Spatial Field Reconstruction with Distributed Kernel Least Squares in Mobile Sensor Networks

Ban-Sok Shin, Henning Paul, and Armin Dekorsy
Department of Communications Engineering
University of Bremen, Bremen, Germany
Email: {shin, paul, dekorsy}@ant.uni-bremen.de

***Abstract*—Reconstructing spatial fields by sensor networks is a common problem in environmental monitoring applications. Usually, this task requires nonlinear techniques due to the underlying physical process. The so-called KDiCE algorithm is able to estimate such a spatial field in a distributed fashion by a nonlinear regression using kernel methods. To further enhance its reconstruction performance we consider a mobile sensor network in this paper. We utilize an iterative distributed scheme based on centroidal Voronoi tessellation where the sensors move to the center of mass of their Voronoi region. We include this sensor movement into the KDiCE algorithm and provide performance results regarding a distributed reconstruction of diffusion fields. Our evaluations show a significant gain in the performance by including sensor movement.**

I. Introduction

A common task in environmental monitoring is the reconstruction of a physical quantity over a specific area or space. The quantity of interest can be, e.g., temperature or the concentration of oil in water. For these applications networks of spatially distributed sensors have been considered by the research community [1]–[3]. The sensors measure the physical process at their positions and based on these measurements, in the network a reconstruction of the spatial field is desired. This is a challenging task since most physical processes are modeled by nonlinear functions and thus require nonlinear techniques for successful recovery. E.g., the diffusion equation is commonly used to model the spatial distribution of temperature, gas or oil and results in nonlinear problems. In order to address this challenge kernel methods can be utilized which provide nonlinear regression techniques [4], [5]. The approach is to transform input samples into a high-dimensional space where the nonlinear function can be described in terms of linear operations on the transformed input samples. Furthermore, it is often desired to utilize distributed schemes in a sensor network (SN) which avoid the use of a central unit and thus the risk of a single point of failure. Then, a reconstruction of the spatial field within the SN is possible. These two demands have been addressed by the kernel distributed consensus-based estimation (KDiCE) algorithm proposed by the authors in [6]. Here, a nonlinear regression algorithm based on a kernel least squares (LS) optimization problem has been developed and successfully applied to the distributed reconstruction of diffusion fields.

However, the KDiCE algorithm has been only considered for a network of stationary sensors. Mobile sensors with the ability to move and acquire new measurements of the spatial field could significantly improve the reconstruction performance. In [7], a coverage control algorithm for an optimal sensor positioning based on a density function defined over the considered area is developed. This algorithm aims at achieving a so-called centroidal Voronoi tessellation (CVT) iteratively where sensors are eventually located at the center of mass of their corresponding Voronoi region. In [8] this method is included in a distributed support vector regression (SVR) algorithm based on kernel methods. Also here, the task is to reconstruct a nonlinear spatial field function by mobile sensors. Nevertheless, in contrast to [6] the corresponding algorithm is not based on a distributed optimization problem. Instead calculations which require knowledge from all sensors are modified such that only information from neighboring sensors are required.

In this paper, we extend the KDiCE algorithm [6] by a stage at which sensors change their position based on their reconstructed field. For this, we include the coverage control algorithm from [7] into the KDiCE to improve the reconstruction performance for nonlinear spatial fields. In contrast to [8] where the distributed SVR is run over several iterations to estimate the field function, the extended KDiCE only includes one iteration. Furthermore, we assume noise on the sensors' measurements while in [8] only the noiseless case has been considered.

II. System Model

A SN of J connected sensors placed over a region $\mathcal{X}$ observes a scalar field described by the nonlinear function $f : \mathcal{X} \rightarrow \mathbb{R}$. The function $f(\boldsymbol{x})$ maps positions $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^L$ to a scalar value in $\mathbb{R}$. E.g., it could model the spatial distribution of temperature over the region $\mathcal{X}$. Hence, each sensor j takes the following measurement d_j of the function $f(\boldsymbol{x})$ at its Cartesian position $\boldsymbol{x}_j$:

$$d_j = f(\boldsymbol{x}_j) + n_j. \tag{1}$$

Here, n_j is white Gaussian noise of variance σ_n^2. We describe the SN by a graph with a set of nodes $\mathcal{J}$ representing the sensors and a set of edges connecting the nodes. We assume that the graph is connected, i.e., each node can be reached by every other node by multiple hops. Moreover, sensors with a distance less than r to each other are assumed to be connected by an edge in the graph and can exchange data. Thus for each sensor j there is a set of neighbors $\mathcal{N}_j$ containing those sensors of the network connected to it. Besides, we assume that the sensors know their positions, resulting in J data pairs $\{(\boldsymbol{x}_j, d_j)\}_{j=1}^J$ in the SN. Based on these data pairs our goal is to approximate the unknown nonlinear function $f(\boldsymbol{x})$ to predict the field value at arbitrary positions $\boldsymbol{x}$ within the area covered by the SN.

