from the link layer protocol. Note that here node x_2 plays multiple roles and hence appears at multiple places in the figure.

Swapping Tree Protocol. Our algorithms compute swapping tree(s), and we need a way to implement them on a network. We build our protocol on top of the link-layer of [33], which is delegated with the task of continuously generating EPs on a link at a desired rate (as per the swapping tree specifications). Note that a link (a, b) may be in multiple swapping trees, and hence, may need to handle multiple link-layer requests at the same time; we implement such link-layer requests by creating independent atom-photon generators at a and b, with one pair of synchronized generators for each link-layer request. As the links generate continuous EPs at desired rates, we need a protocol to swap the EPs. Omitting the tedious bookkeeping details, the key aspect of the protocol is that swap operation is done only when both the appropriate EP pairs have arrived. We implement all the gate operations (including, atomic and optical BSMs) within NetSquid to keep track of the fidelity of the qubits. On BSM success, the swapping node transmits classical bits to the end node which manipulates its qubit, and send the final ack to the other end node. On BSM failure, a classical ack is send to all descendant link leaves, so that they can now start accepting new link EPs; note that in our protocol, a link l does not accept any more EPs, while its ancestor is waiting for its sibling's EP. See Fig. 5.6.

Simulation Setting. We use a similar setting as in the recent work [109]. By default, we use a network spread over an area of $100km \times 100km$. We use the Waxman model [122], used to create Internet topologies, to randomly distribute the nodes and create links; we use the maximum link distance to be 10km. We vary the number of nodes from 25 to 500, with 100 as the default value. We choose the two parameters in the Waxman model to maintain the number of links to 3% of the complete graph (to ensure an average degree of 3 to 15 nodes). For the `QNR-SP` problem, we pick (s, d) pairs within a certain range of distance, with the default being 30-40 kms.

Parameter Values. We use parameter values mostly similar to the ones used in [21] corresponding to a single-atom based quantum memory platform, and vary some of them. In particular, we use atomic-BSM probability of success (p_b) to be 0.4 and latency (t_b) to be 10 μ secs; in some plots, we vary p_b from 0.2 to 0.6. The optical-BSM probability of success (p_{ob}) is half of p_b. We use atom-photon generation times (t_g) and probability of success (p_g) as 50 μsec and 0.33 respectively. Finally, we use photon transmission success probability as $e^{-d/(2L)}$ [21] where L is the channel attenuation length (chosen as 20km for an optical fiber) and d is the distance between the nodes. Each node's memory size is randomly chosen within a range of 15 to 20 units. Fidelity is modeled in NetSquid using two parameter values, viz., depolarization (for decoherence) and dephasing (for operations-driven) rates. We choose a decoherence time of two seconds based on achievable values with single-atom memory platforms [116]; note that decoherence times of even several minutes [113, 102] to hours [139, 120] has been demonstrated for other applicable memory platforms. Accordingly, we choose a depolarization rate of 0.01 such that the fidelity after a second is 90%. Similarly, we choose a dephasing rate of 1000 which corresponds to a link EP fidelity of 99.5% [25].

Algorithms and Performance Metrics. To compare our techniques with prior approaches, we implement most recently proposed approaches, viz., (i) the `WaitLess`-based linear programming (LP) approach from [25] (called `Delft-LP` here), (ii) `Q-Cast` approach from [109] which is `WaitLess`-based but uses multiple links and requires memories. The `Waiting`-based algorithm by Caleffi [21] uses an exponential-time approach, and is thus compared only for small networks. The [87] and [26] approaches are not compared as they were found to be inferior to `Q-Cast`. Alrorithms `DP-OPT` and `DP-Approx` in [50] are also being compared. We compare our `Balanced-Tree` with all the above mentioned algorithms largely in terms of EP generation rates and the execution times.

For all algorithm except for `Q-Cast`, we use only one link between adjacent nodes, since only `Q-Cast` takes advantage of multiple links in a creative way. In particular, for `Q-Cast`, we use $W = 1, 5$, or 10 sub-links ([109] calls them channels) on each link, with the node and link "capacity" divided equally among the them. We note that in `Q-Cast` each node requires $2W$ memories (2 for each sub-link) with sufficient coherence time to allow for the entire swapping operation over the path to be completed. The `Delft-LP` approach explicitly assumes the generation of link EPs is deterministic, i.e., the value ${p_g}^2{p_e}^2p_{ob}$ is 1, and does not model node generation rates. We address these differences by extending their LP formulation: (i) We add a constraint on node generation rates, and (ii) add a ${p_g}^2{p_e}(i,j)^2p_{ob}$ factor to each link (i, j) in any path extracted from their LP solution.

Comparison with [21] for `QNR-SP` Problem. Note that [21] gives only an `QNR-SP` algorithm referred to as `Caleffi`; it takes exponential-time making it infeasible to run for network sizes much larger than 15-20. In particular, for network sizes 17-20, it takes several hours, and our preliminary analysis suggests that it will take of the order of

