Localization Using THz with Machine Learning

Mohamad Lakkis*, Ahmad Dkhan†, and Hadi Mchawrab‡

*American University of Beirut, Beirut, Lebanon, Email: mnl03@mail.aub.edu

†American University of Beirut, Beirut, Lebanon, Email: amd53@mail.aub.edu

‡American University of Beirut, Beirut, Lebanon, Email: him06@mail.aub.edu

Abstract

Terahertz (THz) frequency bands have gained significant attention in the field of wireless communications due to their potential to enable ultra-high data rates and support emerging applications. Among the central challenges in realizing THz-based systems is accurate device localization, which is essential for optimizing network performance, ensuring seamless connectivity, and facilitating context-aware services. Machine Learning (ML) techniques, and Artificial Intelligence (AI) more broadly, offer promising strategies to address these challenges by learning complex propagation patterns and tailoring localization algorithms to the unique characteristics of THz signals.

This report presents our ongoing efforts to improve THz localization accuracy through ML-driven approaches. Over the course of the semester, we have examined and replicated a variety of models and algorithms, building upon a foundation of established results reported in the literature. These explorations provided critical insights into the limitations and advantages of existing methods. Leveraging these insights, we are now in the process of developing a novel algorithm designed to more effectively capture the subtle spatial features of THz signals. While this is not a finalized publication, the work described herein marks a significant step forward in our research. It serves as a summary of our progress, methodologies tested, and preliminary findings, as well as an indication of the promising direction we are heading to enhance THz localization accuracy using cutting-edge ML techniques.

Index Terms

Localization, THz, Machine Learning, Wireless Communication

I. Introduction

Localization is a critical aspect of modern wireless communication systems. The advent of THz technology offers new opportunities for high-precision localization due to its high frequency and large bandwidth. This paper explores the integration of ML and AI techniques to enhance localization accuracy using THz signals.

II. Related Work

Understanding the challenges of localization in Terahertz (THz) communication systems requires a thorough exploration of existing methods. To this end, we began by replicating various models from the literature to comprehend the problem's intricacies and the diversity of approaches. A notable work in this domain is [1], which introduces

a novel combination of Discrete Fourier Transform Spread Orthogonal Frequency Division Multiplexing (DFT-s-OFDM) and Deep Learning for joint communication and sensing in THz Integrated Sensing and Communication (ISAC) systems.

A. SI-DFT-s-OFDM Overview

The authors of [1] propose Sensing Integrated DFT-s-OFDM (SI-DFT-s-OFDM), which enhances traditional OFDM for dual-purpose operation: efficient data transmission and precise sensing of environmental parameters, such as range and velocity. This dual functionality is essential for applications like radar, localization, and high-resolution imaging.

B. Quantities Sensed

- **Range (Distance):** Measures the distance between the receiver and target.
- **Velocity:** Determines the relative speed of the target concerning the receiver.

C. AI Integration for Sensing

Deep learning, through a model named *SensingNet*, plays a pivotal role in enhancing sensing performance by overcoming challenges such as Doppler effects and Non-Line-of-Sight (NLoS) propagation. The key contributions include:

- **Range Estimation:** Utilizing block-wise input processing.
- **Velocity Estimation:** Employing subcarrier-wise input processing to address channel variations.

The supervised learning approach minimizes Mean Squared Error (MSE) between predicted and true values for range and velocity.

D. Joint Communication and Sensing

SI-DFT-s-OFDM leverages a single transmitted signal for both sensing and communication tasks. The system achieves millimeter-level range accuracy and decimeter-per-second velocity precision, demonstrating its applicability in 6G networks and high-speed applications like VR.

E. Technical Challenges and Mitigations

- **High Free-Space Propagation Loss:** Addressed using directional antennas to focus energy and mitigate attenuation.
- **Reflection and Scattering Losses:** Overcome by optimizing antenna design and signal processing.
- **Doppler Shift and RF Front-End Impairments:** Mitigated with deep learning compensation techniques.

F. System Design

At the Transmitter:

- Inserts reference signals for sensing and channel estimation.
- Applies DFT for frequency mapping, followed by IDFT and a Cyclic Prefix (CP) for robust transmission.

