# Communication-Efficient Cooperative Localization: A Graph Neural Network Approach

Yinan Zou, Christopher G. Brinton, and Vishrant Tripathi
Elmore Family School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47907, USA

***Abstract*—Cooperative localization leverages noisy inter-node distance measurements and exchanged wireless messages to estimate node positions in a wireless network. In communication-constrained environments, however, transmitting large messages becomes problematic. In this paper, we propose an approach for communication-efficient cooperative localization that addresses two main challenges. First, cooperative localization often needs to be performed over wireless networks with loopy graph topologies. Second is the need for designing an algorithm that has low localization error while simultaneously requiring a much lower communication overhead. Existing methods fall short of addressing these two challenges concurrently. To achieve this, we propose a vector quantized message passing neural network (VQ-MPNN) for cooperative localization. Through end-to-end neural network training, VQ-MPNN enables the co-design of node localization and message compression. Specifically, VQ-MPNN treats prior node positions and distance measurements as node and edge features, respectively, which are encoded as node and edge states using a graph neural network. To find an efficient representation for the node state, we construct a vector quantized codebook for all node states such that instead of sending long messages, each node only needs to transmit a codeword index. Numerical evaluations demonstrates that our proposed VQ-MPNN approach can deliver localization errors that are similar to existing approaches while reducing the overall communication overhead by an order of magnitude.**

## I. Introduction

Localization of node positions is an essential functionality for many wireless networks. In harsh environments, such as underwater networks, caves, and dense forests, satellite-based positioning systems such as GPS/GNSS are often inaccessible, requiring alternative methods for position estimation. In such settings, there are usually a few anchor nodes with known fixed positions and numerous agent nodes with unknown positions. To utilize this structure, *cooperative localization* methods [1], [2] have been proposed, where agent nodes localize themselves using inter-node measurements and exchanged messages. Cooperative localization is more flexibile, scalable, and robust than centralized implementations and plays a crucial role in modern sensing systems.

### A. Cooperative Localization: Overview and Challenges

Cooperative localization typically involves two phases: a distance measurement phase and a location-update phase. First, during the distance measurement phase, each node measures the inter-node distance using metrics such as received signal strength (RSS), time of arrival (TOA), or round-trip time of arrival (RTOA). Then, during the location-update phase, each node exchanges messages with distance measurements and estimated positions over multiple iterations, eventually using all the information to infer its own position.

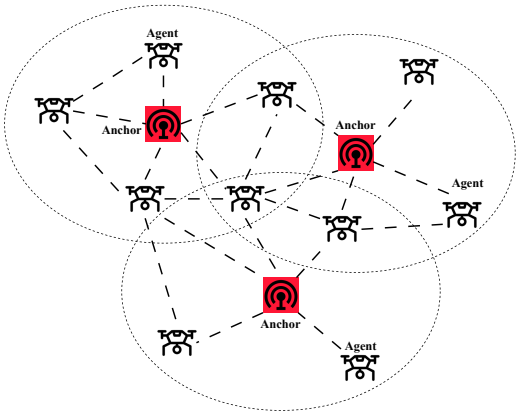



Fig. 1. The cooperative localization framework consists of a few anchors with fixed known locations and many agents with unknown locations. Both anchors and agents communicate with neighboring nodes to measure relative distances and then exchange messages iteratively to perform global localization.

Belief propagation (BP) [3]–[5] and distributed optimization [6]–[8] are two widely used approaches for the location-update phase. The distributed optimization methods are non-Bayesian and treat agents' positions as deterministic parameters while disregarding prior information. In contrast, BP, as a Bayesian method, treats agents' positions as random variables, which allows for the quantification of estimation uncertainties. BP can be categorized into two types based on the representation of messages and beliefs: parametric BP and nonparametric BP. Parametric BP typically employs the mean and covariance of Gaussian distributions to represent messages and beliefs. In contrast, nonparametric BP employs thousands of weighted samples to depict messages in detail, enabling the representation of distributions with arbitrary shapes. For tree-structured graphs, BP works well and can provide the precise marginal posterior. However, physical graphs existing in cooperative localization networks often include many loops. For loopy graphs, BP can only obtain an approximation of the true marginal posterior, leading to suboptimal estimates [9]–[13]. Existing BP-based methods employ approximation techniques to relax the original problem into a more tractable form, which typically yields suboptimal solutions.

During the location-update phase, agent nodes need to exchange message with each other over multiple iterations. In nonparametric BP, iterative transmission of messages rep-

ISBN 978-3-903176-73-7 © 2025 IFIP

resented by weighted particles often encounters a fundamental communication bottleneck due to limited spectrum resources [14], necessitating efforts to reduce communication overhead. Compression techniques, such as quantization [15], Gaussian mixture models [4], [16], and partial mutual information [17], can be used to compress messages in nonparametric BP, which however introduce unavoidable compression errors.

Compressing messages to reduce communication overhead while maintaining localization performance thus remains an open problem. To address this question, in this work, we investigate data-driven techniques. In particular, graph neural networks (GNNs) have recently been utilized for graph-related tasks to rectify errors in loopy belief propagation [13], [18]–[21]. By exchanging information between nodes, GNNs effectively capture topological dependencies. Given that neural networks excel at function approximation, we hypothesize that GNNs can be employed to learn the mapping from distance measurements to actual node positions, which has the potential to improve cooperative localization performance.

### B. Our Approach: Overview and Contributions

In this paper, we study communication-efficient cooperative localization problem where agent nodes localize themselves using inter-node measurements and wireless messages exchanged under communication constraints. We address two main challenges. First, cooperative localization often needs to be performed over wireless networks with loopy graph topologies. Second, we need to design an algorithm that has low localization error while simultaneously requiring a much lower communication overhead. Existing methods fall short of concurrently addressing these two challenges. To simultaneously deal with these two challenges, we propose a vector quantized message passing neural network (VQ-MPNN). Through end-to-end neural network training, VQ-MPNN enables the co-design of node localization and message compression. Our approach can handle loopy graphs (unlike belief propagation based methods) while having a communication overhead that is an order of magnitude less than the best known deep learning based methods. A primary distinction between our proposed VQ-MPNN and MPNN approaches in existing works [19]–[21] lies in the construction of a compact VQ codebook for message representation, which reduces the communication cost to just a few bits per message.