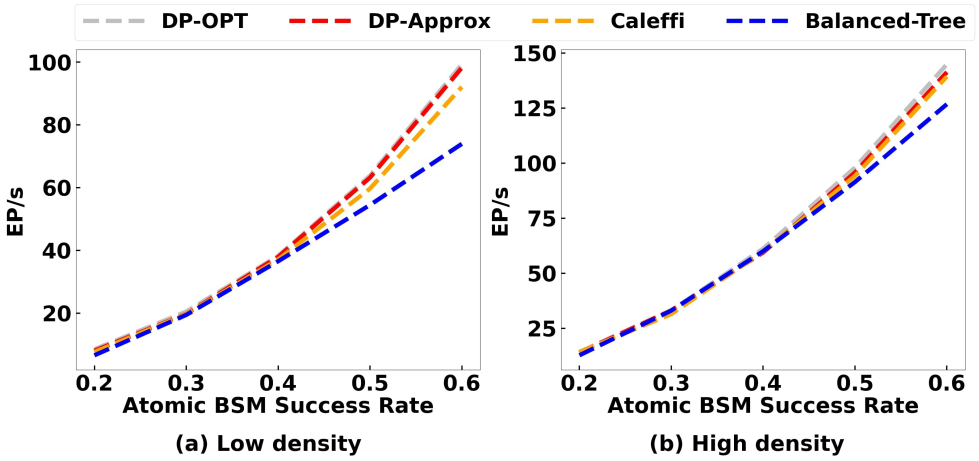


FIGURE 5.7: Compare the performance with Caleffi in a (a) low density network and a (b) high density network.

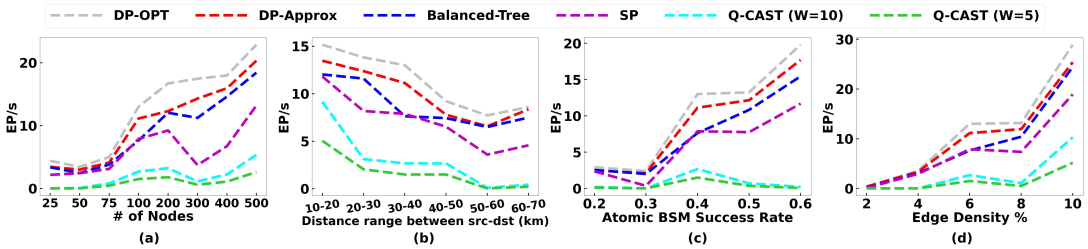


FIGURE 5.8: `QNR-SP` Problem: EP Generation Rates for varying parameters.

10^{40} *years* on our 100-node network. See Table 5.1. Thus, we use a small network of 15 nodes over a 25km × 25km area; we consider average node degrees of 3 or 6. See Fig.5.7. We see that `DP-OPT` outperforms `Caleffi` by 10% on a average for the sparser graph and minimally for the denser graph. We see that `DP-Approx` performs similar to `DP-OPT`, while `Balanced-Tree` is outperformed slightly by `Caleffi`; however, for this small network, since the `DP-OPT` and `DP-Approx` algorithms only take 10-100s of msecs (Table 5.1), `Balanced-Tree` need not be used in practice.

`QNR-SP` **Problem (Single Tree) Results.** We start with comparing various schemes for the `QNR-SP` problem, in terms of EP generation rate. We compare `DP-Approx`, `DP-OPT`, `Balanced-Tree`, `SP`, and `Q-Cast`; See Fig. 5.8, where we plot the EP generation rate for various schemes for varying number of nodes, (s, d) distance, p_b, and network link density. We observe that `DP-Approx` and `DP-OPT` perform very closely, with the `Balanced-Tree` heuristic performing close to them; all these three schemes outperform the `Q-Cast` schemes (for $W = 5, 10$ sub-links) by an order of magnitude. We don't plot `Q-Cast` for $W = 1$ sub-links, as it performs much worse (less than 10^{-3} EP/sec). We note that `Q-Cast`'s EP rates here are much lower than the ones published in [109], because [109] uses link EP success probability of 0.1 or more, while in our more realistic model, the link EP success probability is ${p_g}^2{p_e}^2 p_{ob} = 0.012$ for the default p_b value. We reiterate that our schemes require only 2 memory units per node, while the `Q-Cast` schemes requires $2W$ units. The main reason for poor performance of `Q-Cast` (in spite of higher memory and link synchronization) is that, in the `WaitLess` model, the EP generation over a path is a very low probability event—essentially p^l where p is the link-EP success probability and l is the path length, for the case of $W = 1$ (the analysis

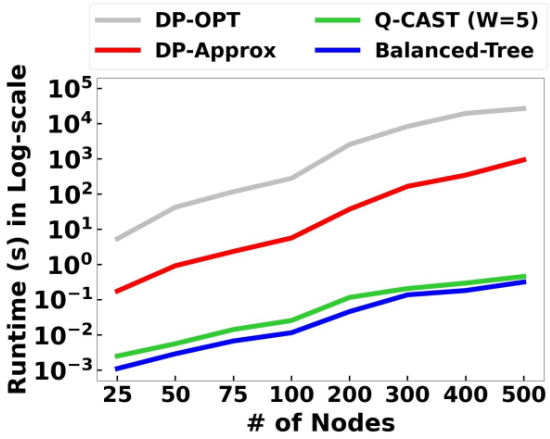


FIGURE 5.9: The execution time comparison of various algorithms for QNR-SP algorithms.

for higher W's is involved [109]). Finally, our proposed techniques also outperform the SP algorithm, especially when the number of possible paths (trees) between (s, d) pair increases. In addition, we see that performance increases with increase in p_b, number of nodes, or network link density, as expected due to availability of better trees/paths; it also increases with decrease in (s, d) distance as fewer hops are needed.

Execution Times. We ran our simulations on an Intel i7-8700 CPU machine, and observed that the WaitLess algorithms as well our Balanced-Tree and ITER-Bal heuristics run in fraction of a second even for a 500-node network; thus, they can be used in real-time. Note that since our problems depend on real-time network state (residual capacities), the algorithms must run very fast. The other algorithms (viz., DP-OPT, DP-Approx, and ITER-DPA) can take minutes to hours on large networks, and hence, may be impractical on large network without significant optimization and/or parallelization. See Fig 5.9.