III. Distributed Nonlinear Regression

A. Problem Formulation

We consider a positive-definite kernel function $\kappa : \mathcal{X}\times\mathcal{X} \rightarrow \mathbb{R}$ being the dot product between two transformed samples in the reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. Input samples $\boldsymbol{x}$ are assigned to kernel functions $\kappa(\boldsymbol{x}, \cdot\,)$ in the RKHS $\mathcal{H}$ with $\boldsymbol{x} \mapsto \kappa(\boldsymbol{x}, \cdot\,)$ by a mapping function $\Phi : \mathcal{X} \rightarrow \mathcal{H}$. Thus, the RKHS enables a richer representation of the input samples such that a nonlinear function $f(\boldsymbol{x})$ can be approximated in terms of linear combinations of transformed samples $\kappa(\boldsymbol{x}, \cdot\,)$. Based on the data pairs $\{(\boldsymbol{x}_j, d_j)\}_{j=1}^J$ available in the SN our goal is to find a function $\hat{f}(\boldsymbol{x})$ in the RKHS $\mathcal{H}$ which fulfills the following LS optimization problem:

$$f^* = \underset{\hat{f}\in\mathcal{H}}{\arg\min} \sum_{j=1}^{J} (d_j - \hat{f}(\boldsymbol{x}_j))^2. \tag{2}$$

We can make use of the *representer theorem* stating that there exists at least one function $\hat{f}(\,\cdot\,)$ minimizing the cost (2) which can be described by a linear combination of kernel functions weighted by coefficients w_ℓ [4]:

$$\hat{f}(\,\cdot\,) = \sum_{\ell=1}^{J} w_\ell \kappa(\boldsymbol{x}_\ell, \cdot\,), \tag{3}$$

where the set $\{\kappa(\boldsymbol{x}_\ell, \cdot\,)\}_{\ell=1}^J$ is called the dictionary $\mathcal{D}$ containing kernel evaluations based on all sensor positions in the network. With this representation of $\hat{f}(\,\cdot\,)$ we can reformulate the LS optimization problem with respect to (w.r.t.) the weight vector $\boldsymbol{w} = [w_1, \ldots, w_J]^{\mathrm{T}}$ and the vector $\boldsymbol{\kappa}_{\mathcal{D}}(\boldsymbol{x}_j) = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}_j), \ldots, \kappa(\boldsymbol{x}_J, \boldsymbol{x}_j)]^{\mathrm{T}}$ correlating $\boldsymbol{x}_j$ to the entries in the dictionary $\mathcal{D}$:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}\in\mathbb{R}^J}{\arg\min} \sum_{j=1}^{J} (d_j - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\kappa}_{\mathcal{D}}(\boldsymbol{x}_j))^2. \tag{4}$$

Thus, problem (2) is modified such that the optimal weight vector $\boldsymbol{w}^*$ needs to be found rather than the function $f^*(\,\cdot\,)$ itself.

B. Kernel DiCE Algorithm

In order to solve the LS optimization problem (4) in a distributed manner we introduce local weight vectors $\boldsymbol{w}_j$ per sensor j and add a consensus constraint on the weight vectors among neighboring sensors:

$$\{\boldsymbol{w}_j^* | j \in \mathcal{J}\} = \underset{\{\boldsymbol{w}_j | j\in\mathcal{J}\}}{\arg\min} \sum_{j=1}^{J} (d_j - \boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{\kappa}_{\mathcal{D}}(\boldsymbol{x}_j))^2 \tag{5a}$$

$$\text{s.t.} \quad \boldsymbol{w}_j = \boldsymbol{w}_i, \quad \forall j \in \mathcal{J}, i \in \mathcal{N}_j. \tag{5b}$$

By the consensus constraint we achieve that all weight vectors converge to the same solution, namely the central kernel LS solution. Due to a direct coupling between neighboring weight vectors $\boldsymbol{w}_j$ and $\boldsymbol{w}_i$ in the consensus constraint solving this problem will not result in an algorithm that enables a parallel update of the weight vectors. Thus, the auxiliary variable $\boldsymbol{z}_j$ per sensor j is introduced and the constraint is modified such that $\boldsymbol{w}_j = \boldsymbol{z}_i, \boldsymbol{w}_j = \boldsymbol{z}_j, \forall j \in \mathcal{J}, i \in \mathcal{N}_j$. Then, the alternating direction method of multipliers [9] can be used to solve this constrained optimization problem resulting in the iterative algorithm KDiCE. Details regarding the derivation of the KDiCE can be found in [6], [10]. We achieve the following update equations for the variables $\boldsymbol{w}_j, \boldsymbol{z}_j$ and $\boldsymbol{\lambda}_{ji}$ per iteration k with $\mathcal{N}_j' = \mathcal{N}_j \cup \{j\}$:

$$\boldsymbol{z}_j^k = \frac{\mu}{|\mathcal{N}_j'|} \sum_{i\in\mathcal{N}_j'} \frac{1}{\mu}\boldsymbol{w}_i^{k-1} - \boldsymbol{\lambda}_{ij}^{k-1} \tag{6a}$$

$$\boldsymbol{\lambda}_{ji}^k = \boldsymbol{\lambda}_{ji}^k - \frac{1}{\mu}\left(\boldsymbol{w}_j^{k-1} - \boldsymbol{z}_i^k\right) \tag{6b}$$

