Cooperative Localization Using Learning-based Constrained Optimization

Changwei Chen and Solmaz S. Kia, *Senior Member, IEEE*

***Abstract*—Loosely coupled multi-agent estimation algorithms such as covariance intersection (CI) for track-to-track fusion, and discorrelated minimum variance (DMV) and practical estimated cross-covariance minimum variance (PECMV) for cooperative localization (CL), which account for inter-agent correlations in an implicit manner, are favored from the less frequent communication aspect. However, they can be computationally too expensive to be online implementable due to the costly optimization process involved. To reduce the computational cost while maintaining the estimation accuracy, in this paper, we report the application of Machine learning (ML) techniques to substitute the optimization processes by learning their optimal solutions to reduce the computational complexity. We focus on the CL problems and propose two data-driven approaches, namely LDMV and LPECMV to generate the solutions of two different constrained optimization procedures contained in implicit CL algorithms. For LDMV, the artificial neural network (NN) technique is used to predict the scalar solution of the problem with a single inequality constraint while in LPECMV the NN works as a matrix predictor to learn the solution of a matrix optimization containing linear matrix inequality (LMI) constraints. We discuss the design of the NNs in detail to respect the constraints. The effectiveness and the generosity of the two methods are demonstrated via CL experiments. And the experimental results show that both our approaches reduce the computational cost significantly without sacrificing the localization accuracy and the estimation consistency.**

***Index Terms*—Keywords: Multi-agent estimation, cooperative localization, machine learning, constrained optimization**

I. Introduction

Distributed multi-agent estimation systems, where each agent is endowed with embedded computing, sensing, and wireless communication abilities, are being deployed in abundance to provide real-time monitoring and control capabilities in many applications. Examples include vehicular ad-hoc networks, environmental monitoring, object tracking, and body area networks. In these multi-agent sensing systems, to reduce communication cost and distribute computation, agents compute and maintain their own local estimates whose accuracies are then improved by fusing the estimates of neighboring agents (in the case of data fusion) or using relative measurement feedback between neighboring agents (in the case of, e.g., cooperative localization (CL)). Such acts, however, create strong correlations among the state estimates of the agents. Neglecting the correlations often leads to excessively optimistic estimates and even the inconsistency of the estimator [1]. In sensor networks, accounting for correlations among the state estimates of the estimation nodes continues to be a challenging task. The approaches to solving this problem can be classified into explicit and implicit methods. Explicitly accounting for the correlations requires a higher computational complexity and demands frequent communication among correlated agents or a server [2]–[5]. In contrast, implicit treatments, either via the use of conservative uncorrelated estimate upper bounds to guarantee estimation consistency or by estimating the unknown correlations, eliminate persistent inter-agent communications but come with higher communication costs.

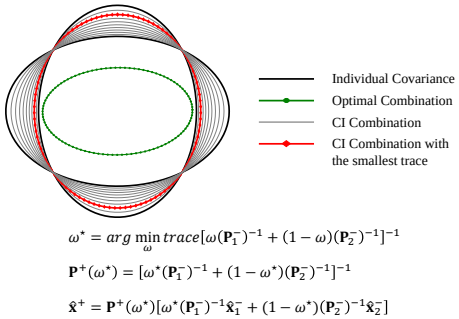


Figure 1 – CI produces the fused estimate $(\hat{\mathbf{x}}^+, \mathbf{P}^+(\omega^\star))$ from two individual estimates $(\hat{\mathbf{x}}_l^-, \mathbf{P}_l^-)$, $l \in \{1,2\}$ whose cross covariance $\mathbf{P}_{12}^-$ is unknown.

The prime example of the implicit fusion algorithms that use conservative uncorrelated estimates is the Covariance Intersection (CI) method [6], see Fig. 1. CI has been used in various sensor network fusion [7]–[9] and also CL problems [10]–[13]. Alternatively, for CL, [14] proposes the discorrelated minimum variance (DMV) relative measurement processing method, which does not directly use the CI but is reminiscent of the CI's approach in using uncorrelated upper bounds on the joint uncertainty matrix of any two estimation nodes. To reduce the conservatism of the CI method, alternative approaches such as Ellipsoidal Intersection (EI) [15] and Inverse Covariance Intersection (ICI) [16] have been proposed in the literature. CI, as well as its alternative approaches, EI and ICI, and also DMV method, have a scalar parameter which is chosen using an optimization process that minimizes the total uncertainty of the fused states so that the best among the

Manuscript received: February 24, 2022; Revised May 8, 2022; Accepted May 26, 2022. This paper was recommended for publication by Editor Sven Behnke upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by NIST award 70NANB17H192.

The authors are with the Department of Mechanical and Aerospace Engineering, University of California, Irvine, CA 92697 changwc3,solmaz@uci.edu

Digital Object Identifier (DOI): see top of this page.

conservative estimates is chosen, see Fig. 1 for the graphical presentation of the CI fusion method. The second class of implicit methods to deal with state estimation in the absence of correlation information trades in extra computation for a better fusion performance by constructing the unknown cross-covariance matrix instead of conservatively over-bounding the joint covariance matrix of the estimates. The results in this class include the Maximum Allocated Covariance (MAC) of [17] and game-theoretic method of [18] for track-to-track fusion and practical estimated cross-covariance minimum variance (PECMV) of [14] for CL. All these three methods estimate the unknown cross-covariance matrix of any two nodes using a matrix optimization framework. They deliver a less conservative estimate than CI and its variants but computationally are expensive, to the point that they may not be solved online, especially on an embedded computing system.

To reduce the computational cost of the consistent implicit track-to-track fusion and CL methods, we propose using machine learning (ML) based solvers to solve the optimization problems used in these methods. We demonstrate our approaches for solving the optimization problems in the DMV and the PECMV methods for CL proposed in [14]. We evaluate the effectiveness of our proposed approaches using a set of experiments for pedestrian localization using an inertial navigation system (INS) aided by CL. In these experiments, we observe that the run time of the proposed approaches compared with DMV and the PECMV is reduced 53.85% and 99.98% with no sacrifices on the localization accuracy and estimation consistency of CL filters. Notice that our proposed ML base optimizer is an approach that can be easily extended to the loosely-coupled estimation algorithms reviewed above as well as covariance union (CU) and arithmetic average (AA) density fusion [19].