TABLE 5.1: Execution times of QNR-SP algorithm over small networks

Algorithm	Number of nodes					
	10	13	15	16	18	20
Balanced-Tree	239μs	360μs	373μs	492μs	530μs	552μs
DP-Approx	4ms	10ms	14.7ms	17.6ms	28ms	34ms
DP-OPT	148ms	363ms	572ms	706ms	1s	1.7s
Caleffi [21]	92ms	4.6s	14s	26mins	3.2hrs	12.8hrs

Here, we give execution times of different algorithms especially Caleffi's for small networks of 10-20 nodes. See Table 5.1. We see that Balanced-Tree and DP-Approx take fractions of a second, while DP-OPT takes upto 2 seconds. However, as expected Caleffi's execution time increases exponentially with increase in number of nodes – with 20-node network takes 10+ hours. Below, we further estimate Caleffi's execution time for larger graphs.

5.6 Conclusion

We have designed techniques for efficient generation of EP to facilitate quantum network communication, by selecting efficient swapping trees in a Waiting protocol. By extensive

simulations, we demonstrated the effectiveness of our `Balanced-Tree`, i.e., compared to `DP-Approx`, it significantly decrease the time complexity while the performance drop is only minor.

Chapter 6

Proposed: Transmitter Localization in QSN with Measurement in the Computational Basis

In Chapter 4, we pose the transmitter localization problem as a quantum state discrimination problem and measure the quantum state reported by the quantum sensor network (QSN) via a positive-operator valued measure (POVM). Although POVM is good in theory, it is hard to implement in practice. Current hardware such as the IBM quantum computer only support measurement in the computational basis.

For the proposed work, we continue the work of transmitter localization via quantum state discrimination in a more practical way. In Chapter 4, our measurement is done via a POVM constructed by equations for square root measurement [57], which is also called pretty good measurement. The measurement problem in the context of quantum state discrimination is an optimization problem that can be formulated as a semidefinite programming problem (SDP) [44]. The pretty good measurement can be viewed as a relatively good heuristic for a measurement optimization problem. However, it has two problems:

1. Practical problem. POVM, the general measurement, is good in theory but implementing it in practice using a combination of single qubit rotation gate, CNOT gate and standard measurement gate (Fig. 6.1) is non-trivial. In the literature, there are works [129, 112] that try to implement a single-qubit two/three element POVM and cliam that their solutions can be generalized to arbitrary number of qubits and arbitrary number of elements. The solution (quantum circuits) in their work has a high circuit depth and requires ancillary qubits. Note that in Chapter 4, a POVM of 8 qubits and up to 256 elements is needed, and using the techniques from their work to implement the POVM we need looks daunting (at least to me).

2. Performance problem. As its name "pretty good" suggests, it is "okayish", but not "very good". SDP solvers [39] can optimally solve the measurement optimization problem, but the solver is not scalable as the number of qubits increases. Note that a variable in the SDP formulation is a matrix (operator) whose dimension is the square of 2 to the power of number of qubits. Using sub-optimal pretty good measurement implies that there are always room for improvement.

To solve first problem of practicality, we plan to use the measurement under the standard computational basis, as in the quantum sensing protocol described in [36]. Measurement in the computation basis is the only measurement operator provided by IBM quantum computers. See Fig. 6.1.

FIGURE 6.1: The only measurement gate provided by the IBM Quantum Computer is a measurement in the standard basis, also known as the z basis or computational basis. It can be used to implement any kind of measurement when combined with other gates.

However, directly using the computational measurement will likely be far from the optimal measurement, even a lot worse than the pretty good measurement. So this leads to serious a performance problem. To solve it, we apply some quantum gates before the standard measurement gates to transform the qubits for better serving the computational measurement. However, the design of the quantum gates is non-trivial. Take the example of single qubit rotational gate, it is hard to determine how large an angle to rotate. Thus, we plan to resort to parameterized quantum circuits (also named quantum neural networks), wherein the angles of the rotation can be *learned* through the training process. Also note that we are in the noisy intermediate-scale quantum (NISQ) era. So we plan to take noise in to consideration, which is completely ignored in Chapter 4.

Our end-goal is to be able to run the evaluation experiments on a *real IBM quantum computer* using computational basis measurement and parameterized quantum circuits. This is a huge leap compared with classical computer simulations done in Chapter 4. We may encounter some unknown road blocks, but the journey is going to be fun.

Appendix A

Appendix

A.1 Deduction of Lemma 1

Use S number of sensors to determine the power for the hypothesis at location l^*, such that the hypothesis has the maximum probability. Let $N(\mu_1, \sigma_1^2), \cdots, N(\mu_S, \sigma_S^2)$ be the PDs of the S sensors, built during the training phase, when a transmitter with power p^* is transmitting at location l^*. Let $\mathbf{x} = \{x_1, x_2, \cdots, x_S\}$ be the observation vector of the S sensors during the localization phase. Then we predict the hypothesis at location l^* most likely has the power $p^* + \delta_p$, where

$$\delta_p = \frac{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}(x_j - \mu_j)}{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}}$$