$$\begin{aligned}\boldsymbol{w}_j^k = {} & \left(\boldsymbol{\kappa}_{\mathcal{D}}(\boldsymbol{x}_j)\boldsymbol{\kappa}_{\mathcal{D}}(\boldsymbol{x}_j)^{\mathrm{T}} + \frac{|\mathcal{N}_j'|}{\mu}\boldsymbol{I}_J\right)^{-1} \\ & \cdot \left(d_j\boldsymbol{\kappa}_{\mathcal{D}}(\boldsymbol{x}_j) + \sum_{i\in\mathcal{N}_j'} \frac{1}{\mu}\boldsymbol{z}_i^k + \boldsymbol{\lambda}_{ji}^k\right).\end{aligned} \tag{6c}$$

The variables are initialized as $\boldsymbol{w}_j^0 = \boldsymbol{\lambda}_{ij}^0 = \boldsymbol{\lambda}_{ji}^0 = \boldsymbol{0}$ for all nodes, while μ is a positive step size. Following (6a)-(6c), each sensor j first calculates its auxiliary variable $\boldsymbol{z}_j^k$ and broadcasts it to its neighbors. After receiving the auxiliary variables from its neighbors, each sensor updates its Lagrange multipliers $\boldsymbol{\lambda}_{ji}^k$ and transmits them to its neighbors. Note that for each neighboring sensor $i \in \mathcal{N}_j$ a different Lagrange multiplier $\boldsymbol{\lambda}_{ji}^k$ needs to be transmitted, so that no broadcast transmission is possible. After receiving the Lagrange multipliers from its neighbors each sensor finally updates its weight vector $\boldsymbol{w}_j^k$. The updated weight vector $\boldsymbol{w}_j^k$ is again broadcast to the neighboring sensors and the next iteration is initiated.

Per iteration k, each sensor j is able to reconstruct the unknown field $f(\boldsymbol{x})$ at arbitrary positions $\boldsymbol{x}$ using its individual weight vector $\boldsymbol{w}_j^k$ and the representer theorem (3):

$$\hat{f}_j^k(\boldsymbol{x}) = \sum_{\ell=1}^{J} w_{j,\ell}^k \kappa(\boldsymbol{x}_\ell, \boldsymbol{x}) = (\boldsymbol{w}_j^k)^{\mathrm{T}}\boldsymbol{\kappa}_{\mathcal{D}}(\boldsymbol{x}) \tag{7}$$

where $w_{j,\ell}^k$ is the ℓ-th entry of the vector $\boldsymbol{w}_j^k$. Then each sensor j can estimate the field $f(\boldsymbol{x})$ over the region covered by the SN.

IV. Mobile Sensor Positioning

As shown above, the KDiCE algorithm is able to reconstruct the nonlinear field function $f(\boldsymbol{x})$ in a distributed fashion. In order to improve its reconstruction performance we now consider the case of mobile sensors which can acquire new measurements of the field $f(\boldsymbol{x})$ as they change their position $\boldsymbol{x}_j$. The question that arises is how the sensors need to move in order to increase the reconstruction performance.

To address this question we utilize the work of [7] where a distributed algorithm for sensor positioning based on CVT [11] is developed.

A. Problem Formulation

Following the work in [7], we define the locational cost function over all J sensors w.r.t. the set of partitions $\mathcal{W} = \{W_1, \ldots, W_J\}$, which separate the considered area and the set of sensor positions $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_J\}$:

$$H(X, \mathcal{W}) = \sum_{j=1}^{J} \int_{W_j} \frac{1}{2} ||\boldsymbol{x} - \boldsymbol{x}_j||^2 f(\boldsymbol{x}) d\boldsymbol{x}. \tag{8}$$

In general, $f(\boldsymbol{x})$ is a density function defined over the considered area $\mathcal{X}$. In our case $f(\boldsymbol{x})$ is the unknown field function which is used for the following derivations. Later we replace $f(\boldsymbol{x})$ by its estimation $\hat{f}_j^k(\boldsymbol{x})$ for each sensor which is provided by the KDiCE algorithm. The choice of $f(\boldsymbol{x})$ is reasonable since it gives higher weight to significant areas of the unknown field function where we assume that $f(\boldsymbol{x})$ is non-negative. The term $||\boldsymbol{x} - \boldsymbol{x}_j||^2$ in $H(X, \mathcal{W})$ is used to indicate a higher cost the greater the distance from the sensor position $\boldsymbol{x}_j$ is. This is due to the assumption that the sensing performance of a sensor degrades with a greater distance from its location. To find $\mathcal{W}$ and X the cost $H(X, \mathcal{W})$ needs to be minimized w.r.t. the J regions in the partition set $\mathcal{W}$ and the sensor positions in X.