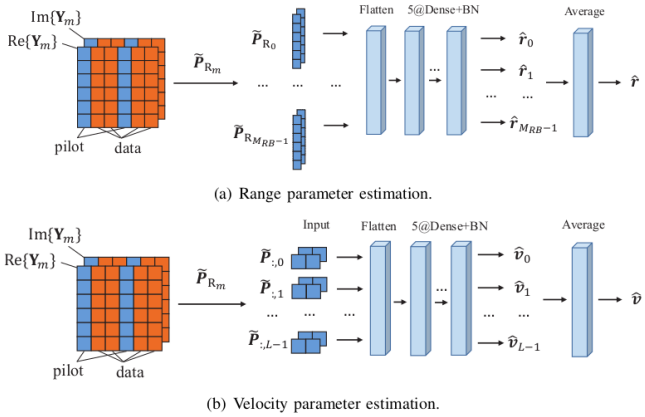
At the Receiver:

- Removes the CP and reconstructs the signal via DFT.
- Sensing parameters are estimated using reference signals, while transmitted data is recovered with channel equalization and deep learning.

G. Significance and Broader Implications

The proposed SI-DFT-s-OFDM framework integrates AI and advanced signal processing to achieve unprecedented sensing and communication performance in THz systems. These insights directly inform our efforts to develop novel algorithms for enhanced localization accuracy. Building on such foundational research, our work aims to address the unique challenges of THz communication and sensing, contributing to the broader advancement of this emerging technology.

H. Our Replication Efforts



(a) Range parameter estimation.

(b) Velocity parameter estimation.

Fig. 4. The structure of the proposed SensingNet network for DL-based sensing receiver.

Fig. 1. Replication of "The structure of the proposed SensingNet network for DL-based sensing receiver." [1]

We have successfully replicated the structure of the SensingNet network proposed in [1], as shown in Figure 1. In addition to replicating the sensing model, we replicated the first level model of the channel estimation model.

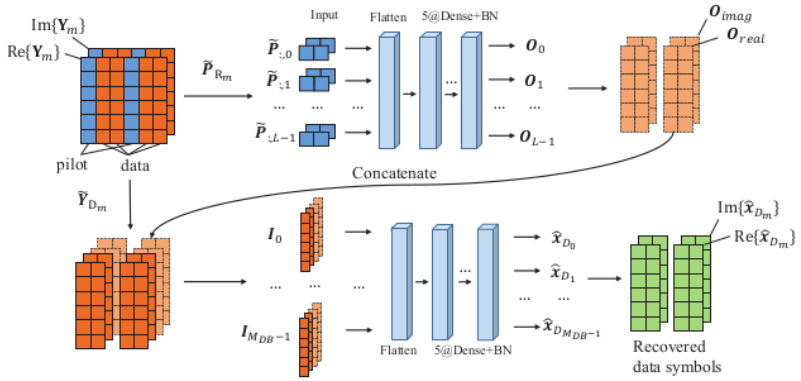


Fig. 5. The structure of the proposed two-level ComNet network for DL-based communication receiver.

Fig. 2. Replication of "The structure of the first level model of the channel estimation model." [1]

More formally: As a deep learning-based communication receiver, the two-level ComNet network shown in Fig. 2 is designed to recover data symbols while improving robustness to Doppler effects and phase noise. In the developed ComNet structure, the first level focuses on extracting channel information at the data blocks from the received reference blocks. Subsequently, the received data symbols and the output of the first-level network are concatenated and input into the second-level network. The second-level network is then tasked with approximating the relationship between the received data symbols and the transmitted data symbols, ultimately outputting the recovered data.

I. Network Structure of the First Level

In Fig. 2, the first-level sub-network of the ComNet consists of an input layer followed by five dense layers. This sub-network performs velocity estimation using the SensingNet model. Specifically, the input layer conducts subcarrier-wise input processing of the received reference signal, where each input fully encapsulates the channel observation at each subcarrier.

Following the input layer, five dense layers are utilized to extract the channel Doppler features from the reference blocks and derive the channel frequency response information at the data blocks. The layer sizes of these dense layers are 500, 250, 120, 60, and $2M_{\mathrm{DB}}$, respectively. The final dense layer produces $2M_{\mathrm{DB}}$ values for each input, representing the real and imaginary components of the Channel Frequency Response (CFR) at the data signals.

Let $g^{(i)}$ denote the activation function of the i-th dense layer. The n-th output of the first sub-network can be represented as:

$$O_n = g^{(5)}\left(g^{(4)}\left(\cdots g^{(1)}\left(\begin{bmatrix}\mathrm{Re}\{\hat{P}_{i,n}\}\\ \mathrm{Im}\{\hat{P}_{i,n}\}\end{bmatrix}\right)\cdots\right)\right), \tag{1}$$

where $O_n \in \mathbb{R}^{2M_{\mathrm{DB}}\times 1}$, $n = 0, 1, \ldots, L-1$. Here, f_{tanh} is used as the activation function for the last dense layer, while f_{ReLU} is used for the other layers.