The deduction relies on the assumption that the path loss between a transmitter and receiver is independent of transmit power, or unchanged. The likelihood of $\mathbf{x}$, given $\mathbf{N} = \{N(\mu_1, \delta_1^2), \cdots, N(\mu_S, \delta_S^2)\}$ and δ_p is,

$$\begin{aligned}
P(\mathbf{x}|\mathbf{N}, \delta_p) =& \prod_{j=1}^{S} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left\{ (-\frac{(x_j - (\mu_j + \delta_p))^2}{2\sigma_j^2}) \right\} \\
log_e(P(\mathbf{x}|\mathbf{N}, \delta_p)) =& log_e(\prod_{j=1}^{S} \frac{1}{\sqrt{2\pi\sigma_j^2}}) - (\sum_{j=1}^{S} \frac{(x_j - (\mu_j + \delta_p))^2}{2\sigma_j^2}) \\
\frac{\partial(log_e(P(\mathbf{x}|\mathbf{N}, \delta_p)))}{\partial \delta_p} =& \sum_{j=1}^{S} \frac{x_j - \mu_j - \delta_p}{\sigma_j^2}
\end{aligned}$$

To maximize the likelihood of $\mathbf{x}$, let the differential equals to 0.

$$\begin{aligned}
\sum_{j=1}^{S} \frac{x_j - \mu_j - \delta_p}{\sigma_j^2} =& 0 \\
\Rightarrow \delta_p =& \frac{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}(x_j - \mu_j)}{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}} \quad , \text{where } \gamma = \prod_{j=1}^{S} \sigma_j^2
\end{aligned}$$

Bibliography

[1] (n.d.). https://www.ettus.com/all-products/ub210-kit/.

[2] (n.d.). https://greatscottgadgets.com/hackrf/one/.

[3] Alizadeh Kharazi, B. & Behzadan, A. H. (2021). Flood depth mapping in street photos with image processing and deep neural networks. *Computers, Environment and Urban Systems*.

[4] Andrews, J. G. [J. G.] et al. (2014). What will 5G be? *IEEE JSAC*, *32*(6).

[5] Andrews, J. G. [Jeffrey G.], Buzzi, S. et al. (2014). What will 5g be? *IEEE Journal on Selected Areas in Communications*.

[6] Arute et al, F. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, *574*, 505–510. doi:\url{https://doi.org/10.1038/s41586-019-1666-5}

[7] Ayyalasomayajula, R., Arun, A. et al. (2020). Deep learning based wireless localization for indoor navigation. In *Mobicom*.

[8] Azuma, K., Tamaki, K., & Lo, H.-K. (2015). All-photonic quantum repeaters. *Nature communications*, *6*(1), 1–7.

[9] Bae, J. & Kwek, L.-C. (2015). Quantum state discrimination and its applications. *Journal of Physics A: Mathematical and Theoretical*, *48*(8), 083001.

[10] Bahl, P. & Padmanabhan, V. N. (2000). RADAR: An in-building RF-based user location and tracking system. In *Ieee infocom*.

[11] Bennett, C. H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A., & Wootters, W. K. (1993). Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Phys. Rev. Lett. 70*(13), 1895–1899.

[12] Bergou, J. (2007). Quantum state discrimination and selected applications. *Journal of Physics: Conference Series*.

[13] Bergou, J. A., Herzog, U., & Hillery, M. (2004). 11 discrimination of quantum states. In *Quantum state estimation* (Chap. 11). Springer.

[14] Bhaskar, M. K., Riedinger, R., Machielse, B., Levonian, D. S., Nguyen, C. T., Knall, E. N., ... Sukachev, D. D. et al. (2020). Experimental demonstration of memory-enhanced quantum communication. *Nature*, *580*(7801), 60–64.

[15] Bhattacharya, A. [A.], Zhan, C., Gupta, H., Das, S. R., & Djuric, P. M. (2020). Selection of sensors for efficient transmitter localization. In *Ieee infocom*.

[16] Bhattacharya, A. [Arani], Abhishek, M., Champati, J. P., Gross, J. et al. (2022). Fast and efficient online selection of sensors for transmitter localization. In *International conference on communication systems & networks*.

[17] Bhattacharya, A. [Arani], Zhan, C., Maji, A., Gupta, H., Das, S. R., & Djurićc, P. M. (2021). Selection of sensors for efficient transmitter localization. *IEEE/ACM Transactions on Networking*.

[18] Briegel, H.-J., Dür, W., Cirac, J. I., & Zoller, P. (1998). Quantum repeaters: The role of imperfect local operations in quantum communication. *Phys. Rev. Lett. 81*, 5932–5935. doi:10.1103/PhysRevLett.81.5932

[19] Bugalho, L., Coutinho, B. C., & Omar, Y. (2021). Distributing multipartite entanglement over noisy quantum networks. *arXiv preprint arXiv:2103.14759*.

[20] Buhrman, H., Cleve, R., Watrous, J., & de Wolf, R. (2001). Quantum fingerprinting. *Phys. Rev. Lett. 87*, 167902.

[21] Caleffi, M. (2017). Optimal routing for quantum networks. *IEEE Access*.

[22] Caleffi, M., Cacciapuoti, A. S., & Bianchi, G. (2018). Quantum internet: From communication to distributed computing! In *5th acm international conference on nanoscale computing and communication*.

[23] Chakraborty, A., Bhattacharya, A. [A.], Kamal, S., Das, S., Gupta, H. et al. (2018). Spectrum patrolling with crowdsourced spectrum sensors. In *Ieee infocom*.

[24] Chakraborty, A., Rahman, M. S., Gupta, H., & Das, S. (2017). SpecSense: Crowdsensing for efficient querying of spectrum occupancy. In *Ieee infocom*.