Our main contributions are summarized as follows:

- We formulate VQ-MPNN to treat prior node positions and distance measurements as node and edge features, respectively, and design its graph neural network to encode these features as node and edge states. During each communication round, then, nodes engage in cooperative message-passing to update their node and edge states. After several communication rounds, the final node state can be fed into the estimation neural network to compute the estimated node position.
- To find an efficient representation for the node state, we construct a vector quantized (VQ) codebook for all node states. The VQ codebook is regarded as a trainable parameter and is constructed during the neural network training phase. Each node only needs to transmit the codeword index rather than the node state, which we find *reduces communication cost by an order of magnitude*.
- Benefiting from the message passing mechanism, the proposed VQ-MPNN approach can perform inference on the graph corresponding to the network topology in a distributed manner. Importantly, even if nodes are added or removed from the graph, the proposed neural network still works, which demonstrates its strong generalizability and scalability.
- Numerical evaluations demonstrate that our proposed approach outperforms or matches existing solutions like PLBP [22] and SP-ADMM [8] in localization accuracy, while drastically improving communication efficiency.

### C. Related Work

In recent years, numerous studies have focused on cooperative localization. Given the distributed nature of computation, message passing algorithms based on probabilistic graphical models [3]–[5] and distributed optimization methods [6]–[8] have been utilized for cooperative localization in prior works. In the distributed methods, each node localizes itself using local network information from its neighbors rather than relying on data from the entire network, making these methods well-suited for large-scale networks.

In the Bayesian framework, BP can be employed to obtain the marginal posterior distribution of a node's position. BP runs on the factor graph to calculate the marginal posteriors by passing messages between nodes. Since the distance measurement model is usually non-linear, parametric BP approximates the non-linear model using Taylor expansions [9], [23], statistical linear regression [22], or variational posterior approximation [24]. Typically, all the BP messages and beliefs are derived in Gaussian form, with only their means and covariances being updated and exchanged during the process. However, the approximation of nonlinear models via linear ones leads to model mismatch and accuracy issues, thereby degrading localization performance.

For nonparametric BP, the authors in [4], [25], [26] use weighted particles to represent messages and beliefs, enabling the flexible representation of distributions in any shape. The iterative broadcasting and aggregation of particles result in significant communication costs. To reduce the communication cost of particle transmission, the authors in [17] proposes selecting important particles for transmission based on partial mutual information. The authors in [4], [16] utilize the Gaussian mixture model to approximate the particles, enabling the exchange of only the mixture representation parameters among agents. Although the aforementioned works utilize compression techniques to minimize communication cost, they inevitably introduce compression errors. Further, both parametric and nonparametric BP methods struggle to perform well on topologies with loops.

Recently, graph neural networks (GNNs) have demonstrated excellent performance in various graph-related tasks, including

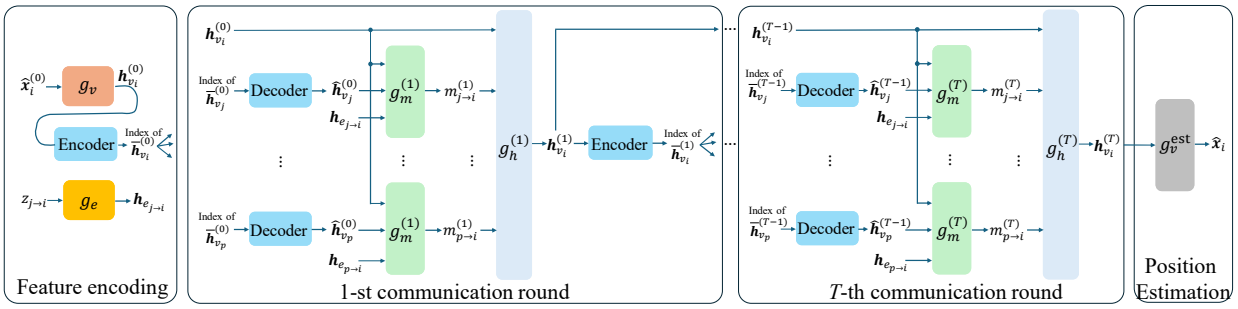



Fig. 2. VQ-MPNN architecture. Each node $i$ takes its prior position and distance measurements as inputs, encoding them into node and edge states. Then, each node applies the encoder to map its node state to a codeword index, which is transmitted to neighboring nodes. During each communication round, node $i$ receives the codeword indices from its neighbors and reconstructs them into node states. Each node then integrates these reconstructed node states with its own node and edge states to update its node state. After several communication rounds, each node uses its final updated node state to estimate its position.

cooperative localization. The authors in [19], [21] introduce the message passing neural network (MPNN) into nonparametric and parametric BP to address the cooperative localization problem modeled by loopy factor graph. By considering the temporal correlation in the motion model of agents, [20] combines a long short-term memory (LSTM) network with a MPNN for agent tracking. Motivated by these works, we propose a GNN based approach that can simultaneously handle loopy graphs and have much lower communication overhead.

## II. System Model

Consider a wireless network consisting of $N_s$ anchor nodes with precisely known positions and $N_a$ agent nodes whose positions are unknown. The index sets of anchor and agent nodes are denoted as $\mathcal{I}_s$ and $\mathcal{I}_a$. The total set of nodes $\mathcal{I}$ is defined as $\mathcal{I} = \mathcal{I}_s \cup \mathcal{I}_a$. We denote $\boldsymbol{x}_i \in \mathbb{R}^2$ as the two dimensional position of node $i, \forall i \in \mathcal{I}$. We assume a known prior distribution on positions given by $p(\boldsymbol{x}_i)$. Each agent node receives noisy distance measurements to neighboring nodes within a communication range $r$ and communicates only with these neighboring nodes. It is assumed that each node is aware of its neighboring nodes. Fig. 1 illustrates the cooperative localization framework.