The L outputs from the first sub-network are reshaped and concatenated into two matrices:

$$\mathbf{O}_{\text{real}} = [O'_0, O'_1, \ldots, O'_{L-1}]^T, \tag{2}$$

$$\mathbf{O}_{\text{imag}} = [O''_0, O''_1, \ldots, O''_{L-1}]^T, \tag{3}$$

where $O'_n \in \mathbb{R}^{M_{\text{DB}} \times 1}$ and $O''_n \in \mathbb{R}^{M_{\text{DB}} \times 1}$ correspond to the first and second halves of O_n, respectively.

One Important observation: the way they did it is relied purely on AI to estimate the channel for data extraction, rather than traditional methods where they get the channel at the pilot points and then rely on extrapolation to get the channel at the data points.

Note: The code is Available upon request.

One interesting concept that we came across from [1] is the fact that they are using shared hidden layers between two models, the first model focusing on range estimation and the second on velocity estimation. This is a very interesting concept because it allows the model to learn features common to both tasks, which can help the model to generalize better.

Building on this idea, we explored related work on shared hidden layers in multi-task learning. One notable paper we reviewed was [2], where the authors proposed a Multi-task Deep Neural Network with Shared Hidden Layers (MT-SHL-DNN) to tackle the challenges of acoustic emotion recognition. By leveraging shared feature transformations across different emotion representations while associating separate output layers to each emotion database, they were able to aggregate data from diverse sources without significant information loss. This approach demonstrated superior performance compared to single-task DNNs, effectively exploiting label interdependencies and overcoming data scarcity issues.

In our case, using shared hidden layers between range and velocity estimation models offers several advantages:

- **Feature Reusability:** The shared hidden layers enable both models to reuse learned features, reducing redundancy and computational overhead.
- **Improved Generalization:** By learning common patterns between the tasks, the model is better equipped to generalize to new, unseen data.
- **Data Efficiency:** Shared layers reduce the need for task-specific training data, as the shared layers leverage information from both tasks.
- **Reduced Model Complexity:** Sharing layers simplifies the overall architecture by consolidating parameters, leading to faster training and inference times.
- **Enhanced Robustness:** By learning correlated features, the shared layers help mitigate noise and inconsistencies in the input data for either task.

These benefits make the shared hidden layer approach particularly attractive for joint range and velocity estimation, where both tasks rely on overlapping signal features.

Another interesting concept that we came across for solving the problem of *Channel Estimation* is the use of the actual weights of the network as the channel itself, this concept is proposed in [3].

*J. **Analysis of the Paper** [3]*

Introduction

The paper focuses on the application of machine learning (ML) techniques for **channel estimation (CE)** in **Terahertz (THz) communication systems**, which are critical for future wireless networks such as 6G. Channel estimation is necessary for predicting how signals will propagate through the environment, particularly in challenging scenarios where high data rates and low-latency communications are required. THz signals face significant challenges due to their high frequency, which causes them to be sensitive to environmental factors such as reflection, scattering, and absorption.

The authors explore various machine learning algorithms to estimate the channel, comparing techniques such as **Neural Networks (NN)**, **logistic regression (LR)**, and **Projected Gradient Ascent (PGA)**. Among these, the results show that PGA provides the best performance, especially under low Signal-to-Noise Ratio (SNR) conditions, which makes it a promising candidate for THz channel estimation.

Challenges in THz Channel Estimation

The estimation of channel response parameters $\mathbf{H}$ in THz communication systems is particularly challenging due to the nature of THz signals. These signals operate at frequencies between 300 GHz and 10 THz, which means they have a very small wavelength and are susceptible to a variety of physical limitations such as high attenuation and molecular absorption. In addition, the hardware limitations, such as the need for high-speed **Analog-to-Digital Converters (ADCs)**, add to the complexity. These converters often operate with low resolution (e.g., 1-bit ADCs), which can introduce significant quantization errors.

Another challenge is that the **signal-to-noise ratio (SNR)** must be carefully managed to ensure reliable channel estimation. THz systems operate in environments where thermal noise is substantial due to the large bandwidths involved. Therefore, traditional channel estimation techniques, which rely on pre-defined models, may not be adequate. Instead, the authors explore **data-driven approaches** using machine learning to improve the accuracy of channel estimation.