[25] Chakraborty, K., Elkouss, D., Rijsman, B., & Wehner, S. (2020). Entanglement distribution in a quantum network: A multicommodity flow-based approach. *IEEE Transactions on Quantum Engineering*.

[26] Chakraborty, K., Rozpedek, F., Dahlberg, A., & Wehner, S. (2019). Distributed routing in a quantum internet. https://arxiv.org/abs/1907.11630. arXiv: 1907.11630 [quant-ph]

[27] Chamberlin, K. & Luebbers, R. (1982a). An evaluation of longley-rice and gtd propagation models. *IEEE Transactions on Antennas and Propagation*. doi:10.1109/TAP.1982.1142958

[28] Chamberlin, K. & Luebbers, R. (1982b). An evaluation of longley-rice and gtd propagation models. *IEEE Transactions on Antennas and Propagation*, *30*(6).

[29] Chen, T.-Y., Liang, H., Liu, Y., Cai, W.-Q., Ju, L., Liu, W.-Y., ... Chen, Z.-B. et al. (2009). Field test of a practical secure communication network with decoy-state quantum cryptography. *Optics express*, *17*(8), 6540–6549.

[30] Coopmans, T., Brand, S., & Elkouss, D. (2022). Improved analytical bounds on delivery times of long-distance entanglement. *Phys. Rev. A*, *105*, 012608. doi:10.1103/PhysRevA.105.012608

[31] Coopmans, T., Knegjens, R., Dahlberg, A., Maier, D., Nijsten, L., Oliveira, J., Wehner, S. et al. (2021). Netsquid, a discrete-event simulation platform for quantum networks. *Communications Physics*.

[32] Corbalan, P., Picco, G., & Palipana, S. (2019). Chorus: UWB concurrent transmissions for GPS-like passive localization of countless targets. In *Acm/ieee ipsn*.

[33] Dahlberg, A., Skrzypczyk, M., Coopmans, T., Wubben, L. et al. (2019). A link layer protocol for quantum networks. In *Sigcomm*.

[34] Dasari, M., Atigue, M. B., Bhattacharya, A., & Das, S. (2019). Spectrum protection from micro-transmissions using distributed spectrum patrolling. In *Pam*.

[35] de Almeida, I. B. F., Chafii, M., Nimr, A., & Fettweis, G. (2021). Blind transmitter localization in wireless sensor networks: A deep learning approach. In *Ieee pimrc*.

[36] Degen, C. L., Reinhard, F., & Cappellaro, P. (2017). Quantum sensing. *Rev. Mod. Phys. 89*, 035002.

[37] Deutsch, C., Ramirez-Martinez, F., Lacroûte, C., Reinhard, F., Schneider, T., Fuchs, J. N., ... Rosenbusch, P. (2010). Spin self-rephasing and very long coherence times in a trapped atomic ensemble. *Phys. Rev. Lett. 105*, 020401. doi:10.1103/PhysRevLett.105.020401

[38] Devitt, S. J., Munro, W. J., & Nemoto, K. (2013). Quantum error correction for beginners. *Reports on Progress in Physics*, *76*(7), 076001. doi:10.1088/0034-4885/76/7/076001

[39] Diamond, S. & Boyd, S. (2016). CVXPY: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, *17*(83), 1–5.

[40] Dieks, D. (1982). Communication by EPR devices. *Physics Letters A*, *92*(6), 271–272. doi:10.1016/0375-9601(82)90084-6

[41] Duan, L.-M., Lukin, M. D., Cirac, J. I., & Zoller, P. (2001). Long-distance quantum communication with atomic ensembles and linear optics. *Nature*.

[42] Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

[43] Dutta, A. & Chiang, M. (2016). "see something, say something" crowdsourced enforcement of spectrum policies. *IEEE Trans. on Wireless Comm. 15*(1).

[44] Eldar, Y. C., Megretski, A. et al. (2003). Designing optimal quantum detectors via semidefinite programming. *IEEE Trans. Information Theory*.

[45] Eldredge, Z., Foss-Feig, M., Gross, J. A., Rolston, S. L., & Gorshkov, A. V. (2018). Optimal and secure measurement protocols for quantum sensor networks. *Physical Review A*, *97*(4), 042337.

[46] *Electromagnetic spectrum superiority strategy*. (2020). US Department of Defence.

[47] Fraval, E., Sellars, M. J., & Longdell, J. J. (2005). Dynamic decoherence control of a solid-state nuclear-quadrupole qubit. *Phys. Rev. Lett.*

[48] Gambetta, J. (2020). IBM's Roadmap For Scaling Quantum Technology. https://www.ibm.com/blogs/research/2020/09/ibm-quantum-roadmap/.

[49] Ghaderibaneh, M. [M.], Dasari, M., & Gupta, H. (2019). Multiple transmitter localization under time-skewed observations. In *Ieee dyspan*.

[50] Ghaderibaneh, M. [Mohammad], Zhan, C., Gupta, H., & Ramakrishnan, C. R. (2022). Efficient quantum network communication using optimized entanglement swapping trees. *IEEE TQE*.

[51] Giovannetti, V., Lloyd, S., & Maccone, L. (2011). Advances in quantum metrology. *Nature Photonics*. doi:10.1038/nphoton.2011.35

[52] Girshick, R. et al. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

[53] Goswami, A., Ortiz, L., & Das, S. (2011). Wigem: A learning-based approach for indoor localization. In *Acm conext*.

[54] Grobwindhager, B., Stocker, M. et al. (2019). SnapLoc: An ultra-fast UWB-based indoor localization system for an unlimited number of tags. In *Acm/ieee ipsn*.