Cooperative localization comprises of two phases: the distance measurement phase and location update phase. In the distance measurement phase, each agent node obtains noisy distance measurement from the received pilot signals of neighboring nodes. Measurements can be obtained through various methods, with the most common approaches including RSS, TOA, or RTOA. When node $j \in \mathcal{N}_i$ transmits a pilot signal to node $i \in \mathcal{I}_a$, the distance measurement from node $j$ to $i$ is modeled as

$$z_{j\to i} = d(\boldsymbol{x}_j, \boldsymbol{x}_i) + n_{j\to i}, \quad j \in \mathcal{N}_i, \tag{1}$$

where $d(\boldsymbol{x}_j, \boldsymbol{x}_i) = \|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2$ denotes true distance between node $i$ and $j$, $\mathcal{N}_i \subseteq \mathcal{I}$ denotes the neighbors of node $i$, and $n_{j\to i}$ denotes measurement noise. Only node $i$ has access to the measurement $z_{j\to i}$. Despite the fact that $d(\boldsymbol{x}_j, \boldsymbol{x}_i) = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, the measurement $z_{j\to i}$ is not necessarily equal to $z_{i\to j}$ because $n_{j\to i}$ is not necessarily equal to $n_{i\to j}$. We assume that each node has already obtained distance measurements from its neighbors.

In the location-update phase, each node exchanges messages containing distance measurements and estimated positions with neighboring nodes over multiple iterations, and then uses all the collected information to infer its position. Since we do not have full network connectivity, we need multiple iterations of message exchanges to obtain all the information needed to infer global node positions accurately.

## III. Communication-Efficient Graph Neural Network: VQ-MPNN

In this section, we propose VQ-MPNN for cooperative localization. Fig. 2 shows the overall architecture.

### A. Graph Representation

We characterize the cooperative localization network as a directed graph, where the agent and anchor nodes are regarded as graph nodes and the distance measurements between nodes are regarded as edges. At initialization, the initial position $\hat{\boldsymbol{x}}_i^{(0)}$ is drawn from prior distribution $p(\boldsymbol{x}_i)$. Note that the node feature associated with node $i$ is the initial position $\hat{\boldsymbol{x}}_i^{(0)}$, while the edge feature associated with each edge from node $j$ to node $i$ is the distance measurement $z_{j\to i}$.

### B. VQ-MPNN Architecture

(1) *Feature Encoding*: We utilize Multi-Layer Perceptrons (MLPs) to map node and edge features into an $M$-dimensional unified space

$$\boldsymbol{h}_{v_i}^{(0)} = g_v(\hat{\boldsymbol{x}}_i^{(0)}), \tag{2}$$

$$\boldsymbol{h}_{e_{j\to i}} = g_e(z_{j\to i}), \tag{3}$$

where $g_v(\cdot) : \mathbb{R}^2 \to \mathbb{R}^M$ and $g_e(\cdot) : \mathbb{R}^1 \to \mathbb{R}^M$ are MLPs, $\boldsymbol{h}_{v_i} \in \mathbb{R}^M$ and $\boldsymbol{h}_{e_{j\to i}} \in \mathbb{R}^M$ denote the node state and the edge state, respectively. Subsequently, each node uses an encoder to map the $M$-dimensional node state to a $D$-dimensional codeword within the VQ codebook, which contains $K$ codewords, and retrieves the corresponding codeword index. The encoder is given by

$$\text{Index}(\bar{\boldsymbol{h}}_{v_i}^{(0)}) = \text{Encoder}(\boldsymbol{h}_{v_i}^{(0)}), \tag{4}$$

where $\text{Index}(\cdot)$ denotes the index of a codeword and $\bar{\boldsymbol{h}}_{v_i} \in \mathbb{R}^D$ denotes quantized node state. Note that the quantized node state is a codeword in the codebook. The codebook index is represented by binary code. For instance, in a VQ codebook with $K = 5$, the index of the

third codeword $\theta_3$ is represented as 010, as illustrated in Fig. 3. The design of the encoder will be elaborated in section III-C.

(2) *Inter-node Communication*: We denote the maximum number of communication rounds as $T$. In the communication round $t \in \{1, \ldots, T\}$, each node $i$ broadcasts the index of its quantized node state and receives the indices of quantized node states from its neighbors. This exchange of node states is analogous to the beliefs exchange in BP. Utilizing the index of the quantized node state, each node $i$ reconstructs the node state using a decoder as follows

$$\hat{\boldsymbol{h}}_{v_j}^{(t-1)} = \text{Decoder}(\text{Index}(\bar{\boldsymbol{h}}_{v_j}^{(t-1)})), \tag{5}$$

where $\hat{\boldsymbol{h}}_{v_j} \in \mathbb{R}^M$ denotes the recovered node state. The design of decoder will be elaborated in section III-C. Subsequently, we concatenate the edge state and the node states into a single vector and input this vector into a MLP to generate the message

$$\boldsymbol{m}_{j\to i}^{(t)} = g_m^{(t)}(\hat{\boldsymbol{h}}_{v_j}^{(t-1)}, \boldsymbol{h}_{v_i}^{(t-1)}, \boldsymbol{h}_{e_{j\to i}}), \tag{6}$$

where $\boldsymbol{m}_{j\to i} \in \mathbb{R}^M$ denotes the message, $g_m^{(t)}(\cdot) : \mathbb{R}^{3M} \to \mathbb{R}^M$ denotes a MLP, $\hat{\boldsymbol{h}}_{v_j}$ denotes the recovered node states of node $j$, $\boldsymbol{h}_{v_i}$ denotes the node state of node $i$, and $\boldsymbol{h}_{e_{j\to i}}$ denotes the edge state of edge from node $j$ to node $i$.

(3) *Combination*: In the communication round $t \in \{1, \ldots, T\}$, each node $i$ integrates its previous node state with messages to obtain the updated node state

$$\boldsymbol{h}_{v_i}^{(t)} = g_h^{(t)}(\boldsymbol{h}_{v_i}^{(t-1)}, \sum(\{\boldsymbol{m}_{j\to i}^{(t)}\}_{j\in\mathcal{N}_i})), \tag{7}$$

where $\sum(\cdot)$ denotes the summation function, $g_h^{(t)}(\cdot) : \mathbb{R}^{2M} \to \mathbb{R}^M$ denotes a MLP. Next, each node employs the encoder to generate the quantized node state for the subsequent communication round

$$\text{Index}(\bar{\boldsymbol{h}}_{v_i}^{(t)}) = \text{Encoder}(\boldsymbol{h}_{v_i}^{(t)}). \tag{8}$$

(4) *Position Estimation*: After $T$ communication rounds, we input the final node state into the estimation neural network to obtain the estimated node position

$$\hat{\boldsymbol{x}}_i = g_v^{\text{est}}(\boldsymbol{h}_{v_i}^{(T)}), \tag{9}$$

where $\hat{\boldsymbol{x}}_i \in \mathbb{R}^2$ denotes the estimated node position and $g_v^{\text{est}}(\cdot) : \mathbb{R}^M \to \mathbb{R}^2$ denotes a MLP.