Handling Complex Numbers

Since the transmitted and received signals in THz systems are complex-valued, the authors of the paper address this by converting the complex definitions into real-valued representations. This conversion is necessary because most machine learning algorithms operate on real numbers.

- The received noisy symbols $\mathbf{y}_n$, the channel matrix $\mathbf{H}$, and the transmitted signal $\mathbf{x}_n$ are all split into their real and imaginary components. For example, $\mathbf{y}_n = [\text{Re}(\mathbf{y}_n), \text{Im}(\mathbf{y}_n)]$, and similar decompositions are applied to the channel matrix and the transmitted signal. - After this decomposition, the real and imaginary parts are processed as separate real-valued inputs, which allows machine learning algorithms to process the data effectively without handling complex numbers directly.

This approach simplifies the problem, allowing the system to operate in a real-valued framework while still preserving the necessary information from the original complex-valued signals.

Neural Network-Based Channel Estimation

In the **Neural Network (NN) architecture** proposed by the authors, the input signals $\mathbf{x}_n$ are passed through multiple neurons, each of which applies a weighted transformation followed by a non-linear activation function. The weights of the network correspond to the channel response parameters $\mathbf{H} = W$, and the bias terms correspond to the noise z. Note that for now assume that $\mathbf{H}, W$ are real valued. We will explain later in depth on how to handle the complex numbers.

The neural network aims to learn the mapping between the input feature $\mathbf{x}_n$ and the output signal $\mathbf{y}_n$. The transformation is represented by the equation:

$$y_n = f(H_i x_n + z_i)$$

where $f(\cdot)$ is the activation function applied to the weighted sum of inputs. Different activation functions are explored, including the **sign function** sign(x), which helps map the network's output to binary values, making it suitable for decision-making processes such as decoding binary signals.

The training of the neural network involves optimizing the weights $\mathbf{H}_i$ and bias terms z_i to minimize the **loss function**. They chose the loss function which measures the difference between the predicted output $\mathbf{y}_n$ and the expected output $\mathbf{y}_n^e$ as follows:

$$[\hat{H}, \hat{z}] = \underset{H,z}{\operatorname{argmin}}[\mathbb{J}(H, z)]$$

where

$$\mathbb{J}(H, z) = \frac{1}{N} \sum_{n=1}^{N} \|y_n - tanh(H_i x_n + z_i)\|_2^2 + \lambda \|H\|_2^2$$

where λ is the regularization parameter.

Note that during the training, the activation function used is the sign function.(so while calculating the loss we are not directly looking at our outputs, because what we care about explicitly is H and not being able to retrieve the output from H, but of course being able to retrieve the output from H is crucial to make it accurate, but here we are just noting on why the loss function doesn't directly use the outputs of the NN)

Once the loss is computed, the weights $\mathbf{H}_i$ are updated using **gradient descent**, where the gradients of the loss function with respect to the weights are calculated, and the weights are updated accordingly:

$$H_i \leftarrow H_i - \eta \nabla_{H_i} \text{Loss}$$

Here, η is the learning rate, which controls the step size of the gradient descent update.

Now I will provide the dimension of each of these entries, to make it clear what are using in each case:

- Note $M_r = N = M_t$

- $\mathbf{y}_n$ is of dimension $M_r \times 1$, where M_r is the number of receive antennas.

- $\mathbf{H}_i$ is of dimension $1 \times M_r$, where M_t is the number of transmit antennas.

- $\mathbf{x}_n$ is of dimension $M_t \times 1$, where M_t is the number of transmit antennas.

- $\mathbf{z}_i$ is of dimension $M_r \times 1$, where M_r is the number of receive antennas.

-$H_i = [H_{i,1}, ..., H_{i,N}]$

- H is $M_r \times M_t$

Note that the the received unquantized signals $\boldsymbol{r}$ are then converted to $\boldsymbol{y} = \hat{g}(\boldsymbol{r}) = Sign(Re(\boldsymbol{r})) + jsign(Im(\boldsymbol{r}))$, and so this is the $\boldsymbol{y}$ that we working with.

