Robust non-linear regression: A greedy approach employing kernels

George Papageorgiou, *Student Member, IEEE*, Pantelis Bouboulis, *Member, IEEE*, and Sergios Theodoridis, *Fellow, IEEE*

***Abstract*—We consider the task of robust non-linear regression in the presence of both bounded noise and outliers. Assuming that the unknown non-linear function belongs to a Reproducing Kernel Hilbert Space (RKHS), our goal is to estimate the set of the associated unknown parameters. Due to the presence of outliers, common techniques such as the Kernel Ridge Regression (KRR) or the Support Vector Regression (SVR) turn out to be inadequate. Instead, we employ sparse modeling arguments to explicitly model and estimate the outliers, adopting a greedy approach. The proposed robust scheme, i.e., Kernel Greedy Algorithm for Robust Denoising (KGARD), is inspired by the classical Orthogonal Matching Pursuit (OMP) algorithm. Specifically, the proposed method alternates between a KRR task and an OMP-like selection step. Convergence properties as well as theoretical results concerning the identification of the outliers are provided. Moreover, KGARD is compared against other cutting edge methods in terms of its performance and the derived theoretical results are verified via a set of experiments. Finally, the proposed robust estimation framework is applied to the task of image denoising, and its enhanced performance in the presence of outliers is demonstrated.**

***Index Terms*—Robust non-linear regression employing kernels, modeling in RKH spaces, sparse modeling via greedy algorithms, kernel greedy algorithm for robust denoising, KGARD, robust kernel ridge regression, RKRR.**

I. Introduction

THE problem of function estimation has attracted significant attention in the machine learning and signal processing communities over the past decades. In this paper, we target the specific task of regression, which is typically described as follows: given a training set of the form $\mathcal{D} = \{(y_i, \boldsymbol{x}_i)\}_{i=1}^N$, we aim to estimate the input-output relation between $\boldsymbol{x}_i$ and y_i; i.e., a function f, such that $f(\boldsymbol{x}_i)$ is "close" to y_i, for all i. This is usually achieved by employing a *loss function*, i.e., a function $C(\boldsymbol{x}_i, y_i, f(\boldsymbol{x}_i))$, that measures the deviation between the observed values, y_i, and the predicted values, $f(\boldsymbol{x}_i)$, and minimizing the so called *Empirical Risk*, i.e., $\sum_{i=1}^N C(\boldsymbol{x}_i, y_i, f(\boldsymbol{x}_i))$. For example, in the least squares regression, one adopts the squared error, i.e., $(y_i - f(\boldsymbol{x}_i))^2$, and minimizes a quadratic function.

Naturally, the choice for f strongly depends on the underlying true model. In this paper, we assume that f belongs to an RKHS. These are inner product function spaces, in which every function is reproduced by an associated (space defining) kernel; that is, for every $\boldsymbol{x} \in \mathcal{X}$, there exists $\kappa(\cdot, \boldsymbol{x}) \in \mathcal{H}$, such that $f(\boldsymbol{x}) = \langle f, \kappa(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}}$. This is the case that has been addressed (amongst others) by two very popular and well-established methods which are commonly referred to as the *Kernel Ridge Regression* (KRR) and the *Support Vector Regression* (SVR).

This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program Education and Lifelong Learning of the National Strategic Reference Framework (NSRF) - Research Funding Program: Aristeia I - 621.

Another important issue that determines the quality of the estimation is the underlying noise model. For the most common noise sources (Gaussian, Laplacian, etc.) the estimation is performed via the KRR by solving a (regularized) Least Squares task, [1]. However, when outliers are present or when the noise distribution exhibits long tails (commonly originating from another noisy source) the performance of the KRR degrades significantly. The non-robustness of the Least Squares estimator is well known even for the case of the simple linear regression task; a variety of methods that deal with this problem have been established over the years, e.g., [1]–[13]. On the other hand, the development of robust estimators for the KRR has been addressed only recently; the task is known as the Robust Kernel Ridge Regression (RKRR), [14], [15]. In this case, y_i is assumed to be generated by

$$y_i = \underline{f}(\boldsymbol{x}_i) + v_i, \; i = 1, ..., N, \tag{1}$$

where v_i are random noise samples which may contain outliers. The present paper focuses on this task in the special case where the unknown function, $\underline{f}$, is assumed to lie in an RKHS, $\mathcal{H}$. It should be noted that both SVR and KRR can be employed to address this problem, but the presence of outliers degrades their performance significantly due to over-fitting, [16], [17]. Of course, in SVR this effect is not as dominant as in typical KRR, due to the ℓ_1 loss function that it is employed.