[55] Gühne, O. & Tóth, G. (2009). Entanglement detection. *Physics Reports*, *474*(1), 1–75. doi:https://doi.org/10.1016/j.physrep.2009.02.004

[56] Hartung, L. & Milind, M. (2015). Policy driven multi-band spectrum aggregation for ultra-broadband wireless networks. In *Ieee dyspan*.

[57] Hausladen, P. & Wootters, W. K. [William K.]. (1994). A 'pretty good' measurement for distinguishing quantum states. *Journal of Modern Optics*.

[58] Henri Nurminen, R. P., Marzieh Dashti. (2017). A survey on wireless transmitter localization using signal strength measurements. *Wireless Communications and Mobile Computing*.

[59] Hunter, J. D. (n.d.). Matplotlib: A 2d graphics environment. https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.psd.html. IEEE COMPUTER SOC.

[60] Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*.

[61] Jiang, L., Taylor, J. M., Khaneja, N., & Lukin, M. D. (2007). Optimal approach to quantum communication using dynamic programming. *Proceedings of the National Academy of Sciences*, *104*(44), 17291–17296. doi:10.1073/pnas.0703284104. eprint: https://www.pnas.org/content/104/44/17291.full.pdf

[62] Karanam, C., Korany, B., & Mostofi, Y. (2019). Tracking from one side – multi-person passive tracking with wifi magnitude measurements. In *Acm/ieee ipsn*.

[63] Khaledi, M. et al. (2017). Simultaneous power-based localization of transmitters for crowdsourced spectrum monitoring. In *Acm mobicom*.

[64] Kim, C. W., Ryoo, J., & Buddhikot, M. M. (2015). Design and implementation of an end-to-end architecture for 3.5 ghz shared spectrum. In *Ieee dyspan*.

[65] Kim, S., Jeon, H., Lee, H., & Ma, J. (2007). Robust transmission power and position estimation in cognitive radio. In *Ieee milcom*.

[66] Kingma, D. P. & Ba, J. (2017). Adam: A method for stochastic optimization.

[67] Komar, P., Kessler, E. M., Bishof, M., Jiang, L., Sørensen, A. S., Ye, J., & Lukin, M. D. (2014). A quantum network of clocks. *Nature Physics*, *10*(8), 582–587.

[68] Kour, H., Jha, R. K., & Jain, S. (2018). A comprehensive survey on spectrum sharing: Architecture, energy efficiency and security issues. *Journal of Network and Computer Applications*.

[69] Kozlowski, W., Dahlberg, A., & Wehner, S. (2020). Designing a quantum network protocol. In *Conext*. Barcelona, Spain.

[70] Langer, C., Ozeri, R., Jost, J. D., Chiaverini, J., DeMarco, B., Ben-Kish, A., ... Wineland, D. J. (2005). Long-lived qubit memory using atomic ions. *Phys. Rev. Lett. 95*, 060502. doi:10.1103/PhysRevLett.95.060502

[71] Li, B., Coopmans, T., & Elkouss, D. (2020). Efficient optimization of cut-offs in quantum repeater chains. In *2020 ieee international conference on quantum computing and engineering (qce)* (pp. 158–168). doi:10.1109/QCE49297.2020.00029

[72] Li, Z. [Zhijing], Xiao, Z., Wang, B., Zhao, B. Y., & Zheng, H. (2019). Scaling deep learning models for spectrum anomaly detection. In *Proceedings of the twentieth acm international symposium on mobile ad hoc networking and computing* (pp. 291–300). ACM.

[73] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B. et al. (2014). Microsoft COCO: common objects in context.

[74] Linder-Norén, E. (2019). *Open source yolov3 implementation*. Retrieved from https://github.com/eriklindernoren/PyTorch-YOLOv3

[75] Liu, L. [Li], Ouyang, W., Wang, X., Fieguth, P. W., Chen, J., Liu, X., & Pietikäinen, M. (2018). Deep learning for generic object detection: A survey. *CoRR*, *abs/1809.02165*.

[76] Liu, L. [Lizheng], Zhang, Y., Li, Z. [Zhangda] et al. (2021). Distributed quantum phase estimation with entangled photons. *Nature Photonics*.

[77] Longdell, J. J., Fraval, E., Sellars, M. J., & Manson, N. B. (2005). Stopped light with storage times greater than one second using electromagnetically induced transparency in a solid. *Phys. Rev. Lett.*

[78] Luo, W., Li, Y., Urtasun, R., & Zemel, R. S. (2017). Understanding the effective receptive field in deep convolutional neural networks. *CoRR*.

[79] Magliacane, J. A. (2008). SPLAT! a terrestrial RF path analysis application for Linux/Unix. https://www.qsl.net/kd2bd/splat.html.

[80] Marcozzi, M. & Mostarda, L. (2021). Quantum consensus: An overview. *arXiv preprint arXiv:2101.04192*.

[81] Muralidharan, S., Li, L., Kim, J., Lütkenhaus, N., Lukin, M. D. [Mikhail D.], & Jiang, L. (2016). Optimal architectures for long distance quantum communication. *Scientific Reports*, *6*. https://doi.org/10.1038/srep20463. doi:10.1038/srep20463

[82] Muralidharan, S., Li, L., Kim, J., Lütkenhaus, N., Lukin, M. D. [Mikhail D], & Jiang, L. (2016). Optimal architectures for long distance quantum communication. *Scientific reports*, *6*(1), 1–10.