Now we are arrive exactly on how we are handling these complex numbers in the neural network.: So the training on a high level is the one we just described, but how are we handling the complex numbers in the neural network? So first let us start with the form of the matrices $W = W_r + iW_i$, $y = y_r + iy_i$, $b = b_r + ib_i$, $x = x_r + ix_i$.
What we care about as like any NN is:

$$f(Wx + b) = f((W_r + iW_i)(x_r + ix_i) + b_r + ib_i) = f((W_r x_r - W_i x_i) + i(W_r x_i + W_i x_r) + b_r + ib_i)$$

And in this particular example: $W = H, b = z$, the values that we actually care about to do the channel estimation are the actual weights and biases. And so from now on we process these numbers as real numbers, and we don't care about the complex numbers, so in other words, at each step we are keep tracking of two things the real part and the imaginary part, nothing hard to grasp. And of course while backpropagating we are essentially doing the same thing, we are just updating the real part and the imaginary part separately.
Lastly, regarding the loss the way we are computing it is

$$Loss = \frac{1}{N}\sum_{n=1}^{N}(|y^e_{r,n} - tanh(x_{r,n} \times H_{r,n} + z_{r,n})|_2^2 + |y^e_{i,n} - tanh(x_{i,n} \times H_{i,n} + z_{i,n})|_2^2)$$
$$+\lambda(|H_r|_2^2 + |H_i|_2^2)$$

Where the subscript r and i denote the real and imaginary parts respectively. And n denotes the n^{th} sample.

Log-Likelihood Minimization

The paper also explores the use of **log-likelihood minimization** as a method for optimizing the channel estimation. In this approach, the log-likelihood function is derived from the assumption that the real and imaginary components of the received signal are modeled by a Gaussian distribution. The loss function based on log-likelihood is designed to estimate a transformed variable $\mathbf{X}^{Re}$, where:

$$\hat{X}^{Re} = \underset{X^{Re}}{\operatorname{argmin}} \sum_{i=1}^{M_t}\sum_{j=1}^{M_r} \left[\mathbf{1}[y^{Re}_{i,j} = 1]\log(\phi(X^{Re}_{i,j}/\sigma)) + \mathbf{1}[y^{Re}_{i,j} = -1]\log(1 - \phi(X^{Re}_{i,j}/\sigma))\right]$$

where $\phi(\cdot)$ is the cumulative distribution function (CDF) of the standard normal distribution, and σ represents the standard deviation.

By minimizing this log-likelihood function, the model attempts to estimate the channel parameters that maximize the probability of the observed binary outputs.

Projected Gradient Descent (PGD)

Now we will go through the **Projected Gradient Descent (PGD)** algorithm. This algorithm is usually used when we have a constrained problem:

Unconstrained minimization

$$\min_{x \in \mathbb{R}^n} f(x).$$

- All $x \in \mathbb{R}^n$ is feasible.
- Any $x \in \mathbb{R}^n$ can be a solution.

Constrained minimization

$$\min_{x \in Q} f(x).$$

- Not all $x \in \mathbb{R}^n$ is feasible.
- Not all $x \in \mathbb{R}^n$ can be a solution.
- The solution has to be inside the set Q.

An example:

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 \quad \text{s.t.} \quad \|x\|_2 \leq 1$$

can be expressed as

$$\min_{\|x\|_2 \leq 1} \|Ax - b\|_2^2.$$

Here $Q := \{v \in \mathbb{R}^n : \|v\|_2 \leq 1\}$ is known as the unit ℓ_2 ball.

Let us recap for a moment how gradient Descent Works for Unconstrained Problems:

Reminder of the problem:

$$\min_{x \in Q} f(x).$$

- Starting from an initial point $x_0 \in \mathbb{R}^n$, **GD** iterates the following until a stopping condition is met:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k),$$

- $k \in \mathbb{N}$: the current iteration counter
- $k + 1 \in \mathbb{N}$: the next iteration counter
- x_k: the current variable
- x_{k+1}: the next variable

- ∇f is the gradient of f with respect to differentiation of x (i.e. $\nabla f = \frac{\partial f}{\partial x}$)
- $\nabla f(x_k)$ is the ∇f at the current variable x_k
- $\alpha_k \in (0, +\infty)$: gradient stepsize, could be fixed or learned at each iteration.
- Note: Here we are assuming that the function f is differentiable. If that is not the case, we can use subgradient descent.