B. Centroidal Voronoi Tessellation

For fixed sensor positions the optimal partition $\mathcal{W}$ is given by the Voronoi tessellation $\mathcal{V} = \{V_1, \ldots, V_J\}$ [7]. A Voronoi region V_j with the sensor position $\boldsymbol{x}_j$ being its *generating point* is defined as

$$V_j = \{\boldsymbol{x} \in \mathcal{X} | \; ||\boldsymbol{x} - \boldsymbol{x}_j|| \leq ||\boldsymbol{x} - \boldsymbol{x}_i||, \forall i \neq j\}. \tag{9}$$

Thus, a Voronoi cell contains those positions $\boldsymbol{x}$ which lie closer to the generating point $\boldsymbol{x}_j$ than to any other generating point $\boldsymbol{x}_i \in X \backslash \{\boldsymbol{x}_j\}$. Furthermore, to determine a Voronoi cell V_j at sensor j knowledge of sensor positions $\boldsymbol{x}_i$ from neighboring sensors is sufficient. Thus, a distributed calculation of the Voronoi partition is possible assuming that neighboring sensors share a communication link. With the Voronoi partition $\mathcal{V}(X)$ on the sensor positions X, the set of partition $\mathcal{W}$ is determined and the locational cost function (8) reads

$$H_{\mathcal{V}} = H(X, \mathcal{V}(X)) = \sum_{j=1}^{J} \int_{V_j} \frac{1}{2} ||\boldsymbol{x} - \boldsymbol{x}_j||^2 f(\boldsymbol{x}) d\boldsymbol{x}. \tag{10}$$

Hence, based on the Voronoi tessellation of the sensor positions $\mathcal{V}(X)$ the optimal sensor positions minimizing $H_{\mathcal{V}}$ need to be found. To find these positions, the derivative of the cost function $H_{\mathcal{V}}$ w.r.t. the sensor position $\boldsymbol{x}_j$ needs to be calculated and set to zero. For this, we first define the mass m_{V_j} and the centroid (center of mass) $\boldsymbol{c}_{V_j}$, respectively, regarding a region V_j and a scalar field function $f(\boldsymbol{x})$ over this region:

$$m_{V_j} = \int_{V_j} f(\boldsymbol{x}) d\boldsymbol{x} \tag{11}$$

$$\boldsymbol{c}_{V_j} = \frac{1}{m_{V_j}} \int_{V_j} \boldsymbol{x} f(\boldsymbol{x}) d\boldsymbol{x} \tag{12}$$

The derivative of the cost function $H_{\mathcal{V}}$ w.r.t. the sensor position $\boldsymbol{x}_j$ is then calculated as

$$\begin{aligned} \frac{\partial H_{\mathcal{V}}}{\partial \boldsymbol{x}_j} &= \frac{\partial}{\partial \boldsymbol{x}_j} \sum_{j=1}^{J} \int_{V_j} \frac{1}{2} ||\boldsymbol{x} - \boldsymbol{x}_j||^2 f(\boldsymbol{x}) d\boldsymbol{x} \\ &= - \int_{V_j} \boldsymbol{x} f(\boldsymbol{x}) - \boldsymbol{x}_j f(\boldsymbol{x}) d\boldsymbol{x} \\ &= -m_{V_j} (\boldsymbol{c}_{V_j} - \boldsymbol{x}_j). \end{aligned}$$

From the above equation it is obvious that setting $\boldsymbol{x}_j = \boldsymbol{c}_{V_j}$ minimizes the cost function (10), i.e., the sensors should be positioned at the centroids of their corresponding Voronoi cell. In the literature, such a constellation is called *centroidal Voronoi tessellation*. Accordingly, sensors need to move into the direction of the centroid of their Voronoi cell to improve their positioning based on the function $f(\boldsymbol{x})$.

C. Distributed Lloyd Algorithm

To achieve such a CVT the Lloyd algorithm from [12] can be utilized. The Lloyd algorithm is an iterative approach which based on the current generating points X^k at iteration k determines the Voronoi regions $\mathcal{V}(X^k)$ and calculates the centroids $\{\boldsymbol{c}_{V_j}^k\}_{j=1}^J$ of each region. These centroids are then set as the new generators of each region as $X^{k+1} = \{\boldsymbol{c}_{V_j}^k\}_{j=1}^J$ and the Voronoi regions and corresponding centroids are updated. This process is repeated until the generating points correspond to the centroids in each cell resulting in the desired CVT.

For a realization of the Lloyd algorithm in the SN we let the sensors move into the direction of their cell's centroid. The sensors change their position $\boldsymbol{x}_j$ by $\Delta \boldsymbol{x}_j$ following a gradient-descent step with a step size β [7]:

$$\begin{aligned} \Delta \boldsymbol{x}_j &= -\beta \frac{\partial H_{\mathcal{V}}}{\partial \boldsymbol{x}_j} \\ &= \beta m_{V_j} (\boldsymbol{c}_{V_j} - \boldsymbol{x}_j). \end{aligned} \tag{13}$$

This gradient is calculated in each iteration k based on the current Voronoi cells $\mathcal{V}(X^k)$, the current mass m_{V_j} and the current centroid $\boldsymbol{c}_{V_j}$. Then each sensor j moves into the direction of its centroid by

$$\boldsymbol{x}_j^{k+1} = \boldsymbol{x}_j^k + \Delta \boldsymbol{x}_j^k \tag{14}$$

and the next iteration is initiated by updating the Voronoi partition based on the new sensor positions X^{k+1}. Eventually, a CVT is achieved where the sensors' positions correspond to the centroids of the Voronoi cells [13].