[83] Narayanan, A., Zhang, X. et al. (2021). A variegated look at 5G in the wild: Performance, power, and QoE implications. In *Acm sigcomm*.

[84] Nelson, J. K. et al. (2009). A quasi EM method for estimating multiple transmitter locations. *IEEE Signal Processing Letters*, *16*(5).

[85] Nelson, J., Hazen, M., & Gupta, M. (2006). Global optimization for multiple transmitter localization. In *Ieee milcom*.

[86] Nielsen, M. A. & Chuang, I. L. (2011). *Quantum computation and quantum information: 10th anniversary edition* (10th). Cambridge University Press.

[87] Pant, M., Krovi, H., Towsley, D., Tassiulas, L., Jiang, L., Basu, P., ... Guha, S. (2020). Routing entanglement in the quantum internet. *NPJ Quantum Information*.

[88] Patwari, N. et al. (2005). Locating the nodes: Cooperative localization in wireless sensor networks. *IEEE Signal processing magazine*, *22*(4).

[89] Patwari, N. et al. (2010). Rf sensor networks for device-free localization: Measurements, models, and algorithms. *Proceedings of the IEEE*.

[90] Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

[91] Penumarthi, P. K. et al. (2017). Multirobot exploration for building communication maps with prior from communication models. In *Intl. symp. on multi-robot and multi-agent systems*.

[92] Press, D., De Greve, K., McMahon, P. L., Ladd, T. D., Friess, B., Schneider, C., ... Yamamoto, Y. (2010). Ultrafast optical spin echo in a single quantum dot. *Nature Photonics*, *4*(6), 367–370.

[93] Rajendran, S. [S.] et al. (2018). Deep learning models for wireless signal classification with distributed low-cost spectrum sensors. *IEEE TOCCN*.

[94] Rajendran, S. [Sreeraj], Calvo-Palomino, R., Giustiniano, D. et al. (2018). Electrosense: Open and big spectrum data. *IEEE Communications Magazine*.

[95] Rappaport, T. (2001). *Wireless communications: Principles and practice* (2nd). Upper Saddle River, NJ, USA: Prentice Hall PTR.

[96] Redmon, J. et al. (2018). Yolov3: An incremental improvement. *CoRR*.

[97] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Cvpr*.

[98] Reynders, B., Minucci, F., Perenda, E. et al. (2020). Skysense: Terrestrial and aerial spectrum use analysed using lightweight sensing technology with weather balloons. In *Mobisys*.

[99] Rizk, H. et al. (2019). MonoDCell: A ubiquitous and low-overhead deep learning-based indoor localization with limited cellular information. In *Sigspatial*. Chicago, IL, USA.

[100] Roffe, J. (2019). Quantum error correction: An introductory guide. *Contemporary Physics*, *60*(3), 226–245. doi:10.1080/00107514.2019.1667078

[101] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Miccai 2015*.

[102] Saeedi, K., Simmons, S., Salvail, J. Z., Dluhy, P., Riemann, H., Abrosimov, N. V., ... Thewalt, M. L. (2013). Room-temperature quantum bit storage exceeding 39 minutes using ionized donors in silicon-28. *Science*, *342*(6160), 830–833.

[103] Sagi, Y., Almog, I., & Davidson, N. (2010). Process tomography of dynamical decoupling in a dense cold atomic ensemble. *Phys. Rev. Lett. 105*, 053201. doi:10.1103/PhysRevLett.105.053201

[104] Sangouard, N., Simon, C., De Riedmatten, H., & Gisin, N. (2011). Quantum repeaters based on atomic ensembles and linear optics. *Reviews of Modern Physics*.

[105] Sarkar, S., Buddhikot, M., Baset, A., & Kasera, S. K. (2021). DeepRadar: A deep-learning-based environmental sensing capability sensor design for CBRS. MobiCom.

[106] Scarani, V., Bechmann-Pasquinucci, H., Cerf, N. J., Dušek, M., Lütkenhaus, N., & Peev, M. (2009). The security of practical quantum key distribution. *Reviews of modern physics*, *81*(3), 1301.

[107] Schmidt, M. et al. (2017). Wireless interference identification with convolutional neural networks. In *Ieee intl. conf. on industrial informatics (indin)*.

[108] Schuld, M. & Petruccione, F. (2018). *Supervised learning with quantum computers*. Springer Publishing Company, Incorporated.

[109] Shi, S. & Qian, C. (2020). Concurrent entanglement routing for quantum networks: Model and designs. In *Sigcomm*.

[110] Shokry, A. & Youssef, M. (2022). A quantum algorithm for rf-based fingerprinting localization systems. In *Ieee 47th conference on local computer networks (lcn)*.

[111] Simon, C. (2017). Towards a global quantum network. *Nature Photonics*, *11*(11), 678–680.

[112] Singh, J., Arvind, & Goyal, S. K. (2022). Implementation of discrete positive operator valued measures on linear optical systems using cosine-sine decomposition. *Phys. Rev. Research*, *4*, 013007. doi:10.1103/PhysRevResearch.4.013007

[113] Steger, M., Saeedi, K., Thewalt, M., Morton, J., Riemann, H., Abrosimov, N., ... Pohl, H.-J. (2012). Quantum information storage for over 180 s using donor spins in a 28si "semiconductor vacuum". *Science*, *336*(6086), 1280–1283.