Now the Question is Can we tune this algorithm to make it work for Constrained Problems?: And the answer is yes, the key is Euclidean projection operator. $\mathbb{R}^n \to \mathbb{R}^n$. Like before we will start with some assumptions:

- f is assumed to be continously differentiable. (i.e. $f \in C^1$ or $\nabla f(x)$ exists $\forall x$)
- In general one also assumes that f is globally L-Lipschitz continuous. (i.e. $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$)
- $\emptyset \neq Q \subset \mathbb{R}^n$ is convex and compact
- The constraint is represented by a set Q
- $Q \subset \mathbb{R}^n$ means Q is a subset of $\mathbb{R}^n$, the domain of f
- $Q \neq \emptyset$ means Q is not an empty set. *it is not useful for discussion if* Q is empty
- Q is a convex set $\forall x \forall y \forall \lambda \in (0,1)\,(x \in Q, y \in Q \implies \lambda x + (1-\lambda)y \in Q)$
- Q is compact *compact = bounded + closed*

Now PGD = GD + Projection.

- Starting from an initial point $x_0 \in \mathbb{R}^n$, **PGD** iterates the following until a stopping condition is met:
- $x_{k+1} = \text{Proj}_Q(x_k - \alpha_k \nabla f(x_k))$
- Proj_Q is the Euclidean projection operator onto Q, which maps a point in $\mathbb{R}^n$ to the closest point in Q.(i.e. $\text{Proj}_Q(x) = \underset{y \in Q}{\text{argmin}} \|x - y\|_2$)
- $\alpha_k \in (0, +\infty)$: gradient stepsize, could be fixed or learned at each iteration.
- $k \in \mathbb{N}$: the current iteration counter
- $k + 1 \in \mathbb{N}$: the next iteration counter
- x_k: the current variable
- x_{k+1}: the next variable
- ∇f is the gradient of f with respect to differentiation of x (i.e. $\nabla f = \frac{\partial f}{\partial x}$)

Projected Gradient Ascent (PGA)

This is the same as PGD, but instead of minimizing the function, we are maximizing it. (so in ML, we changed the objective function, not the same ONE we were trying to minimize, because that doesn't make sense!)
And instead of taking a step in the opposite direction of the gradient, we are taking a step in the direction of the gradient.
Lastly Note that the proj function stays the same, if the point is outisde our desired set, we need to projected back by finding the closest point to it from this set. (In the Euclidean Projection we use Euclidean Distance!)

Relating this to what we are we doing in the paper:: Some Questions to answer:

- What is the constraint we are working under (i.e. what is the set Q)?
- How are we projecting this Matrix H that we got back to Q?

Very interesting questions, let's dive into them:

Our constrained Set Q: The constraint is that H should have a low rank $r \ll N$. In many real-world systems, the channels can often be well-approximated by matrices of much lower rank. For example, in multiple-input multiple-output (MIMO) systems, low-rank approximations are often used to reduce computational complexity. For more details, refer to the work in this paper [4] available at IEEE Xplore.

So the update step in the PGA algorithm is as follows:

$$H_{k+1} = H_k + \alpha_k \nabla f(H_k)$$

Where $\nabla f(H_k)$ is the gradient of the loss function with respect to H_k, and α_k is the step size.

But now we have a Problem How do we make sure this matrix $H_{k+1} \in Q$?

Projection Step to Q: There is many ways to project a matrix to a low-rank space, but one of the most known way is to use the **Singular Value Decomposition (SVD)**. Now I will explain how we are doing stuff for PGA. SVD is a matrix factorization technique that can decompose a matrix into its singular values and corresponding singular vectors. By keeping only the largest singular values (up to rank r), you can project the matrix back onto the space of low-rank matrices.

Simplex projection: Ensures the solution remains within certain constraints.(maybe H non-negative or any other constrain)

The SVD of a matrix H is given by:

$$H = U\Sigma V^T$$

where U and V are orthogonal matrices, and Σ is a diagonal matrix containing the singular values of H. To project H to a low-rank matrix, we keep only the r largest singular values and setting the rest to zero. This operation effectively reduces the rank of H to r. The projected matrix is then given by:

$$H_{\text{proj}} = U\Sigma_{\text{proj}} V^T$$

where Σ_{proj} is the diagonal matrix with only the r largest singular values. This projection step ensures that the updated matrix H_{k+1} remains within the low-rank constraint.

Note: requires projecting the updated solution back into the feasible set after each gradient step. Which might be computationally expensive. That is where Franke-Wolfe algorithm comes into play!