ML-based solutions for optimization problems have been successfully used in the applications such as combinatorial optimization [20], wireless network optimization [21], and supply chain management [22]. ML-based approaches have also been used in CL problems but for different purposes. For example, [23] uses a deep neural network (DNN) to assist a CL for vehicular networks, where DNN is designed to solve the chronic nonlinear approximation problem. Or, [24] uses an ML-based surrogate model as a measurement scheduling merit function.

II. Problem Definition

Consider a group of N mobile agents with communication and computation capabilities. The equation of motion of each agent $i \in \mathcal{V} = \{1, \cdots, N\}$ at time step $t \in \mathbb{Z}^+$ is described by

$$\mathbf{x}^i(t) = \mathbf{f}(\mathbf{x}^i(t-1), \mathbf{u}^i_m), \quad \mathbf{x}^i \in \mathbb{R}^{n^i} \tag{1}$$

where $\mathbf{x}^i(t) \in \mathbb{R}^{n^i}$ is the state of agent i (e.g., position, velocity, attitude) and $\mathbf{u}^i_m = \mathbf{u}^i(t) + \boldsymbol{\nu}^i_u$ is the self-motion measurement command obtained, e.g., from odometry or inertial measurement unit (IMU). Here, $\boldsymbol{\nu}^i_u$ is the self-motion measurement noise. Each agent uses a local filter to obtain an estimate of its own state $\hat{\mathbf{x}}^{i-}(t) \in \mathbb{R}^{n^i}$ and its corresponding error covariance matrix $\mathbf{P}^{i-}(t) \in \mathbb{S}^{++}_{n^i}$ at each timestep $t \in \mathbb{Z}^+$ using its motion model and occasional access to absolute measurements through e.g. GPS or measurement from known landmarks. Here, $\mathbb{S}^{++}_n$ is the set of positive definite matrices of size n. We call $\text{bel}^{i-}(t) = (\hat{\mathbf{x}}^{i-}(t), \mathbf{P}^{i-}(t))$ the prior belief of agent i at time t.

Due to the contaminating noise in the self-motion measurements, the localization accuracy of the agents degrades during the mission horizon. If access to absolute measurements to correct dead-reckoning based localization is limited, to bound the error and improve accuracy, CL via joint processing of *occasional* relative measurements among two agents is used. Let the relative measurement (e.g., relative range, relative bearing, relative pose, or a combination of them) $\mathbf{z}^i_j(t)$ obtained by agent i from agent j at time t be

$$\mathbf{z}^i_j(t) = \mathbf{h}(\mathbf{x}^i(t), \mathbf{x}^j(t)) + \boldsymbol{\nu}^i(t), \quad \mathbf{z}^i_j \in \mathbb{R}^{n^i_z} \tag{2}$$

where $\boldsymbol{\nu}^i(t) \in \mathcal{N}(\mathbf{0}, \mathbf{R}^i(t))$ is the zero mean Gaussian measurement noise with covaraince matrix $\mathbf{R}^i(t) \in \mathbb{S}^{++}_{n^i_z}$.

When no inter-agent measurement is available to update the local belief, the updated belief is set to be the propagated belief, i.e., $\text{bel}^{i+}(t) = \text{bel}^{i-}(t) = (\hat{\mathbf{x}}^{i-}(t), \mathbf{P}^{i-}(t))$. Otherwise, the local belief is corrected using the fusion approaches as briefly outlined below. For more information, see [14]. To simplify the notation, hereafter we only include the time index t when the clarification is needed.

A. Relative measurement processing via DMV method in the absence of explicit knowledge of the inter-agent cross-covariances

Let the joint belief of the agents i and j be $\text{bel}^-_J(t) = (\hat{\mathbf{x}}^-_J(t), \mathbf{P}^-_J(t))$, where

$$\hat{\mathbf{x}}^-_J(t) = \begin{bmatrix} \hat{\mathbf{x}}^{i-}(t) \\ \hat{\mathbf{x}}^{j-}(t) \end{bmatrix}, \quad \mathbf{P}^-_J(t) = \begin{bmatrix} \mathbf{P}^{i-}(t) & \mathbf{P}^-_{ij}(t) \\ \mathbf{P}^-_{ij}(t)^\top & \mathbf{P}^{i-}(t) \end{bmatrix}. \tag{3}$$

When agent i takes a relative measurement from agent j, agent i can correct its local belief using the measurement feedback $\mathbf{z}^i_j(t) - \hat{\mathbf{z}}^i_j(t)$, where $\hat{\mathbf{z}}^i_j(t)$ is the estimated relative measurement based on agent i and j's prior beliefs. When the cross-covariance term $\mathbf{P}^-_{ij}(t)$ is known, the feedback gain can be computed from an Extended Kalman like update procedure. DMV algorithm aims to process the relative measurement $\mathbf{z}^i_j(t)$ to correct the local beliefs in the absence of explicit knowledge of the cross-covariance $\mathbf{P}^-_{ij}(t)$, see [14]. In the DMV method, similar to the joint estimate interpretation of the data fusion problem in [25], a discorrelated upper bound is used to account for any unknown cross-covariance term $\mathbf{P}^-_{ij}(t)$ as

$$\begin{bmatrix} \mathbf{P}^{i-}(t) & \mathbf{P}^-_{ij}(t) \\ \mathbf{P}^-_{ij}(t)^\top & \mathbf{P}^{j-}(t) \end{bmatrix} \leq \begin{bmatrix} \frac{1}{\omega}\mathbf{P}^{i-}(t) & \mathbf{0} \\ \mathbf{0} & \frac{1}{1-\omega}\mathbf{P}^{j-}(t) \end{bmatrix}, \ \omega \in [0,1]. \tag{4}$$