Alg. 1 outlines the workflow of the proposed VQ-MPNN for communication-efficient cooperative localization.

### C. Encoder and Decoder Design

Motivated by [27], we construct a VQ codebook for node states that is shared among all nodes. In particular, the codebook is represented as $\mathcal{C} \in \mathbb{R}^{D\times K}$, where $D$ denotes the length of the codeword and $K$ denotes the number of the codewords. In other words, there are $K$ codewords, each with $D$ dimensions, represented as $\boldsymbol{\theta}_k \in \mathbb{R}^D, k \in \{1, \ldots, K\}$. All of the codewords are trainable parameters and are optimized during the training phase.

The process of utilizing encoder to compress node states is outlined as follows. For sake of simplicity, we omit the index of the communication round $t$. Specifically, we use a projection encoder to map the node state $\boldsymbol{h}_{v_i} \in \mathbb{R}^M$ to the latent vector $\tilde{\boldsymbol{h}}_{v_i} \in \mathbb{R}^D$ as follows

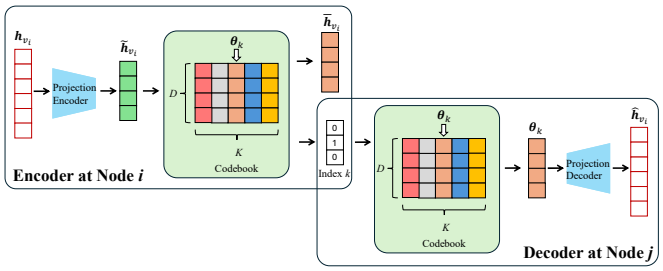



Fig. 3. Encoder and decoder for node states. Node $i$ employs the projection encoder to map its node state into a latent vector, determines the nearest codeword to the latent vector, and transmits the corresponding codeword index to its neighbors. Neighboring node $j$ receives the codeword index, retrieves the corresponding codeword from the codebook, and inputs this codeword into the projection decoder to recover the node state.

$$\tilde{\boldsymbol{h}}_{v_i} = \text{ProjEncoder}(\boldsymbol{h}_{v_i}), \tag{10}$$

where $\text{ProjEncoder}(\cdot) : \mathbb{R}^M \to \mathbb{R}^D$ is a MLP. Next, the nearest codeword to the latent vector $\tilde{\boldsymbol{h}}_{v_i}$ is determined as follows

$$k = \arg\min_j \|\tilde{\boldsymbol{h}}_{v_i} - \boldsymbol{\theta}_j\|_2, \tag{11}$$

where $k$ denotes the index of the nearest codeword. Thus, the quantized node state corresponds to the codeword $\boldsymbol{\theta}_k$ as follows

$$\bar{\boldsymbol{h}}_{v_i} = \boldsymbol{\theta}_k. \tag{12}$$

Through the encoding process, each node only needs to transmit the index $k$ of the codeword instead of the original high-dimensional node state.

Upon receiving the index $k$, each node locates the corresponding codeword in the codebook to obtain $\boldsymbol{\theta}_k$ and then recovers the node state using the projection decoder

$$\hat{\boldsymbol{h}}_{v_i} = \text{ProjDecoder}(\boldsymbol{\theta}_k), \tag{13}$$

where $\text{ProjDecoder}(\cdot) : \mathbb{R}^D \to \mathbb{R}^M$ is a MLP.

In prior works on using MPNN for cooperative localization [19], [20] node states are directly exchanged among nodes. In contrast, we construct a VQ codebook to represent all node states, requiring only codeword indices to be transmitted, which significantly reduces the communication overhead.

### D. Communication Cost

The communication cost per round per node is the number of bits required for transmitting message to neighboring nodes and the fixed overhead for preambles or headers of message packets. The fixed overhead is assumed to be $H$ bits per message. In the proposed VQ-MPNN, the communication bits for transmitting the index of the codeword is $\lceil \log_2 K \rceil$. Thus, VQ-MPNN has a total communication cost of $(H + \lceil \log_2 K \rceil) N_i T$ bits per node.

### E. Training Strategy

To train the proposed VQ-MPNN, we create a loss function that consists of the mean squared error (MSE) and VQ loss

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{VQ}} = \underbrace{\sum_{i=1}^{I} \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|_2^2}_{\mathcal{L}_{\text{MSE}}}$$