[114] Tittel, W., Afzelius, M., Cone, R. L., Chanelière, T., Kröll, S., Moiseev, S. A., & Sellars, M. (2008). Photon-echo quantum memory. https://arxiv.org/abs/0810.0172. arXiv: 0810.0172 [quant-ph]

[115] Üreten, O. & Willink, T. J. (2011). Joint estimation of emitter power and location in cognitive radio networks. In *2011 ieee 12th international workshop on signal processing advances in wireless communications* (pp. 61–65).

[116] van Loock, P., Alt, W., Becher, C., Benson, O., Boche, H., Deppe, C., ... Weinfurter, H. (2020). Extending quantum links: Modules for fiber- and memory-based quantum repeaters. *Advanced Quantum Technologies*, *3*(11), 1900141. doi:https://doi.org/10.1002/qute.201900141. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qute.201900141

[117] Vetsch, E., Reitz, D., Sagué, G., Schmidt, R., Dawkins, S. T., & Rauschenbeutel, A. (2010). Optical interface created by laser-cooled atoms trapped in the evanescent field surrounding an optical nanofiber. *Phys. Rev. Lett. 104*, 203603. doi:10.1103/PhysRevLett.104.203603

[118] Wang, H., He, Y.-M., Chung, T.-H., Hu, H., Yu, Y., Chen, S., ... Yang, X. et al. (2019). Towards optimal single-photon sources from polarized microcavities. *Nature Photonics*, *13*(11), 770–775.

[119] Wang, P., Luan, C.-Y., Qiao, M., Um, M., Zhang, J., Wang, Y., ... Kim, K. (2021a). Single ion qubit with estimated coherence time exceeding one hour. *Nature communications*, *12*(1), 1–8.

[120] Wang, P., Luan, C.-Y., Qiao, M., Um, M., Zhang, J., Wang, Y., ... Kim, K. (2021b). Single ion qubit with estimated coherence time exceeding one hour. *Nature Communications*, *12*(1). doi:10.1038/s41467-020-20330-w

[121] Wang, X., Gao, L., Mao, S., & Pandey, S. (2017). Csi-based fingerprinting for indoor localization: A deep learning approach. *IEEE Transactions on Vehicular Technology*.

[122] Waxman, B. (1988). Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*. doi:10.1109/49.12889

[123] Wootters, W. K. [William K] & Zurek, W. H. (1982). A single quantum cannot be cloned. *Nature*, *299*(5886), 802–803.

[124] Wright, D. & Clegg, A. (2016). The 3.5 GHz citizens broadband radio service (CBRS).

[125] Wu, Y. & He, K. (2018). Group normalization. *CoRR*. Retrieved from http://arxiv.org/abs/1803.08494

[126] Xia, Y. (2021). *Distributed quantum sensing: Theoretical foundation, experimental platform and applications* (Doctoral dissertation, U of Arizona).

[127] Xia, Y., Li, W., Clark, W., Hart, D., Zhuang, Q., & Zhang, Z. (2020). Demonstration of a reconfigurable entangled radio-frequency photonic sensor network. *Phys. Rev. Lett.*

[128] Xiong, J. & Jamieson, K. (2013). ArrayTrack: A Fine-Grained indoor location system. In *Nsdi*. USENIX Association.

[129] Yordanov, Y. S. & Barnes, C. H. W. (2019). Implementation of a general single-qubit positive operator-valued measure on a circuit-based quantum computer. *Phys. Rev. A*, *100*, 062317. doi:10.1103/PhysRevA.100.062317

[130] Youssef, M. & Agrawala, A. (2005). The horus wlan location determination system. In *Mobisys*.

[131] Zafari, F., Gkelias, A., & Leung, K. K. (2019). A survey of indoor localization systems and technologies. *IEEE Communications Surveys Tutorials*, *21*(3).

[132] Zafer, M., Ko, B. J., & Ho, I. W.-H. (2010). Transmit power estimation using spatially diverse measurements under wireless fading. *IEEE/ACM Transactions on Networking*.

[133] Zeng, Y., Chandrasekaran, V., Banerjee, S. et al. (2019). A framework for analyzing spectrum characteristics in large spatio-temporal scales. In *Acm mobicom*.

[134] Zhan, C., Ghaderibaneh, M., Sahu, P., & Gupta, H. (2021). DeepMTL: Deep learning based multiple transmitter localization. In *Ieee 22nd international symposium on a world of wireless, mobile and multimedia networks (wowmom)*.

[135] Zhan, C., Ghaderibaneh, M., Sahu, P., & Gupta, H. (2022). DeepMTL Pro: Deep learning based multiple transmitter localization and power estimation. *Pervasive and Mobile Computing*.

[136] Zhan, C., Gupta, H., Bhattacharya, A., & Ghaderibaneh, M. (2020). Efficient localization of multiple intruders for shared spectrum system. In *IPSN*.

[137] Zhang, R., Liu, J., Du, X., Li, B., & M, M. G. (2018). AOA-based three-dimensional multi-target localization in industrial WSNs for LOS conditions. *Sensors*.

[138] Zhang, Z. & Zhuang, Q. (2021). Distributed quantum sensing. *Quantum Science and Technology*, *6*(4), 043001.

[139] Zhong, M., Hedges, M. P., Ahlefeldt, R. L., Bartholomew, J. G., Beavan, S. E., Wittig, S. M., ... Sellars, M. J. (2015). Optically addressable nuclear spins in a solid with a six-hour coherence time. *Nature*, *517*(7533), 177–180.

[140] Zubow, A., Bayhan, S., Gawłowicz, P., & Dressler, F. (2020). Deeptxfinder: Multiple transmitter localization by deep learning in crowdsourced spectrum sensing. In *ICCCN*.