Then, the DMV algorithm updates the propagated belief of agent i at time t, $\text{bel}^{i-}(t)$, according to

$$\hat{\mathbf{x}}^{i+}(t) = \hat{\mathbf{x}}^{i-}(t) + \bar{\mathbf{K}}^i(\omega^i_\star)\,(\mathbf{z}^i_j(t) - \hat{\mathbf{z}}^i_j(t)), \tag{5a}$$

$$\mathbf{P}^{i+}(t) = \bar{\mathbf{P}}^{i}(\omega_\star^i), \tag{5b}$$

where $\bar{\mathbf{K}}^{i}(\omega) = \frac{\mathbf{P}^{i-}}{\omega}\mathbf{H}_i^{i\top}\left(\mathbf{H}_i^i\frac{\mathbf{P}^{i-}}{\omega}\mathbf{H}_i^{i\top} + \mathbf{H}_j^i\frac{\mathbf{P}^{j-}}{1-\omega}\mathbf{H}_j^{i\top} + \mathbf{R}^i\right)^{-1}$, and $\bar{\mathbf{P}}^{i}(\omega) = \left(\omega(\mathbf{P}^{i-})^{-1} + (1-\omega)\mathbf{H}_i^{i\top}(\mathbf{H}_j^i\mathbf{P}^{j-}\mathbf{H}_j^{i\top} + (1-\omega)\mathbf{R}^i)^{-1}\mathbf{H}_i^i\right)^{-1}$, with $\mathbf{H}_i^i = \partial\mathbf{h}(\hat{\mathbf{x}}^{i-}, \hat{\mathbf{x}}^{j-})/\partial\mathbf{x}^i$ and $\mathbf{H}_j^i = \partial\mathbf{h}(\hat{\mathbf{x}}^{i-}, \hat{\mathbf{x}}^{j-})/\partial\mathbf{x}^j$. The optimal ω, denoted by $\omega_\star^i$, is obtained from the optimization problem (6)

$$\omega_\star^i = \underset{0\le\omega\le 1}{\operatorname{argmin}} \ \log\det\bar{\mathbf{P}}^{i}(\omega). \tag{6}$$

According to [14, Theorem 3.1], despite the unknown $\mathbf{P}_{ij}^{-}(t)$, the DMV update is guaranteed to be no worse than the local belief of the agents.

B. Relative measurement processing via PECMV in the absence of explicit knowledge about the inter-agent cross-covariance

Since the upper bound of DMV on the joint covariance matrix accounts for all the possible values for the unknown cross-covariance $\mathbf{P}_{ij}^{-}(t)$, DMV produces conservative updates. To reduce the conservatism of DMV, the PECMV [14] proposes an alternative way in which the unknown cross-covariance $\mathbf{X}$ in the joint covariance matrix

$$\mathbf{P}_{\mathcal{J}}^{-}(\mathbf{X}) = \begin{bmatrix}\mathbf{P}^{i-} & \mathbf{X}\\ \mathbf{X}^\top & \mathbf{P}^{j-}\end{bmatrix} \tag{7}$$

is estimated from the following optimization problem

$$\mathbf{X}^\star = \arg\max_{\mathbf{X}} \det\begin{bmatrix}\mathbf{I}_{n^i}\\ \mathbf{0}\end{bmatrix}^\top (\mathbf{P}_{\mathcal{J}}^{-}(\mathbf{X})^{-1} + \mathbf{H}^{i\top}\mathbf{R}^{i-1}\mathbf{H}^i)^{-1}\begin{bmatrix}\mathbf{I}_{n^i}\\ \mathbf{0}\end{bmatrix}, \tag{8a}$$

$$\text{subject to } \begin{bmatrix}\mathbf{P}^{i-} & \mathbf{X}\\ \mathbf{X}^\top & \mathbf{P}^{j-}\end{bmatrix} > 0. \tag{8b}$$

where $\mathbf{I}_{n^i}$ is the identity matrix. PECMV estimates $\mathbf{P}_{ij}^{-}$ by obtaining $\mathbf{X}^\star$ that provides the most conservative updated covariance. It is shown in [14] that this optimization problem can be cast in an equivalent convex matrix optimization with linear inequality (LMI) constraints and solved by CVXOPT [26], a Python software for convex optimization.

Once $\mathbf{X}^\star$ is obtained from (8), the PECMV updated belief $\text{bel}_{\text{PECMV}}^{i+}(t) = (\hat{\mathbf{x}}_{\text{PECMV}}^{i+}(t), \mathbf{P}_{\text{PECMV}}^{i+}(t))$ for agent i is

$$\hat{\mathbf{x}}_{\text{PECMV}}^{i+} = \hat{\mathbf{x}}^{i-} + \mathbf{K}_{\text{PECMV}}^{i}\,(\mathbf{z}_j^i - \hat{\mathbf{z}}_j^i), \tag{9a}$$

$$\mathbf{P}_{\text{PECMV}}^{i+} = \begin{bmatrix}\mathbf{I}_{n^i}\\ \mathbf{0}\end{bmatrix}^\top (\mathbf{P}_{\mathcal{J}}^{-}(\mathbf{X}^\star)^{-1} + \mathbf{H}^{i\top}\mathbf{R}^{i-1}\mathbf{H}^i)^{-1}\begin{bmatrix}\mathbf{I}_{n^i}\\ \mathbf{0}\end{bmatrix} \tag{9b}$$

where $\mathbf{K}_{\text{PECMV}}^{i} = \begin{bmatrix}\mathbf{I}_{n^i} & \mathbf{0}\end{bmatrix}\mathbf{P}_{\mathcal{J}}^{-}(\mathbf{X}^\star)\mathbf{H}_i^{i\top}(\mathbf{H}_i^i\mathbf{P}_{\mathcal{J}}^{-}(\mathbf{X}^\star)\mathbf{H}_i^{i\top} + \mathbf{R}^i)^{-1}$. The PECMV update satisfies $\mathbf{P}_{\text{PECMV}}^{i+}(t) \le \mathbf{P}^{i-}(t)$ and $\mathbf{P}_{\text{PECMV}}^{i+}(t) \le \mathbf{P}_{\text{DMV}}^{i+}(t)$.

C. Objective statement

The optimization problems (6) and (8) used respectively in DMV and PECMV algorithms can be viewed as the the following functions

$$\omega_\star^i = f_{\text{DMV}}(\mathbf{P}^{i-}, \mathbf{P}^{j-}, \mathbf{H}_i^i, \mathbf{H}_j^i), \tag{10a}$$

$$\mathbf{X}^\star = f_{\text{PECMV}}(\mathbf{P}^{i-}, \mathbf{P}^{j-}, \mathbf{H}_i^i, \mathbf{H}_j^i). \tag{10b}$$

Evaluating these functions to obtain the desired outputs is computationally expensive and time-consuming. Our objective is to use an ML approach to learn these functions and circumvent solving constrained optimization problems required to evaluate the function values.