**Algorithm 1:** Workflow of the proposed VQ-MPNN

**Input:** Initial position $\hat{\boldsymbol{x}}_i^{(0)}$ and distance measurements $\{z_{j\to i}\}_{j\in\mathcal{N}_i}$.
**Output:** Estimated node position $\hat{\boldsymbol{x}}_i$.
**for** each node $i$ **do**
  $\boldsymbol{h}_{v_i}^{(0)} = g_v(\hat{\boldsymbol{x}}_i^{(0)})$, $\boldsymbol{h}_{e_{j\to i}} = g_e(z_{j\to i})$, $\forall j \in \mathcal{N}_i$
  $\text{Index}(\bar{\boldsymbol{h}}_{v_i}^{(0)}) = \text{Encoder}(\boldsymbol{h}_{v_i}^{(0)})$
**end for**
**for** communicaton round $t \in \{1, \ldots, T\}$ **do**
  **for** each node $i$ **do**
    Broadcast $\text{Index}(\bar{\boldsymbol{h}}_{v_i}^{(t-1)})$
    Receive $\{\text{Index}(\bar{\boldsymbol{h}}_{v_j}^{(t-1)})\}_{j\in\mathcal{N}_i}$ from its neighbors
    Recover node states:
    $\hat{\boldsymbol{h}}_{v_j}^{(t-1)} = \text{Decoder}(\text{Index}(\bar{\boldsymbol{h}}_{v_j}^{(t-1)}))$, $\forall j \in \mathcal{N}_i$
    Construct Message:
    $\boldsymbol{m}_{j\to i}^{(t)} = g_m^{(t)}(\hat{\boldsymbol{h}}_{v_j}^{(t-1)}, \boldsymbol{h}_{v_i}^{(t-1)}, \boldsymbol{h}_{e_{j\to i}})$, $\forall j \in \mathcal{N}_i$
    Update node state:
    $\boldsymbol{h}_{v_i}^{(t)} = g_h^{(t)}(\boldsymbol{h}_{v_i}^{(t-1)}, \sum(\{\boldsymbol{m}_{j\to i}^{(t)}\}_{j\in\mathcal{N}_i}))$
    Obtain codeword index:
    $\text{Index}(\bar{\boldsymbol{h}}_{v_i}^{(t)}) = \text{Encoder}(\boldsymbol{h}_{v_i}^{(t)})$
    Broadcast $\text{Index}(\bar{\boldsymbol{h}}_{v_i}^{(t)})$
  **end for**
**end for**
**for** each node $i$ **do**
  $\hat{\boldsymbol{x}}_i = g_v^{\text{est}}(\boldsymbol{h}_{v_i}^{(\text{T})})$
**end for**

$$+\underbrace{\alpha\cdot\Bigg(\sum_{i=1}^{I}\sum_{i=1}^{T}\|\boldsymbol{h}_{v_i}^{(t)}-\text{ProjDecoder}(\tilde{\boldsymbol{h}}_{v_i}^{(t)}+\text{sg}(\bar{\boldsymbol{h}}_{v_j}^{(t)}-\tilde{\boldsymbol{h}}_{v_i}^{(t)}))\|_2^2}_{+\|\text{sg}(\tilde{\boldsymbol{h}}_{v_i}^{(t)})-\bar{\boldsymbol{h}}_{v_j}^{(t)}\|_2^2+\beta\cdot\|\tilde{\boldsymbol{h}}_{v_i}^{(t)}-\text{sg}(\bar{\boldsymbol{h}}_{v_j}^{(t)})\|_2^2\Bigg)}_{\mathcal{L}_{\text{VQ}}}. \tag{14}$$

Here $\alpha$ and $\beta$ denote penalty parameters, and $\text{sg}(\cdot)$ denotes the stopgradient operator [27]. We set $\alpha = 0.1$ to make MSE loss and VQ loss are on the same order of magnitude. We follow [27] to design the VQ loss and set $\beta = 0.25$.

To reduce the training complexity and improve the scalability, all neural networks (i.e., $g_v(\cdot), g_e(\cdot), \{g_m^{(t)}(\cdot), g_h^{(t)}(\cdot)\}_{t=1}^T$, Encoder$(\cdot)$, Decoder$(\cdot)$, and $g_v^{\text{est}}(\cdot)$) share weights among different nodes and edges. After a centralized training stage, the proposed VQ-MPNN can be implemented in a distributed manner during the testing phase. Each node possesses its own neural network. Even when a node is added or removed from the graph, the proposed VQ-MPNN continues to function, which demonstrates its strong generalizability and scalability.

## IV. Simulation Results

### A. Scenario Setting

We consider a two-dimensional static scenario of area spanning $[0, 50]\,m \times [0, 50]\,m$ consisting of 9 anchor nodes and 20 agent nodes. In particular, the agents are uniformly distributed in this $50 \times 50\,(m^2)$ area, and the locations of anchors are set to the four vertices, the midpoints of the sides, and the center of the square area. The communication range $r$ is set to 25 $m$. Fig. 4 illustrates an example of network topology. The prior distribution of each agent is $p(\boldsymbol{x}_i) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$. The covariance $\boldsymbol{\Sigma}$ is given by $\boldsymbol{\Sigma} = \text{diag}\{10, 10\}$. The mean $\boldsymbol{\mu}_i$ is randomly drawn from Gaussian distribution $\mathcal{N}(\boldsymbol{x}_i, \boldsymbol{\Sigma})$. Following [8], [28], we consider two types of measurement noise of distance measurement from node $j$ to $i$: additive white Gaussian noise (AWGN) with zero mean and standard deviation $\sigma$, and range-dependent Gaussian noise with zero mean and a standard deviation of $\sigma\|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2$.

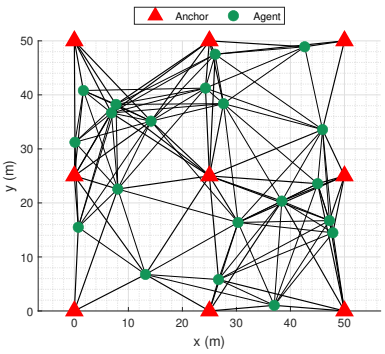



Fig. 4. An example of a cooperative localization network topology. Red triangles denote the positions of 9 anchors and green circles denote the positions of 20 agents. The communication range is set to 25 $m$.

### B. Neural Network Hyperparameters

The neural network hyperparameters we use for our simulations of VQ-MPNN are given in Table I. The dimension of the codeword in the codebook is set to $D = 12$ and the dimension of node state is set to $M = 16$. We generate 3000 training data samples, i.e., network realizations, to train VQ-MPNN. We generate 300 validation data samples to validate the performance of neural network. If the best validation loss does not change for 30 epochs, the training ends. In the test stage, we generate 300 test data samples to test the trained neural networks.

### C. Performance Metric and Baseline Methods

We use root mean square error (RMSE) to evaluate performance, which is defined as

$$\text{RMSE} = \frac{1}{N_{\text{mc}}}\sum_{n=1}^{N_{\text{mc}}}\sqrt{\frac{\sum_{i\in\mathcal{I}_a}\|\hat{\boldsymbol{x}}_i(n)-\boldsymbol{x}_i(n)\|_2^2}{N_a}}, \quad \forall\, n \in \{1, \ldots, N_{\text{mc}}\}, \tag{15}$$

where $N_{\text{mc}}$ denotes the number of Monte Carlo experiments, $\hat{\boldsymbol{x}}_i(n)$ denotes the estimate of node $i$'s position $\hat{\boldsymbol{x}}_i(n)$ in the $n$-th experiment.