Regarding a distributed implementation of the Lloyd algorithm we note that the determination of the Voronoi region V_j (9) at each sensor j requires information from neighboring sensors only, namely their sensor positions $\boldsymbol{x}_i, i \in \mathcal{N}_j$. Once the Voronoi cells are determined, the mass m_{V_j}, the centroid $\boldsymbol{c}_{V_j}$ and the gradient step $\Delta \boldsymbol{x}_j$ can be computed locally at sensor j based on the knowledge of $f(\boldsymbol{x})$. Detailed distributed implementations can be found in [7]. Hence, the Lloyd algorithm can be performed in a distributed manner within a SN and can therefore be combined with the KDiCE algorithm.

V. Combining CVT and KDiCE

For a combination of the KDiCE with the distributed Lloyd algorithm from above we adapt the system model (1) such that at each iteration k based on the current position $\boldsymbol{x}_j^k$ each sensor j takes the following measurement:

$$d_j^k = f\left(\boldsymbol{x}_j^k\right) + n_j^k. \tag{15}$$

Then with d_j^k each sensor j performs the KDiCE algorithm according to (6a)-(6c) where d_j in (6c) is replaced by d_j^k. After that each sensor j performs a reconstruction of the field function by (7) to achieve $\hat{f}_j^k(\boldsymbol{x})$:

$$\hat{f}_j^k(\boldsymbol{x}) = \sum_{\ell=1}^{J} w_{j,\ell}^k \kappa(\boldsymbol{x}_\ell^k, \boldsymbol{x}) = (\boldsymbol{w}_j^k)^{\mathrm{T}} \boldsymbol{\kappa}_{\mathcal{D}}(\boldsymbol{x}) \tag{16}$$

Note that this step requires knowledge of all current sensor positions $\{\boldsymbol{x}_\ell^k\}_{\ell=1}^J$ at each sensor j per iteration k. Hence, we need to update the dictionary with $\mathcal{D} = \{\kappa(\boldsymbol{x}_\ell^k, \cdot)\}_{\ell=1}^J$ based on the new sensor positions. For this step we assume that each sensor can acquire the updated dictionary by efficient flooding algorithms [14]. Then, after determining the Voronoi partitions $\mathcal{V}(X^k)$ the mass $m_{V_j^k}$ and the centroid $\boldsymbol{c}_{V_j^k}$ are calculated by each sensor j based on its reconstruction $\hat{f}_j^k(\boldsymbol{x})$:

$$m_{V_j^k} = \int_{V_j^k} \hat{f}_j^k(\boldsymbol{x}) d\boldsymbol{x} \tag{17a}$$

$$\boldsymbol{c}_{V_j^k} = \frac{1}{m_{V_j^k}} \int_{V_j^k} \boldsymbol{x} \hat{f}_j^k(\boldsymbol{x}) d\boldsymbol{x}. \tag{17b}$$

With the mass $m_{V_j^k}$ and the centroid $\boldsymbol{c}_{V_j^k}$ each sensor j computes the gradient $\Delta \boldsymbol{x}_j^k$ according to (13) and changes its position by it. Then the next iteration is initiated by sampling the function $f(\boldsymbol{x})$ at the updated sensor positions $\{\boldsymbol{x}_j^{k+1}\}_{j=1}^J$ and performing the above described steps. Algorithm 1 summarizes the CVT-KDiCE algorithm.

VI. Performance Evaluation

We evaluate the performance of the CVT-KDiCE algorithm with regard to the reconstruction of static diffusion fields which are known to be highly nonlinear. A diffusion field $f(\boldsymbol{x})$ at position $\boldsymbol{x}$ with M instantaneous and localized sources each having intensity c_m and fixed position vector $\boldsymbol{p}_m$ can be described by [15]

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} \frac{c_m}{4\pi\nu} \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{p}_m||^2}{4\nu}\right), \tag{18}$$

Algorithm 1 CVT-KDiCE

```
1:  Initialize $\boldsymbol{w}_j^0 = \boldsymbol{\lambda}_{ij}^0 = \boldsymbol{\lambda}_{ji}^0 = \boldsymbol{0}$
2:  for iteration $k$ and sensor $j$ do
3:      acquire measurement $d_j^k$ at $\boldsymbol{x}_j^k$
4:      compute $\boldsymbol{z}_j^k$ with (6a) and broadcast to current neighbors $i \in \mathcal{N}_j^k$
5:      compute $\boldsymbol{\lambda}_{ji}^k$ with (6b) and transmit to current neighbors $i \in \mathcal{N}_j^k$
6:      compute $\boldsymbol{w}_j^k$ with (6c) based on $d_j^k$ and broadcast to current neighbors $i \in \mathcal{N}_j^k$
7:      reconstruct field by $\hat{f}_j^k(\boldsymbol{x})$ with (16)
8:      determine Voronoi region $V_j^k$ with (9)
9:      compute mass $m_{V_j^k}$ (17a) and centroid $\boldsymbol{c}_{V_j^k}$ (17b)
10:     compute $\Delta \boldsymbol{x}_j^k$ with (13) and move sensor to new position
11:     update dictionary $\mathcal{D}$ with new sensor positions, flood
12: end for
```