Franke-Wolfe Algorithm

The Frank-Wolfe algorithm (also called Conditional Gradient Descent) is particularly useful for solving convex optimization problems with constraints (a lot of the work in convex analysis is based on [5]). The algorithm works directly in the Q, meaning it doesn't find a solution in $\mathbb{R}^n$ and then projects it to Q, it immediatly find the solution in Q.

The algorithm works by solving a linear approximation of the objective function at each step, instead of directly taking a gradient step followed by a projection.

Given a convex objective function $f(x)$ and a convex constraint set C, the Frank-Wolfe algorithm seeks to minimize the objective function while ensuring that the solution stays within the feasible region C. The steps of the algorithm are as follows:

- **Input:** Initial point $x_0 \in C$, step size γ_t, and maximum iterations T.
- **For each iteration** $t = 0, 1, \ldots, T$**:**
  1) Compute the gradient of the objective function at the current point x_t:
  $$\nabla f(x_t)$$
  2) Solve the linear approximation of the objective function over the constraint set C:
  $$s_t = \arg\min_{s \in C} \nabla f(x_t)^T s$$
  3) Compute the step size γ_t (via line search or predefined):
  $$\gamma_t = \arg\min_{\gamma \in [0,1]} f((1-\gamma)x_t + \gamma s_t)$$
  4) Update the solution x_t by moving toward the new point s_t:
  $$x_{t+1} = (1-\gamma_t)x_t + \gamma_t s_t$$
- **Output:** The final solution x_T.

Special Case: Low-Rank Matrix Constraint

In the case where the objective function is defined over matrices and we impose a low-rank constraint on the matrix H, the algorithm is adapted as follows:

- **Input:** Initial matrix H_0 with rank constraint r, gradient step size γ_t, and maximum iterations T.
- **For each iteration** $t = 0, 1, \ldots, T$**:**
  1) Compute the gradient of the objective function with respect to the matrix H_t:
  $$\nabla f(H_t)$$
  2) Perform Singular Value Decomposition (SVD) on the gradient matrix and compute the top singular vector:
  $$\nabla f(H_t) = U \Sigma V^T$$
  $$\text{Top singular vector: } u_1 v_1^T$$
  3) Subtract the top singular vector from the gradient to avoid domination by the largest singular value:
  $$\nabla f(H_t)_{\text{modified}} = \nabla f(H_t) - u_1 v_1^T$$

4) Solve the linear subproblem to find the next point S_t in the space of low-rank matrices:

$$S_t = \arg \min_{S \in \text{Rank}(r)} \nabla f(H_t)^T S$$

5) Update the matrix H_t with the computed step size γ_t:

$$H_{t+1} = (1 - \gamma_t) H_t + \gamma_t S_t$$

- **Output:** The final low-rank matrix H_T.

Conclusion

This paper provides a comprehensive comparison of various machine learning techniques for **THz channel estimation**. By converting the complex-valued channel estimation problem into a real-valued one, the authors are able to leverage powerful machine learning models such as neural networks and log-likelihood minimization techniques to achieve accurate channel estimation. The results indicate that methods like **Projected Gradient Ascent (PGA)** and **Neural Networks (NN)** perform well in challenging SNR conditions, making them suitable candidates for future THz communication systems.

For the code, it is available upon request.

K. Relating this to Our Work

This paper provided valuable insights, particularly on two critical aspects: handling complex numbers within neural networks and projecting matrices back to a low-rank space. These concepts are highly relevant to our work, as we also deal with complex-valued data and impose a low-rank constraint (as discussed in Section II-J). The methodologies outlined in this paper significantly enhanced our understanding and informed our approach to addressing these challenges in our research.

III. Methodology

From all of this paper work and analysis and replication, we can now start to build our own methodology.

A. Problem Formulation

Based on pilot signals received we care to find the location of the user. We have a set of pilot signals that are transmitted from the user to the base station. The base station receives these signals and estimates the exact location of the user.

B. Data Collection

The data is collected from a simulated environment, in matlab we simulate the environment and collect the data. And then our data will consists of the received signals at the base station, and the target location of the user(x,y,z).

C. Model Architecture

We are currently working on different models to estimate the location of the user.

1) Model 1: Range Estimation: This model is designed to estimate the range of the user based on signals received from multiple antennas. Each antenna independently calculates the distance to the user, which represents the radius of a sphere centered at the antenna's location. Since the user must lie somewhere on the surface of this sphere, this calculation defines the possible locations of the user relative to each antenna.