III. LDMV AND LPECMV

The universal approximation theorem claims that a neural network (NN) with enough depth can approximate any continuous function given certain weights [27]. If one perceives f_{DMV} and f_{PECDMV} as a continuous function, then they can be approximated by NNs similar to the one in Fig.3. But learning these functions cannot be carried out in a trivial manner. $\omega_\star^i$ and $\mathbf{X}^\star$, the output of f_{DMV} and f_{PECDMV}, are each constrained values, i.e., $\omega_\star^i \in [0,1]$, and $\mathbf{X}^\star$ should be computed such that the joint covariance matrix (7) is positive definite. As we explain below, we ensure $\omega_\star^i \in [0,1]$ by appropriate choice of a proper activation functions for the output layer of NN representation of f_{DMV}. To ensure positive definiteness of joint covaraince matrix, however, we need to take further actions when designing an NN model of f_{PECDMV}. The main tool aiding us is the following result. Here, recall that a matrix $\mathbf{M} \in \mathbb{R}^{p\times q}$ is a strict contraction matrix if and only if $\|\mathbf{M}\|_2 < 1$, where $\|.\|_2$ is the 2-norm [28].

Lemma 3.1 (c.f. [28, page 207 and page 350]): Let $\mathbf{A}\in\mathbb{R}^{p\times p}$ and $\mathbf{B}\in\mathbb{R}^{q\times q}$, and $\mathbf{X}\in\mathbb{R}^{p\times q}$ be given. Then, the joint matrix $\left[\begin{smallmatrix}\mathbf{A} & \mathbf{X}\\ \mathbf{X}^\top & \mathbf{B}\end{smallmatrix}\right]$ is positive definite if and only if $\mathbf{A}$ and $\mathbf{B}$ are positive definite and there is a strict contraction matrix $\mathbf{M}$ such that $\mathbf{X} = \sqrt{\mathbf{A}}^\top\,\mathbf{M}\,\sqrt{\mathbf{B}}$.

Invoking Lemma 3.1, to ensure positive definiteness of the joint covariance matrix (7), we can write

$$\mathbf{X} = \sqrt{\mathbf{P}^i}^\top\,\mathbf{C}\,\sqrt{\mathbf{P}^j} \tag{11}$$

and require that $\|\mathbf{C}\|_2 < 1$. Thus, to obtain $\mathbf{X}^\star$ using an NN we will learn

$$\mathbf{C}^\star = \left(\sqrt{\mathbf{P}^i}^\top\right)^{-1}\mathbf{X}^\star\left(\sqrt{\mathbf{P}^j}\right)^{-1}, \mathbf{C}^\star \in \mathbb{R}^{n^i\times n^j} \tag{12}$$

i.e., the function we want to learn is

$$\mathbf{C}^\star = f_{\text{PECMV}}(\mathbf{P}^{i-}, \mathbf{P}^{j-}, \mathbf{H}_i^i, \mathbf{H}_j^i).$$

For the learning process, all the matrices are flatten into vectors, meaning that the matrices are learned in an element-wise manner. We carry out a supervised learning in which we collect the labeled data by solving the optimization problem (6) to

obtain $\omega^i_\star$ and optimization problem (8) followed by (12) for $\mathbf{C}^\star$. The input and the corresponding label values are

$$input = \mathsf{vectorization}(\mathbf{P}^{i-}, \mathbf{P}^{j-}, \mathbf{H}^i_i, \mathbf{H}^i_j), \tag{13a}$$

$$label_{\text{DMV}} = \omega^i_\star, \tag{13b}$$

$$label_{\text{PECMV}} = \mathsf{vectorization}(\mathbf{C}^\star). \tag{13c}$$

The feed-forward result of the $\mathbf{C}^\star$ prediction network is reconstructed to have the same shape as the $\mathbf{C}^\star$ matrix and used to compute the predicted $\mathbf{X}^\star$.

For the ω prediction network, we set the activation function of the output layer to the standard logistic function (Sigmoid) $S(\chi) = \frac{1}{1+\mathrm{e}^{-\chi}}$ which maps $\mathbb{R} \to [0, 1]$. This choice naturally constraints the learned $\omega^i_\star$, denoted as $\hat{\omega}^i_\star$, to $[0, 1]$. Enforcing the learned $\mathbf{C}^\star$, denoted as $\hat{\mathbf{C}}^\star$, to be strict contraction matrix, i.e., $\|\hat{\mathbf{C}}^\star\|_2 < 1$ is not straightforward and requires a careful selection of the activation functions for the output layer. To enforce $\|\hat{\mathbf{C}}^\star\|_2 < 1$, we implement three methods. In the first method, we add a barrier function [29] into the loss function for training the NN as shown below,

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{M} \sum_{m=1}^{M} \Big(\|\mathbf{Y}_m - \hat{\mathbf{Y}}_m\|_2 + \lambda \log(1 - \|\hat{\mathbf{Y}}_m\|_2) \Big), \tag{14}$$

where M is the number of the total data points, $\mathbf{Y}_m \in \mathbb{R}^{n^i n^j \times 1}$ is the m-th vector of label values, $\hat{\mathbf{Y}}_m \in \mathbb{R}^{(n^i \times n^j) \times 1}$ is the m-th feed-forward prediction of the network and $\mathbf{B} = \lambda \log(1 - \|\hat{\mathbf{Y}}_m\|_2)$ is the barrier function with the barrier parameter λ selected properly [29]. It is observed from (14), when $\hat{\mathbf{Y}}_m$ approaches 1, the value of the barrier function approaches infinity, which prevents the solution from violating the inequality constraint $\|\hat{\mathbf{Y}}_m\|_2 < 1$ for all $m = 1, 2, \cdots, M$. For the output layer activation function, we use hyperbolic tangent function (Tanh), $T(\chi) = \frac{\mathrm{e}^{\chi} - e^{-\chi}}{e^{\chi} + e^{-\chi}}$ to constraint every output to $[-1, 1]$. Given the norm relations [28]

$$\|\hat{\mathbf{C}}^\star\|_2 \le \|\hat{\mathbf{C}}^\star\|_{\text{F}} = \|\hat{\mathbf{Y}}_m\|_2 < 1, \tag{15}$$