The proposed VQ-MPNN is compared with the following baseline methods:

- PLBP [22]: PLBP is a parametric BP algorithm for cooperative localization. PLBP includes double loop iterations: linearization, which calculates linearization parameters, and BP, which calculates beliefs. After each linearization of distance measurements, BP is executed

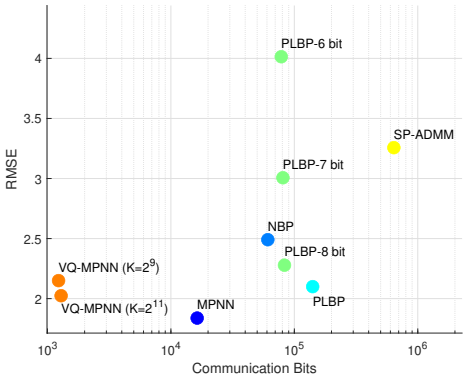
TABLE I
NEURAL NETWORK SETTING

| $g_v(\cdot)$ | | $g_e(\cdot)$ | | $g_m^{(t)}(\cdot)$ | | $g_h^{(t)}(\cdot)$ | | $g_v^{\text{est}}(\cdot)$ | |
|---|---|---|---|---|---|---|---|---|---|
| Layer | Size | Layer | Size | Layer | Size | Layer | Size | Layer | Size |
| Linear + GELU | $2 \times 64$ | Linear + GELU | $1 \times 32$ | Linear + GELU | $3M \times 80$ | Linear + GELU | $2M \times M$ | Linear + GELU | $M \times 128$ |
| Linear + GELU | $64 \times M$ | Linear + GELU | $32 \times 64$ | Linear + GELU | $80 \times 16$ | | | Linear + GELU | $128 \times 256$ |
| | | Linear + GELU | $64 \times 32$ | Linear + GELU | $16 \times M$ | | | Linear + GELU | $256 \times 128$ |
| | | Linear + GELU | $32 \times M$ | | | | | Linear | $128 \times 2$ |

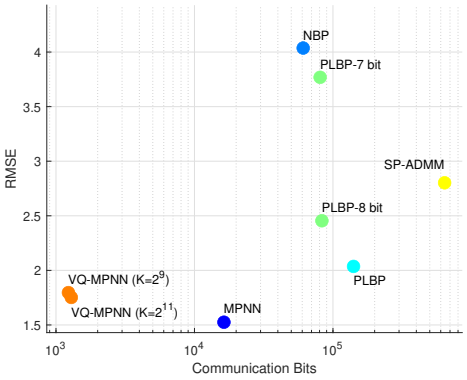
| Projection Encoder | | Codebook | | Projection Decoder | |
|---|---|---|---|---|---|
| Layer | Size | Layer | Size | Layer | Size |
| Linear + GELU | $M \times 16$ | Embedding | $K \times D$ | Linear + GELU | $D \times 16$ |
| Linear + GELU | $16 \times D$ | | | Linear + GELU | $16 \times M$ |

TABLE II
COMPARISON OF COMMUNICATION COST PER NODE

| Method | PLBP | PLBP-$C$ bit | MPNN | VQ-MPNN | SP-ADMM | NBP |
|---|---|---|---|---|---|---|
| Communication Cost (bits) | $JN_i(6Q+H+4QT+HT)$ | $JN_i(6Q+H+4CT+HT)$ | $(MQ+H)N_iT$ | $(H+\lceil\log_2 K\rceil)N_iT$ | $(4Q+H)N_iT$ | $(2N_p+H)N_iT$ |




(a) RMSE versus communication bits with AWGN model.




(b) RMSE versus communication bits with range-dependent noise model.

Fig. 5. Communication cost comparison of different algorithms. The proposed VQ-MPNN achieves a communication bit count several orders of magnitude smaller than that of other algorithms.

on the linearized model. Before calculating linearization parameters, means and variances of posteriors, which are represented by a $2 \times 1$ vector and a $2 \times 2$ matrix, are exchanged between two nodes to calculate the joint posterior of two nodes. Then, linearization parameters are calculated by statistical linear regression with respect to the joint posterior over the two nodes. To calculate belief of node $i$, there are $4N_i$ scalars to be exchanged between node $i$ and its neighbors. Assuming a fixed packet header overhead per message, the total communication cost per node is given by $J \cdot ((6Q+H)N_i + (4Q+H)N_iT)$ bits, where $Q$ represents the quantization bits, $J$ denotes the number of linearization iterations, $T$ denotes the number of BP iterations and $H$ denotes the communication bits of the packet header.

- PLBP-$C$ bit: We modify PLBP by using a scalar quantizer to quantize the messages to be exchanged. The scalar quantizer is as follows. We denote $v$ as the input scalar to the quantizer. We manually set the quantization range $b$. To enforce the scalar within the quantization range $[-b, b]$, we apply a clipping function $\text{clip}(v) = -b + \max(v+b, 0) - \max(v-b, 0)$. The quantization step size is set as $\delta = \frac{2b}{2^C}$ where $C$ denotes the quantization bits. The quantized scalar is given by $\hat{v} = \delta\lceil\frac{v}{\delta}\rceil - \frac{\delta}{2}$. The total communication cost per node is $J \cdot ((6Q+H)N_i + (4C+H)N_iT)$ bits.
- NBP [3], [25]: A nonparametric particle-based BP with beliefs and messages are represented as weighted particles. The total communication cost per node is $(2N_p + H)N_iT$ bits, where $N_p$ denotes the number of particles.
- SP-ADMM [8]: SP-ADMM is an distributed alternating direction method of multipliers (ADMM) algorithm for the nonconvex localization problem. In each iteration, each node sends two two-dimensional variables to its neighbors. The total communication cost per node is $(4Q + H)N_iT$ bits.
- MPNN [20]: A MPNN is trained to directly estimate agents' positions. In each communication round, each node sends one $M$-dimensional vector to its neighbors. The total communication cost for MPNN is $(MQ + H)N_iT$ bits.