By performing this process for all antennas in the system, we obtain multiple overlapping spheres, each representing the set of possible user locations as determined by an individual antenna. To pinpoint the exact location of the user, the intersection of these spheres is computed. Geometrically, the intersection of these spheres identifies a unique point in three-dimensional space that satisfies all range constraints simultaneously.

This approach leverages the spatial diversity provided by multiple antennas, significantly improving the accuracy of the user's location estimate. The method assumes that the antennas are deployed in a configuration that ensures sufficient overlap of the spheres, which is crucial for resolving ambiguities and achieving a precise intersection point. Additionally, the accuracy of this technique depends on the precision of the range measurements, which can be affected by factors such as noise, signal attenuation, and multipath effects.

2) Model 2: Direct Position Estimation (Two-Step Process): This model is designed to directly estimate the user's position in three-dimensional space based on the received signals. The estimation process involves two main steps: first, determining the user's approximate position in the space, along with the confidence interval associated with this estimation.

Next, the confidence interval is incorporated into the optimization process using a Lagrange multiplier to refine the position estimate. This refinement is achieved by minimizing a convex function, such as the Mean Squared Error (MSE) [5], which ensures a more accurate and reliable position estimate.

For instance, suppose the initial output of the model is the user's position (x, y, z) with a confidence range of ± 5 meters. In the second step, the Lagrange multiplier method is applied to minimize the MSE function, leveraging the confidence range to constrain the optimization. This process results in a refined position estimate that better aligns with the received signals and the confidence bounds, thereby improving the overall accuracy of the user's location.

IV. Current Work and Conclusion

We are currently focused on implementing and evaluating the two models described in Sections III-C1 and III-C2. The primary objective is to compare the performance of these models in estimating the user's position and validate their accuracy under various conditions.

In addition to model implementation, we are rigorously preparing the data to ensure reliable results. This includes applying **Exploratory Data Analysis (EDA)** techniques [6] to clean, visualize, and identify key patterns within the data. Through EDA, we aim to detect and address potential anomalies, inconsistencies, and missing values, which could otherwise adversely impact the models' performance.

A critical aspect of our current work is the use of the **Frank-Wolfe algorithm** to project the estimated channel matrix back to a low-rank space. This step ensures that the matrix satisfies the low-rank constraints imposed by the problem while maintaining computational efficiency. The Frank-Wolfe algorithm offers an advantage over traditional projection methods by performing optimization directly within the feasible region, reducing the computational cost associated with explicit projection steps.

To evaluate the effectiveness of the proposed models and the low-rank projection method, we are benchmarking our results against the **Least Squares Estimation (LSE)** technique. This comparison serves as a baseline, allowing us to assess the relative performance improvements offered by our methods. By contrasting the outputs of the two models with LSE, we aim to quantify the accuracy and robustness of the proposed approaches, particularly in scenarios with noise, incomplete data, or complex signal propagation environments.

As we progress, our goal is to refine the models and methodologies further, ensuring that they can generalize effectively to real-world scenarios and provide reliable position estimates in challenging communication environments.

REFERENCES

[1] Y. Wu, F. Lemic, C. Han, and Z. Chen, "Sensing Integrated DFT-Spread OFDM Waveform and Deep Learning-powered Receiver Design for Terahertz Integrated Sensing and Communication Systems," *arXiv preprint arXiv:2109.14918*, 2021. Available at: https://doi.org/10.48550/arXiv.2109.14918.

[2] Y. Zhang, Y. Liu, F. Weninger, and B. Schuller, "Multi-task deep neural network with shared hidden layers: Breaking down the wall between emotion representations," in *Proc. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4990–4994. doi: 10.1109/ICASSP.2017.7953106.

[3] A. Author, B. Author, and C. Author, "Paper Title," *arXiv preprint arXiv:2104.08122*, 2021. Available at: https://doi.org/10.48550/arXiv.2104.08122.

[4] N. J. Myers, K. N. Tran, and R. W. Heath, "Low-rank mmwave mimo channel estimation in one-bit receivers," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 5005–5009. doi: 10.1109/ICASSP40776.2020.9054484.

[5] A. Ang, "Special classes of function in optimization in machine learning (alternative title – some basic convex analysis for optimization)," ECS, University of Southampton, UK. Available at: https://angms.science.

[6] M. Komorowski, D. Marshall, J. Salciccioli, and Y. Crutain, "Exploratory Data Analysis," in *Secondary Analysis of Electronic Health Records*, Springer, 2016, pp. 185–203. doi: 10.1007/978-3-319-43742-2_15.