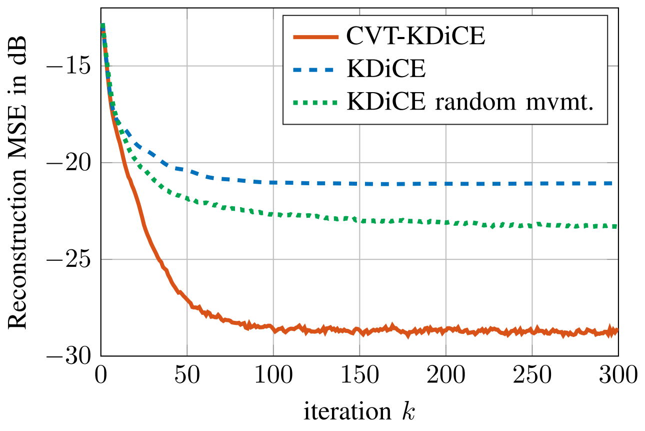

Fig. 1. MSE of the KDiCE and CVT-KDiCE for a static diffusion field with $M = 1$ source and $J = 20$ sensors.

where ν is the diffusion constant of the medium. For the following evaluations we assume a diffusion constant of $\nu = 0.01$ and white Gaussian noise with power $\sigma_n^2 = 0.01$ on the sensor measurements. For the reconstruction stage by the KDiCE algorithm we use the Gaussian kernel given by

$$\kappa(\boldsymbol{x}_j, \boldsymbol{x}_n) = \exp\left(-\frac{||\boldsymbol{x}_j - \boldsymbol{x}_n||^2}{2\zeta^2}\right), \tag{19}$$

where ζ is the kernel bandwidth which we adapt to the diffusion constant by setting $\zeta = \sqrt{2\nu}$. We assume that the diffusion constant ν can be estimated properly beforehand since the environment of the SN is known. Furthermore, the step-size of the KDiCE is set to $\mu = 8$ and of the CVT to $\beta = 10$. This choice of parameter values led to a good performance with stable behavior in our evaluations. Regarding the topology of the network, sensors are placed randomly on the unit-square, where an error-free communication link is established between sensors having a distance less than $r = 0.22$ to each other. Note that through the movement of the sensors the individual neighborhood $\mathcal{N}_j^k$ per node j can change

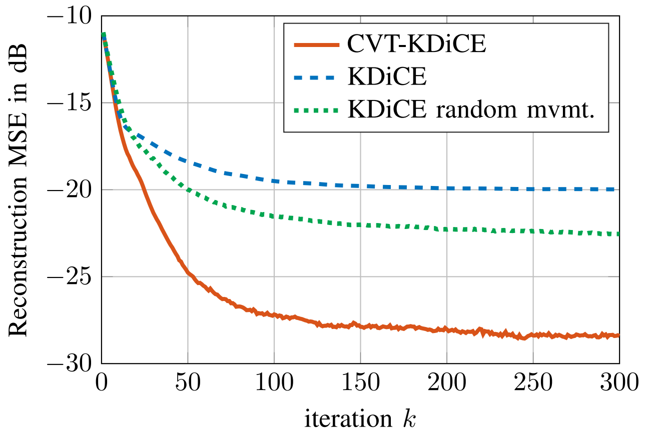

Fig. 2. MSE of the KDiCE and CVT-KDiCE for a static diffusion field with $M = 2$ sources and $J = 30$ sensors.

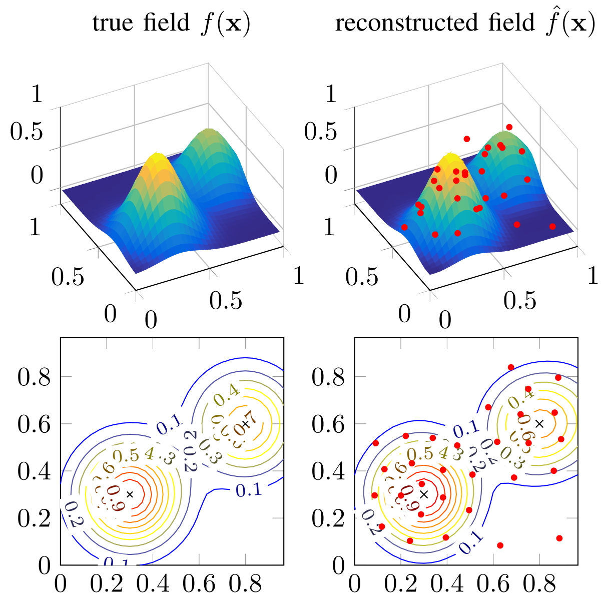

Fig. 3. Surface and contour plot of true and reconstructed field after 300 iterations for one sensor. Red circles indicate the sensor positions and the measured field values. Black crosses indicate the position of the diffusion sources.

over the iterations. For each evaluation 100 trials with different realizations of noise and network topologies are performed.