### D. Communication Cost Comparison

In this subsection, we compare communication cost of all algorithms under two different noise models. Table II shows the per-node communication costs for all algorithms. In non-quantized algorithms, scalars are typically represented as floating-point numbers. Additionally, the average number of neighbors for each agent node in our scenario is approximately

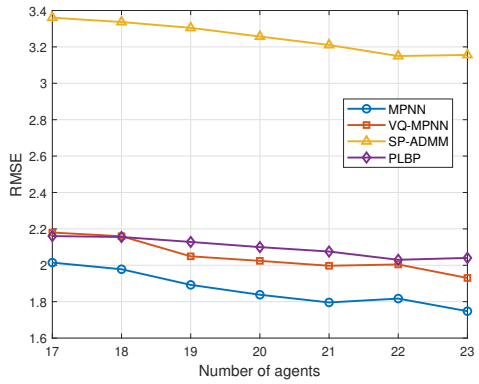



Fig. 6. RMSE versus number of agents under AWGN model. The well-trained VQ-MPNN is able to generalize to different number of agents.

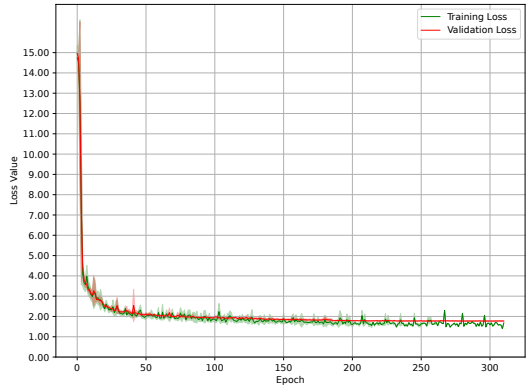
10. Therefore, in the calculations, the quantization bits $Q$ are set to 32 bits, and the number of neighbors $\mathcal{N}_i$ is set to 10. We set $T = 3$ for VQ-MPNN and MPNN, NBP, PLBP and PLBP-$C$ bit. For SP-ADMM, we set $T = 400$ to ensure convergence. Besides, the number of linearization for PLBP and PLBP-$C$ bit is set to 20. The number of particles for NBP is set to $N_p = 1000$. The fixed overhead of packet headers per message $H$ is set to 32 bits.

Based on Table II, Fig. 5(a) illustrates the communication bits versus RMSE under AWGN model with a standard deviation of 4. As shown in Fig. 5(a), the MPNN achieves lowest RMSE and the proposed VQ-MPNN achieves a second lowest RMSE among all the methods. The proposed VQ-MPNN and the existing MPNN outperform BP-based methods in wireless networks with loopy graph topologies, such as the one illustrated in Fig. 4. This is because the data-driven nature of GNNs is able to refine message accuracy and improve overall positioning performance. The localization performance of VQ-MPNN improves with a larger codebook, as it provides greater capacity for message representation. This is because the larger codebook offers greater capacity for message representation. We observe that PLBP-$C$ bit exhibits a higher RMSE compared to the PLBP due to the quantization error and reducing the quantization bits further increases the RMSE. Among all the algorithms, VQ-MPNN clearly offers the best tradeoff in localization error and communication efficiency. It requires the least communication bits (at least one order of magnitude smaller compared to other algorithms) while achieving RMSE almost as good as MPNN.

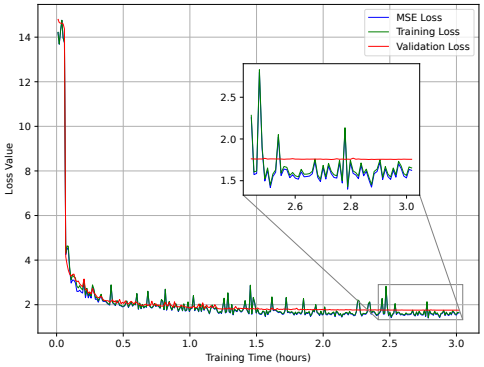
Fig. 5(b) illustrates the communication bits versus RMSE under the range dependent noise model with $n_{j\to i} \sim \text{Gauss}(0, 0.2\|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2)$, where Gauss() denotes the Gaussian distribution. Similar performance trends in terms of RMSE and communication bits can be observed for all algorithms under the range dependent noise model. This shows that our approach is robust to the way in which noise is modeled.

### *E. Assessing Generalizability*

In practice, networks are often dynamic, which requires algorithms that can adapt to varying numbers of agents. This subsection demonstrates that the proposed VQ-MPNN is




(a) Loss value versus epoch. The shaded area depicts the mean $\pm$ standard deviation over five random seeds.




(b) Loss value versus training time.

Fig. 7. Learning curves for the proposed VQ-MPNN.

scalable with respect to varying numbers of agents. We train VQ-MPNN and MPNN using a dataset with a fixed number of agents, and subsequently evaluate the well-trained models across varying agent counts. We train the neural networks with the setup where the number of agents is 20, the number of communication rounds is 3, and the size of the codebook is $2^{10}$. The number of iterations for SP-ADMM is set to 400. For PLBP, The number of linearization iterations and BP iterations is set to 20 and 5, respectively. Fig. 6 illustrates that RMSE under different number of agents. We observe that the RMSE decreases with the increase of the number of agents because more neighbors provide more topology information. The VQ-MPNN trained with with a fixed number of agents still outperform PLBP and SP-ADMM, indicating that the proposed VQ-MPNN generalizes well across different agent counts.

### *F. Convergence Behavior*

Fig. 7 illustrates the training process of the proposed VQ-MPNN with range-dependent noise model of $n_{j\to i} \sim \text{Gauss}(0, 0.2\|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2)$. The size of codebook and the number of communication rounds are set to $2^{11}$ and 3. We train our model on a server equipped with an NVIDIA Tesla V100S PCIe 32GB. Fig. 7(a) shows the loss values over epochs for both training and validation, The results are obtained by

running VQ-MPNN with five different seeds. We observe that the gap between the training and validation losses remains relatively small throughout the training process, indicating that the model neither overfits nor underfits. Additionally, the narrow shaded area suggests that the model is both stable and robust. In Fig. 7(b), we observe that training takes approximately three hours, and the training and validation losses converge to nearly identical values in the final stage. Thus, our method does not require significant computational resources, very large datasets, or time for training. VQ-MPNN provides a quick and efficient data-driven approach to cooperative localization. Additionally, we present the MSE loss in the figure to demonstrate that the gap between the MSE and training losses is minimal, indicating that the VQ loss is very small and our codebook design introduces only a negligible loss due to quantization.