where $\|.\|_{\text{F}}$ denotes the Frobenius norm, the barrier $\mathbf{B} = \lambda \log(1 - \|\hat{\mathbf{Y}}_m\|_2)$ constraints the output of the NN to respect the constraint $\|\hat{\mathbf{C}}^\star\|_2 < 1$ for the training data but there is no guarantee for non-training data. The potential distributional mismatch between the training and non-training data can lead to inconsistent performance or even unsafe execution [30]. In the second method, given (15) and $\|\hat{\mathbf{C}}^\star\|_{\text{F}} = \sqrt{\sum_{i=1}^{n^i} \sum_{j=1}^{n^j} |\hat{c}^\star_{ij}|^2}$, we choose the activation function of the output layer to be Tanh normalized by $D = \sqrt{n^i \times n^j}$, i.e., $T(\chi) = \frac{1}{D} \frac{\mathrm{e}^{\chi} - e^{-\chi}}{e^{\chi} + e^{-\chi}}$. Since the constraint $\|\hat{\mathbf{C}}^\star\|_2 < 1$ is embedded in the NN, we drop the barrier function from the loss function (14). Our second method enforces the constraint $\|\hat{\mathbf{C}}^\star\|_2 < 1$ in a hard way, but comes with some degree of conservatism, as it limits the ranges of the elements of matrix $\hat{\mathbf{C}}^\star$. For cases that $n^i = n^j = n$, we propose an alternative method that enforces the constraint $\|\hat{\mathbf{C}}^\star\|_2 < 1$ and it also comes with lower computational complexity. For this third method, we approximate $\mathbf{C}^\star$ with a diagonal matrix, i.e., $\mathbf{C}^\star \approx \mathbf{C}^\star_{\text{Diag}} = \text{Diag}(\boldsymbol{\rho}^\star)$, where $\boldsymbol{\rho}^\star = [\rho^\star_1, \rho^\star_2, \cdots, \rho^\star_n]$ and to enforce $\|\hat{\mathbf{C}}^\star\|_2 < 1$ we set the output layer activation functions to Tanh function. We set our objective to find a $\mathbf{C}^\star_{\text{Diag}}$ that results in minimum value for $\|\mathbf{C}^\star - \hat{\mathbf{C}}^\star\|_F$. Given the definition of the Frobenius norm, the off-diagonal elements of $\mathbf{C}^\star$ does not play a role in choosing $\mathbf{C}^\star_{\text{Diag}}$. Therefore, our loss function is $\mathcal{L}(\mathbf{Y}, \bar{\mathbf{Y}}) = \frac{1}{M} \sum_{m=1}^{M} \|\mathbf{Y}_m - \bar{\mathbf{Y}}_m\|_2$ where $\bar{\mathbf{Y}}_m$ is the vectorized form of diagonal elements of training $\mathbf{C}^\star$s.

Methods	Output layer activation function	Loss function
1	$T(\chi) = \frac{\mathrm{e}^{\chi} - e^{-\chi}}{e^{\chi} + e^{-\chi}}$	$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{M} \sum_{m=1}^{M} \Big(\|\mathbf{Y}_m - \hat{\mathbf{Y}}_m\|_2 + \lambda \log(1 - \|\hat{\mathbf{Y}}_m\|_2) \Big)$
2	$T(\chi) = \frac{1}{D} \frac{\mathrm{e}^{\chi} - e^{-\chi}}{e^{\chi} + e^{-\chi}}$	$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{M} \sum_{m=1}^{M} \Big(\|\mathbf{Y}_m - \hat{\mathbf{Y}}_m\|_2 \Big)$
3	$T(\chi) = \frac{\mathrm{e}^{\chi} - e^{-\chi}}{e^{\chi} + e^{-\chi}}$	$\mathcal{L}(\mathbf{Y}, \bar{\mathbf{Y}}) = \frac{1}{M} \sum_{m=1}^{M} \|\mathbf{Y}_m - \bar{\mathbf{Y}}_m\|_2$

Table I – Summary of the three NNs using three methods, where $D = \sqrt{n^i \times n^j}$.

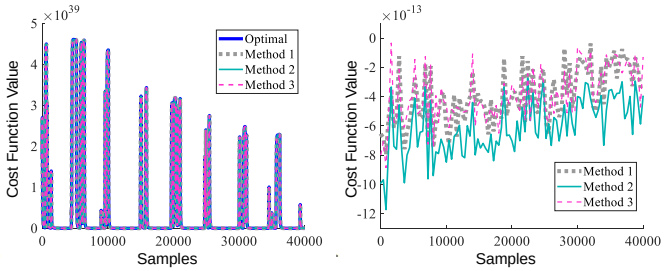


Figure 2 – The objective function value associated with the optimal $\mathbf{C}^\star$ and $\hat{\mathbf{C}}^\star$ obtained from different learning methods (left) and the difference between each learning method and the optimal one, respectively (right).

We train three NNs summarized in Table I, each for one of the methods we discussed to enforce $\|\hat{\mathbf{C}}^\star\|_2 < 1$ and test each well-trained NN on a test data set. We evaluate the performance of each proposed method by comparing the value of the objective function (8a) with (11) computed using the prediction of each NN, i.e., $\hat{\mathbf{C}}^\star$ to the optimal objective function value attained by $\mathbf{C}^\star$, and we subtract the optimal value from the value of each learning method to show the difference. The result is shown in Fig. 2.

As depicted in Fig.2, all proposed learning methods perform less optimally comparing to the original optimization (8) as expected, but the differences are insignificant. Also, the information loss when approximating matrix $\mathbf{C}^\star$ with a diagonal matrix in the third method is negligible while normalizing the output value in the second method degrades the optimality further since the hard constraint imposed by the normalized activation function restricts conservatively the feasible set for the optimization of NN weights, which may exclude the optimal NN weights in the corresponding unconstrained optimization, resulting in a prediction that performs less optimally than others. Given the possibility of $\hat{\mathbf{C}}^\star \ge 1$ in the first method, we therefore adopt the third method, i.e., learning a diagonal approximation $\mathbf{C}^\star_{\text{Diag}}$ of matrix $\mathbf{C}^\star$ to build the NN to learn the matrix constrained optimization. The label value of the $\mathbf{C}^\star_{\text{Diag}}$ prediction network is $\boldsymbol{\rho}^\star$, the diagonal of $\mathbf{C}^\star_{\text{Diag}}$, instead

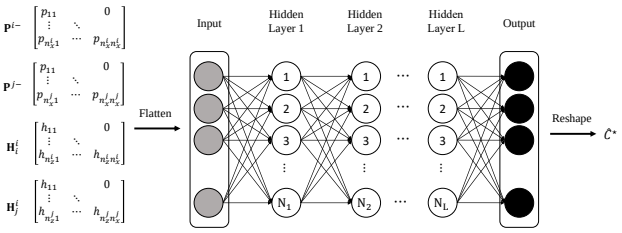


Figure 3 – The NN structure for learning $\mathbf{C}^\star$ of PECMV. The structure for learning $\omega^\star \in [0, 1]$ of DMV looks similar except that at the output layer we have a single neuron.

of (13c). The prediction $\hat{\mathbf{C}}^\star_{\text{Diag}}$ is used to calculate the predicted $\mathbf{X}^\star$. Figure 3 shows the NN structure that we used to learn the outputs of f_{DMV} and f_{PECMV}. The NN regression we use for learning-based DMV (LDMV) and learning-based PECMV (LPECMV) is fast and suitable for real-time application since the training process is conducted off-line despite the large amount of the training data collected with a high sampling rate. The training and the hyperparameter fine-tuning process for the NNs for the experimental demonstration is presented in Section IV.