Fig. 1 depicts the reconstruction MSE over 300 iterations for $M = 1$ source at $\boldsymbol{p}_1 = [0.5, 0.5]^{\mathrm{T}}$ with $c_1 = 1$ and $J = 20$ sensors. We show the KDiCE, CVT-KDiCE and KDiCE with a random sensor movement where the movement step is based on a normal distribution of variance $\sigma_m = 0.01$. We compute the reconstruction MSE as described in [6]. For the first approximately ten iterations all algorithms perform similarly before the movement of the sensors significantly increases the reconstruction performance of the CVT-KDiCE. We observe that it clearly outperforms the KDiCE and the random movement in terms of steady-state error by a gain of 7 dB. Note that by the required update of the dictionary with the new sensor positions the generated communication overhead will be increased in CVT-KDiCE. Comparison over the communication overhead will be covered in future research.

In Fig. 2 the reconstruction performance for $M = 2$ sources with positions $\boldsymbol{p}_1 = [0.3, 0.3]^{\mathrm{T}}, \boldsymbol{p}_2 = [0.8, 0.6]^{\mathrm{T}}$ and intensities $c_1 = 1, c_2 = 0.7$ is compared among the algorithms. Since one further source is present we increase the number of sensors to $J = 30$. Also here a significant performance gain for the CVT-KDiCE over the KDiCE and the random movement can be seen. Nevertheless, due to the additional diffusion source the performance of both algorithms slightly degrades compared to Fig. 1. Corresponding surface and contour plots for a reconstruction by the CVT-KDiCE for one sensor node after 300 iterations are shown as an example in Fig. 3. As mentioned before, we observe a high reconstruction performance of the unknown field function $f(\boldsymbol{x})$. The two peaks caused by the two diffusion sources are clearly visible in the reconstruction. Furthermore regarding the positioning of the sensors we notice that these move to the regions of interest which lie around the diffusion sources due to the included CVT stage. The specific movement of the sensors is illustrated in Fig. 4 with the trajectories and Voronoi regions of the sensors.

Fig. 5 depicts the reconstruction MSE over the number

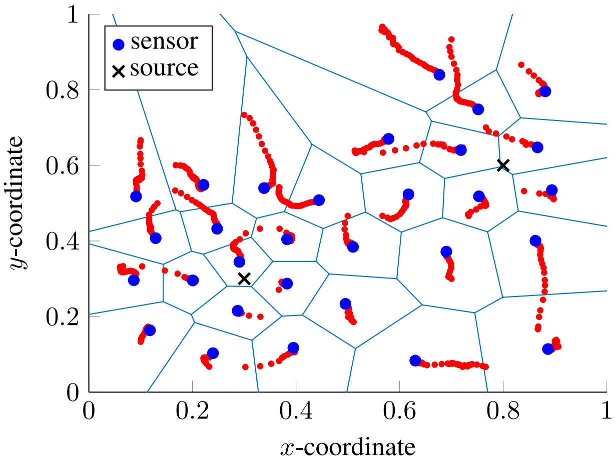

Fig. 4. Sensor trajectories and Voronoi regions of the CVT-KDiCE algorithm for a static diffusion field with $M = 2$ sources and $J = 30$ sensors.

of sensor nodes for both algorithms with different numbers of iterations. As expected, the CVT-KDiCE outperforms the KDiCE. Specifically, it requires only 30 iterations in order to achieve the same performance as the KDiCE algorithm with 100 iterations. Furthermore, we observe significant performance gains for the CVT-KDiCE if more than 20 sensors are used. For more than 40 sensors the gain lies between 5 dB and 7 dB. For the previous performance evaluations we assumed that the dictionary $\mathcal{D}$ is updated at every sensor j per iteration k based on the new sensor positions. As mentioned before, we assume that this step in Algorithm 1 is processed by efficient flooding algorithms which distribute the current sensor positions X^k to each sensor. However, such algorithms

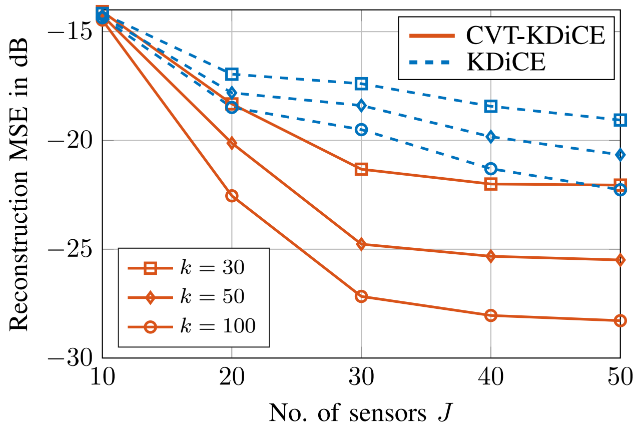

Fig. 5. MSE of the CVT-KDiCE over no. of sensors J for a static diffusion field with $M = 2$ sources.