## V. Conclusion

This paper proposed VQ-MPNN to achieve communication-efficient cooperative localization. In our approach, prior node position and distance measurements are treated as node and edge features, which are then encoded as node and edge states. Employing a vector quantized codebook to map the node states to the codewords, each node only need to transmit the indices of the codewords, which significantly enhances communication efficiency. The proposed VQ-MPNN can perform inference on the graph that corresponds to the network topology in a distributed manner. Simulation results demonstrated that the proposed VQ-MPNN simultaneously achieve low localization error and high communication efficiency.

While we have focused on static networks with independent additive noise, an interesting direction of future work would be to consider mobile networks and possibly correlated noise models for distance measurements. We believe that data-driven methods can strongly outperform traditional approaches in mobile networks with correlated noise, since they are hard to model accurately.

## Acknowledgment

This project was supported in part by the National Science Foundation (NSF) under grant CNS-2212565, the Office of Naval Research (ONR) under grants N00014-21-1-2472 and N00014-23-C-1016, and the Air Force Office of Scientific Research (AFOSR) under grant FA9550-24-1-0083.

## References

[1] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, 2005.

[2] Y. Xiong, N. Wu, Y. Shen, and M. Z. Win, "Cooperative localization in massive networks," *IEEE Trans. Inf. Theory*, vol. 68, no. 2, pp. 1237–1258, 2021.

[3] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, 2009.

[4] A. Ihler, J. Fisher, R. Moses, and A. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 809–819, 2005.

[5] F. Meyer, T. Kropfreiter, J. L. Williams, R. Lau, F. Hlawatsch, P. Braca, and M. Z. Win, "Message passing algorithms for scalable multitarget tracking," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 221–259, 2018.

[6] N. Piovesan and T. Erseghe, "Cooperative localization in wsns: A hybrid convex/nonconvex solution," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 162–172, 2016.

[7] E. Gur, S. Sabach, and S. Shtern, "Alternating minimization based first-order method for the wireless sensor network localization problem," *IEEE Trans. Signal Process.*, vol. 68, pp. 6418–6431, 2020.

[8] M. Zhang, Z. Wang, F. Yin, and X. Shen, "Distributed scaled proximal ADMM algorithms for cooperative localization in WSNs," *IEEE Trans. Signal Process.*, 2023.

[9] B. Li, N. Wu, Y.-C. Wu, and Y. Li, "Convergence-guaranteed parametric bayesian distributed cooperative localization," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 8179–8192, 2022.

[10] Y. Weiss and W. Freeman, "Correctness of belief propagation in gaussian graphical models of arbitrary topology," *Proc. Neural Inf. Process. Syst. (NeurIPS)*, vol. 12, 1999.

[11] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2282–2312, 2005.

[12] E. Riegler, G. E. Kirkelund, C. N. Manchón, M.-A. Badiu, and B. H. Fleury, "Merging belief propagation and the mean field approximation: A free energy approach," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 588–602, 2012.

[13] V. G. Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 685–693.

[14] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1801–1819, 2014.

[15] A. T. Ihler, J. W. Fisher III, A. S. Willsky, and D. M. Chickering, "Loopy belief propagation: convergence and effects of message errors." *J. Mach. Learn. Res.*, vol. 6, no. 5, 2005.

[16] V. Savic and S. Zazo, "Reducing communication overhead for cooperative localization using nonparametric belief propagation," *IEEE Wireless Commun. Lett.*, vol. 1, no. 4, pp. 308–311, 2012.

[17] R. Mendrzik, J. Lewandowsky, and G. Bauch, "Particle-based message compression for cooperative localization," in *Proc. IEEE 84th Veh. Technol. Conf. (VTC-Fall)*, 2016, pp. 1–5.

[18] S. Adiga, X. Xiao, R. Tandon, B. Vasić, and T. Bose, "Generalization bounds for neural belief propagation decoders," *IEEE Trans. Inf. Theory*, 2024.

[19] M. Liang and F. Meyer, "Neural enhanced belief propagation for cooperative localization," in *Proc. IEEE Stat. Signal Process. Workshop (SSP)*, 2021, pp. 326–330.

[20] B. C. Tedeschini, M. Brambilla, and M. Nicoli, "Message passing neural network versus message passing algorithm for cooperative positioning," *IEEE Trans. on Cogn. Commun. Netw.*, 2023.

[21] Y. Cao, S. Yang, and Z. Feng, "Distributed cooperative positioning in dense wireless networks: A neural network enhanced fast convergent parametric message passing method," *Proc. IEEE Global Commun. Conf. (Globecom)*, 2024.

[22] Á. F. García-Fernández, L. Svensson, and S. Särkkä, "Cooperative localization using posterior linearization belief propagation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 832–836, 2017.

[23] B. Li, N. Wu, H. Wang, P.-H. Tseng, and J. Kuang, "Gaussian message passing-based cooperative localization on factor graph in wireless networks," *Signal Processing*, vol. 111, pp. 1–12, 2015.

[24] Q. Yu, Y. Wang, and Y. Shen, "Robust distributed cooperative localization in wireless sensor networks with a mismatched measurement model," *IEEE Trans. Signal Process.*, 2024.

[25] J. Lien, U. J. Ferner, W. Srichavengsup, H. Wymeersch, and M. Z. Win, "A comparison of parametric and sample-based message representation in cooperative localization," *Int. J. Navig. Observ.*, vol. 2012, 2012.

[26] H. Kim, S. W. Choi, and S. Kim, "Connectivity information-aided belief propagation for cooperative localization," *IEEE Wireless Commun. Lett.*, vol. 7, no. 6, pp. 1010–1013, 2018.

[27] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Proc. Neural Inf. Process. Syst. (NeurIPS)*, vol. 30, 2017.

[28] D. Jin, F. Yin, A. M. Zoubir, and H. C. So, "Exploiting sparsity of ranging biases for nlos mitigation," *IEEE Trans. Signal Process.*, vol. 69, pp. 3782–3795, 2021.