IV. Experimental Result

We generated the training data for LDMV and LPECMV via a set of CL-aided pedestrian inertial navigation experiments implementing DMV and PECMV, which were conducted in the Calit2 building at the University of California, Irvine (UCI) and the Firstnet building of the National Institute of Standards and Technology (NIST) in Colorado. Two agents with shoe-mounted IMUs and UWB sensors, shown in Fig. 4, walked along different trajectories, e.g., rectangles, circles, lemniscates, and even more complex trajectories. Two example trajectories are shown in Fig. 5. Two additional UWB sensors are placed at known locations acting as beacons. Only agent 1 has access to the beacons, and agent 2 can merely communicate with agent 1. Based on the UWB range measurements, CL can be implemented to correct the zero velocity update (ZUPT) [31] aided pedestrian inertial navigation solution of the agent 2. To generate independent and identically distributed samples for training, validating, and testing, we performed single-step propagation and update from initial belief with randomly generated errors for each pair of self-motion measurement and relative measurement. We solved the optimization problems in both DMV and PECMV using the data from each experiment and obtained an abundant and diverse solution set. The training data is generated by randomly sampling from the entire solution set.

The execution time of DMV is much shorter than PECMV in the sample experiments, as expected. From all experiments, we sampled 50000 samples which are split into a training set (40000 samples), validating set (5000 samples), and test set (5000 samples). The number of the states of each agent i is $n^i = 9$, and the size of the measurement vector is $n^i_z = 1$. Each sample training data after vectorization is flattened to

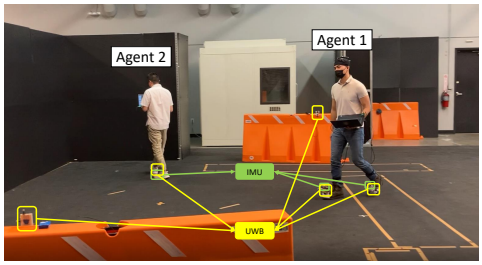


Figure 4 – Training data generation experiments in Firstnet building.

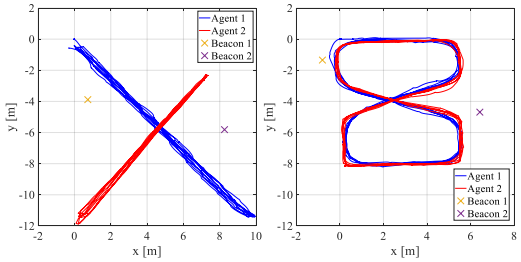


Figure 5 – Examples of the designed trajectories.

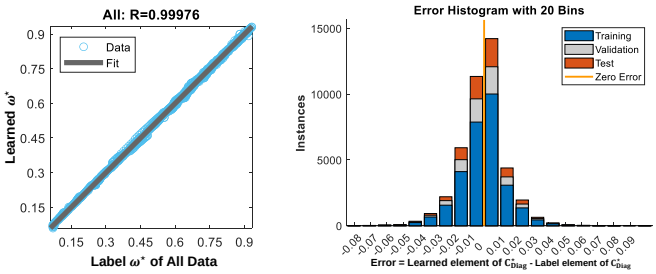
Hyperparameters	$\omega^\star$ network	$\mathbf{C}^\star_{\text{Diag}}$ network
Number of hidden layers L	4	3
Number of units N_L per layer	9	20
Loss	MSE	MSE
Activation function	Sigmoid	Tanh
Optimizer	SGD	Adam
Batch size	256	512
Learning rate α	0.05	0.025
Epochs	200	200

Table II – The hyperparameter design of the NNs.

a vector with $2 \times \big(9 \times (9+1)/2\big) + 2 \times 9 = 108$ features due to the symmetry of covariance matrices. The features are normalized to achieve easier optimization and faster learning. Based on the NN structures in Fig. 3, we put 108 nodes in the input layer and only 1 node in the output layer of $\omega^\star$ prediction network while the output layer node number of $\mathbf{C}^\star_{\text{Diag}}$ prediction network is 9, i.e., $n^i = n^j = n = 9$. The number of hidden layers and the number of nodes in each hidden layer are fine-tuned using a grid search method during the training process. The activation function of the $\omega^\star$ network is Sigmoid for each layer while the one of $\mathbf{C}^\star_{\text{Diag}}$ network is Tanh. We use Mean-squared-error (MSE) as the loss function due to its convexity and also facilitate the learning using the stochastic gradient descent (SGD) and Adam [32]. The fine-tuned design of the two NNs is shown in Table II, and the training results are shown in Fig. 6.

A. Numerical Evaluation

We evaluated the efficiency of our proposed LDMV and LPECMV algorithms by implementing them in another set of CL-aided pedestrian inertial navigation experiments conducted in the KCS lab in the Engineering Gateway building at UCI. In these experiments, two agents equipped with shoe-mounted IMUs and UWB sensors walked along two rectangular trajectories maintaining the communication as depicted in Fig. 7.



(a) $\omega^\star$ NN Fitting (b) $\mathbf{C}^\star_{\text{Diag}}$ NN Error

Figure 6 – The training results of the $\omega^\star$ prediction NN (a) and the $\mathbf{C}^\star_{\text{Diag}}$ prediction NN (b), respectively. (a) is the fitting performance of $\omega^\star$ prediction NN for all data sets with R to be the square root of the coefficient of determination which indicates the fitness of the predicted output and the target value. The closer to 1, the better the fitness. Also, the perfect fit line is shown as "Fit" in gray. (b) is the training error histogram of the $\mathbf{C}^\star_{\text{Diag}}$ prediction NN which represents the difference between the predicted outputs and the label values of the NN for the training, validation and test set respectively.