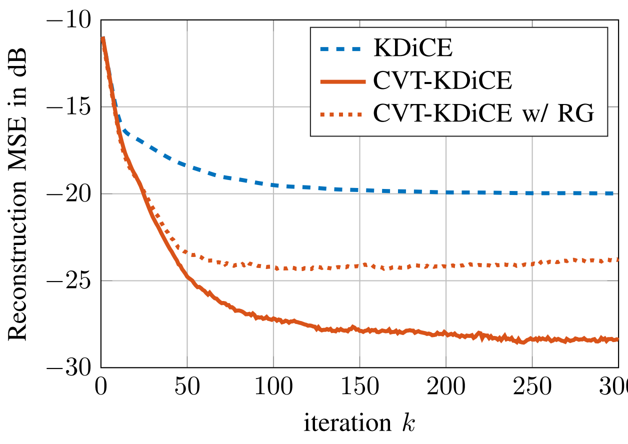

Fig. 6. MSE of KDiCE, CVT-KDiCE and CVT-KDiCE with regular grid dictionary for a static diffusion field with $M = 2$ sources.

increase the communication overhead in the network. In order to avoid this drawback, we initialize the dictionary $\mathcal{D}$ at each sensor j with a set of fixed positions instead of updating it in each iteration. For the dictionary we choose a set of J distributed positions building a regular grid (RG) over the unit-square area. Then each sensor j uses this dictionary during the execution of the CVT-KDiCE algorithm. Obviously, we expect the reconstruction performance to decrease since the dictionary based on the true sensor positions at each iteration is not used by the sensors. This is depicted in Fig. 6 where we can observe a loss of $4\,\mathrm{dB}$ w.r.t. the CVT-KDiCE with the true dictionary. Nevertheless, for the first 50 iterations both algorithms show the same performance. Note that with a pretuned dictionary the overhead is lower such that a comparison over the communication exchanges against the CVT-KDiCE should show a faster convergence. Furthermore, compared to the KDiCE we still see a gain of $4\,\mathrm{dB}$ in the MSE. Of course, depending on the position of the sources this gain may vary. Although the dictionary $\mathcal{D}$ is not updated with the new sensor positions the CVT-KDiCE outperforms the KDiCE while keeping the same communication overhead per iteration.

VII. Conclusion

In this paper, we presented the CVT-KDiCE as a distributed scheme to reconstruct a nonlinear spatial field function combining kernel regression techniques with a CVT-based sensor movement. We showed its performance gains compared to the KDiCE algorithm with fixed sensors. Regarding the update of the dictionary we proposed a fixed dictionary to avoid a higher communication overhead caused by flooding algorithms. The fixed dictionary outperformed the KDiCE algorithm while keeping the same communication overhead. Future work involves a distributed update of the dictionary per sensor node as well as a mutual optimization of the LS and the locational cost function.

Acknowledgment

The work leading to this publication was partially funded by the German Research Foundation (DFG) under grant Pa2507/1.

References

[1] S. N. Simic and S. Sastry, "Distributed environmental monitoring using random sensor networks," in *Proc. of the 2nd Int. Workshop on Information Processing in Sensor Networks*, 2003, pp. 582–592.
[2] V. Schwarz and G. Matz, "Distributed reconstruction of time-varying spatial fields based on consensus propagation," *ICASSP*, pp. 2926–2929, 2010.
[3] J. Murray-Bruce and P. L. Dragotti, "Estimating localized sources of diffusion fields using spatiotemporal sensor measurements," *IEEE Transactions on Signal Processing*, vol. 63, no. 12, pp. 3018–3031, 2015.
[4] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press, 2001.
[5] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering*. John Wiley & Sons, 2010.
[6] B.-S. Shin, H. Paul, and A. Dekorsy, "Distributed kernel least squares for nonlinear regression applied to sensor networks," in *EUSIPCO*, September 2016.
[7] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.
[8] B. Lu, D. Gu, and H. Hu, "Environmental field estimation of mobile sensor networks using support vector regression," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2926–2931, 2010.
[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.
[10] H. Paul, J. Fliege, and A. Dekorsy, "In-network-processing: Distributed consensus-based linear estimation," *IEEE Commun. Lett.*, vol. 17, no. 1, pp. 59–62, 2013.
[11] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: Applications and algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, 1999.
[12] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.
[13] Q. Du, M. Emelianenko, and L. Ju, "Convergence properties of the Lloyd algorithm for computing the centroidal voronoi tessellations," *SIAM Journal on Numerical Analysis*, vol. 44, no. 1, pp. 102–119, 2006.
[14] H. Liu, P.-J. Wan, X. Jia, X. Liu, and F. F. Yao, "Efficient flooding scheme based on 1-hop information in mobile ad hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 5, pp. 1–14, 2007.
[15] Y. M. Lu, P. L. Dragotti, and M. Vetterli, "Localizing point sources in diffusion fields from spatiotemporal samples," in *Proc. of Int. Conf. on Sampling Theory and Applications (SampTA)*, May 2011.