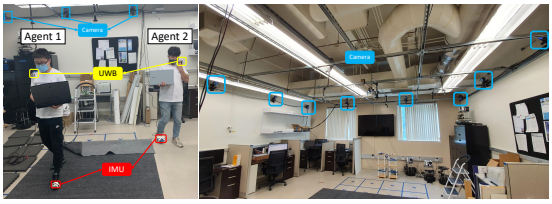
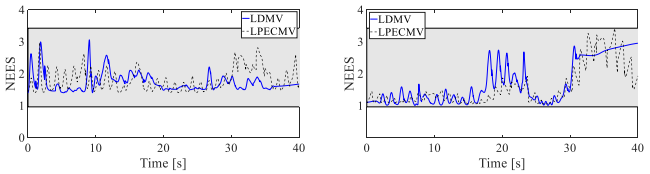


Figure 7 – The test experiments for LDMV and LPECMV conducted in the KCS lab with the aid of the OptiTrack motion capture system.

Agent 1 kept receiving information from both a UWB anchor at a known location and agent 2. The sampling rates of the IMU for agent 1 and agent 2 were, respectively, 200 Hz and 40 Hz, making agent 1 the more accurate agent and thus creating a non-homogeneous scenario where agent 2 improved its accuracy via CL with agent 1. Both agents implemented a local ZUPT aided INS. We used an OptiTrack real-time motion capture system, as shown in Fig. 7, to provide high accuracy reference trajectories for evaluating the performance of our CL algorithms. We attached the reflective markers of the OptiTrack system to the IMUs so that their locations are obtained as references. With the aid of the reference trajectory, the Root Mean Square Error (RMSE) and the Normalized Estimation Error Squared (NEES) are calculated and used as the performance metric to evaluate different localization algorithms. The experiments were repeated 10 times. The experiment results for the position RMSE and the NEES plots are shown in Table III and Fig. 8. The two-sided 95% region for the NEES is $[0.96, 3.42]$ for this set of experiments, see [33]. Also, we compared the average execution time of the CL algorithms. The results are reported in Table IV. The estimated trajectories of the agents are shown in Fig. 9.

The NEES plots in Fig. 8 show that LDMV and LPECMV CL algorithms maintain the consistency of the estimates the same way as DMV and PECMV as reported in [14]. As



(a) Agent 1 (b) Agent 2

Figure 8 – The NEES plots for the two agents over 50 Monte Carlo runs. The shaded area in the NEES plots show the consistency zone.

	ZUPT-only	DMV	LDMV	PECMV	LPECMV
Agent 1	0.4732	0.1050	0.1158	0.1035	0.1125
Agent 2	0.4902	0.3221	0.3528	0.2950	0.3027

Table III – The average RMSE (m) of the estimated trajectories.

	DMV	LDMV	PECMV	LPECMV
Full simulation time (s)	86.9828	40.1316	189282.6081	43.9001
Single execution time (ms)	3.2197	0.2122	7261.5138	0.2002
Computing platform	Dell Laptop: Intel Core™ i5-1135G7@4.20GHz, quad-core, 8 GB memory			

Table IV – The average run time of each algorithm with the computing platform.

it can be concluded from Fig. 9 and Table III, all the CL algorithms significantly improve the ZUPT/INS-only solution. As expected, PECMV and LPECMV outperform DMV and LDMV respectively in terms of the RMSE due to their less conservatism as reported in [14], and the PECMV and DMV have better localization accuracy compared to the corresponding learning-based algorithms while the differences are insignificant. Because the solution of DMV and PECMV is provably the global optimum [14], LDMV and LPECMV can only perform no better than DMV and PECMV. Moreover, Table IV shows that LDMV achieves a considerable reduction in execution time in contrast to DMV. Besides, despite the colossal amount of computation time of PECMV, LPECMV takes only 43.9001 seconds. Overall, the percentage reduction of the execution time is 53.85% and 99.98% after applying LDMV and PECMV, respectively, which indicates a significant improvement in the time complexity. In summary, the computational cost is reduced substantially by the proposed learning-based optimization without compromising the localization accuracy and consistency, which enables conducting CL updates at higher rates, especially in embedded computing systems.

V. Conclusions

Multi-agent estimation solutions in which the correlation among the local estimates of the agents in cooperative estimation problems are accounted for implicitly, either via the use of conservative uncorrelated estimate upper bounds or by estimating the unknown correlation locally, are attractive because of eliminating persistent inter-agent communications. But, these methods often come with higher computation costs. To reduce the computation cost, this paper proposed to use NN surrogate functions to produce the solution of the time-consuming optimization problems that appear in the implicit approaches to account for correlations. To demonstrate our

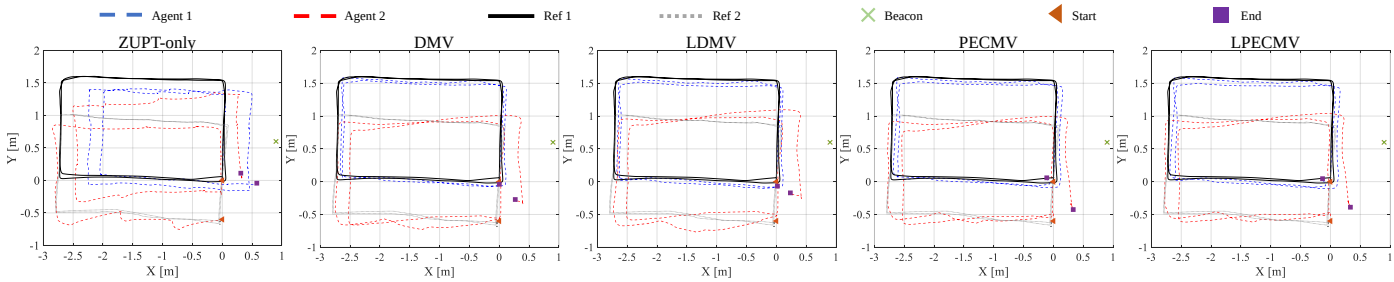


Figure 9 – The estimated trajectories of the agents implementing ZUPT-only, DMV, LDMV, PECMV and LPECMV algorithms.

idea, we focused on the problem of cooperative localization and two particular solutions for this problem from literature: DMV and PECMV methods [14]. We proposed the learning-based algorithms, LDMV and LPECMV, to substitute the time-consuming constrained optimization problems that appear in DMV and PECMV methods by learning their optimal solutions via a NN that is designed elaborately to incorporate the constraints. The training data is obtained by solving the optimization problems directly using the experimental data. Then, the well-trained NNs are used to predict the optimal solution. The efficacy and the generosity of the LDMV and LPECMV are demonstrated in a different set of CL experiments. The agents used the direct predictions from the output of the NNs rather than solve the complex optimization problems in real-time, which makes the computational cost reduced remarkably without compromising the localization accuracy and enables the real-time implementation of the CL algorithms in embedded systems.

ACKNOWLEDGMENT

The authors thank Joe Grasso from NIST and Chico (Chi-Shih) Jao and Minwon Seo from UCI for assisting with the experimental data collection.

REFERENCES

[1] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[2] S. J. Juliera and J. K. Uhlmannb, "Using covariance intersection for SLAM," *Robotics and Autonomous Systems*, vol. 55, pp. 3–20, 2007.

[3] R. Olfati-Saber, "Kalman-consensus filter : Optimality, stability, and performance," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, (Shanghai, China), pp. 7036–7042, 2009.

[4] C.-Y. Chong, S. Mori, F. Govaers, and W. Koch, "Comparison of tracklet fusion and distributed kalman filter for track fusion," in *17th International Conference on Information Fusion (FUSION)*, (Salamanca, Spain), pp. 1–8, 2014.

[5] S. S. Kia, J. Hechtbauer, D. Gogokhiya, and S. Martínez, "Server-assisted distributed cooperative localization over unreliable communication links," *IEEE Transactions on Robotics*, vol. 34, no. 5, pp. 1392–1399, 2018.

[6] S. Julier and J. Uhlmann, "A non-divergent estimation algorithm in the presence of unknown correlations," in *Proceedings of the 1997 American Control Conference (Cat. No.97CH36041)*, vol. 4, (Albuquerque, New Mexico, USA), pp. 2369–2373 vol.4, 1997.

[7] Z. Deng, P. Zhang, W. Qi, J. Liu, and Y. Gao, "Sequential covariance intersection fusion Kalman filter," *Information Sciences*, vol. 189, pp. 293–309, 2012.

[8] J. Hu, L. Xie, and C. Zhang, "Diffusion Kalman filtering based on covariance intersection," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 891–902, 2012.

[9] O. Hlinka, O. Slučiak, F. Hlawatsch, and M. Rupp, "Distributed data fusion using iterative covariance intersection," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (Florence, Italy), pp. 1861–1865, IEEE, 2014.

[10] P. O. Arambel, C. Rago, and R. K. Mehra, "Covariance intersection algorithm for distributed spacecraft state estimation," in *American Control Conference*, (Arlington, Virginia, USA), pp. 4398–4403, 2001.

[11] L. C. Carrillo-Arce, E. D. Nerurkar, J. L. Gordillo, and S. I. Roumeliotis, "Decentralized multi-robot cooperative localization using covariance intersection," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, (Tokyo, Japan), pp. 1412–1417, 2013.

[12] H. Li and F. Nashashibi, "Cooperative multi-vehicle localization using split covariance intersection filter," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 2, pp. 33–44, 2013.

[13] D. Marinescu, N. O'Hara, and V. Cahill, "Data incest in cooperative localisation with the common past-invariant ensemble Kalman filter," in *IEEE Int. Conf. on Information Fusion*, (Istanbul, Turkey), pp. 68–76, 2013.

[14] J. Zhu and S. S. Kia, "Cooperative localization under limited connectivity," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1523–1530, 2019.

[15] J. Sijs, M. Lazar, and P. Bosch, "State fusion with unknown correlation: Ellipsoidal intersection," in *Proceedings of the 2010 American Control Conference*, (Baltimore, Maryland, USA), pp. 3992–3997, 2010.

[16] B. Noack, J. Sijs, M. Reinhardt, and U. D. Hanebeck, "Decentralized data fusion with inverse covariance intersection," *Automatica*, vol. 79, pp. 35–41, 2017.

[17] M. Zarei-Jalalabadi, S. M. B. Malaek, and S. S. Kia, "A track-to-track fusion method for tracks with unknown correlations," *IEEE Control Systems Letters*, vol. 2, no. 2, pp. 189–194, 2018.

[18] S. Leonardos and K. Daniilidis, "A game-theoretic approach to robust fusion and Kalman filtering under unknown correlations," in *American Control Conference*, (Seattle, USA), pp. 2568–2573, 2017.

[19] T. Li, Y. Xin, Y. Song, E. Song, and H. Fan, "Some statistic and information-theoretic results on arithmetic average density fusion," 2021. `https://arxiv.org/pdf/2110.01440.pdf`.

[20] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in neural information processing systems*, vol. 30, (Long Beach, California, USA), 2017.

[21] L. Liu, Y. Cheng, L. Cai, S. Zhou, and Z. Niu, "Deep learning based optimization in wireless network," in *2017 IEEE International Conference on Communications (ICC)*, (Paris, France), pp. 1–6, 2017.

[22] B. Abbasi, T. Babaei, Z. Hosseinifard, and K. Smith-Miles, "Predicting solutions of large-scale optimization problems via machine learning: A case study in blood supply chain management," *Computers Operations Research*, vol. 119, p. 104941, 2020.

[23] J. Eom, H. Kim, S. H. Lee, and S. Kim, "DNN-assisted cooperative localization in vehicular networks," *Energies*, vol. 12, no. 14, 2019.

[24] J. Zhu and S. S. Jia, "Learning-based measurement scheduling for loosely-coupled cooperative localization," 2021. https://arxiv.org/pdf/2112.02843.pdf.

[25] S. Julier and J. K. Uhlmann, "General decentralized data fusion with covariance intersection,"

[26] "Cvxopt." http://cvxopt.org. [Online; accessed 14-August-2014].

[27] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward

networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.
[28] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. Cambridge University Press, 1991.
[29] D. G. Luenberger, Y. Ye, *et al.*, *Linear and nonlinear programming*, vol. 2. Springer, 1984.
[30] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.
[31] E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *IEEE Computer Graphics and Applications*, vol. 25, no. 6, pp. 38–46, 2005.
[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2017.
[33] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.