Our proposed method adopts a model of the form $y = f(\boldsymbol{x})$, where $f \in \mathcal{H}$. In addition, a decomposition of the noise into two parts, a sparse outlier vector $\boldsymbol{u}$ and the inlier vector $\boldsymbol{\eta}$ is employed. The method employs a two step algorithmic procedure attempting to estimate both the outliers and also the original (unknown) function $\underline{f}$. The algorithm alternates between a) a greedy-type algorithm based on the popular *Orthogonal Matching Pursuit* (OMP) [18]–[20], that selects the dominant outlier sample in each step, and b) a kernel ridge regression task to update the current estimate of $\underline{f}$. Results regarding convergence as well as the theoretical properties concerning the identification of the outliers are also provided. Moreover, comparisons against the previously published approaches, based on the Bayesian framework and on the minimization of the ℓ_1-norm for the sparse outlier vector, are performed.

The rest of the paper is organized as follows. In section II, the basic properties of RKHS are summarized, and in section III the problem is formulated and comparable state-of-the-art methods are presented. Next, in section IV, the proposed scheme is introduced and described in detail. Section V provides the theoretical results regarding the convergence of the scheme as well as the identification of the outliers. In section VI, extended tests of the proposed scheme against other cutting edge methods are performed. There, the efficiency of the proposed method in terms of both the achieved mean square error (MSE) as well as the convergence time is studied. Finally, in section VII, the method is applied to the task of robust image denoising in order to optimally remove the noise component that comprises a mix of impulsive and Gaussian sources.

Notation: Throughout this work, capital calligraphic letters are employed to denote sets, e.g., $\mathcal{S}$, where $\mathcal{S}^c$ denotes the complement of $\mathcal{S}$. Small letters denote scalars, e.g., ε, while bold capital letters denote matrices, e.g., $\boldsymbol{X}$, bold lowercase letters are reserved for vectors, e.g., $\boldsymbol{\theta}$ (each vector is regarded as a column vector) and the symbol $\cdot^T$ denotes the transpose of the respective matrix/vector. Also, $\mathrm{diag}(\boldsymbol{a})$, where $\boldsymbol{a}$ is a vector, denotes the respective square diagonal matrix[1], while $\mathrm{supp}(\boldsymbol{a})$ denotes the support set of the vector $\boldsymbol{a}$. The $j-$th column of matrix $\boldsymbol{X}$ is denoted by $\boldsymbol{x}_j$ and the element of the $i-$th row and $j-$th column of matrix $\boldsymbol{X}$ by x_{ij}. Moreover, the $i-$th element of vector $\boldsymbol{\theta}$ is denoted by θ_i. An arithmetic index in parenthesis, i.e., (k), $k = 0, 1, \dots$, is reserved to declare an iterative (algorithmic) process, e.g., on matrix $\boldsymbol{X}$ and vector $\boldsymbol{r}$ the iteratively generated matrix and vector are denoted by $\boldsymbol{X}_{(k)}$ and $\boldsymbol{r}_{(k)}$, respectively. Following this rationale, $r_{(k),i}$ is reserved for the $i-$th element of the iteratively generated vector $\boldsymbol{r}_{(k)}$. The notation $\boldsymbol{X}_{\mathcal{S}}$ denotes the matrix $\boldsymbol{X}$ restricted over the set $\mathcal{S}$, i.e., the matrix that comprises the columns of $\boldsymbol{X}$, whose indices belong to the ordered index set $\mathcal{S} = \{j_1 < \dots < j_s\}$. Accordingly, the notation $\boldsymbol{u}_{\mathcal{S}}$ denotes the elements of vector $\boldsymbol{u}$, restricted over the set $\mathcal{S} \subseteq \mathrm{supp}(\boldsymbol{u})$. Finally, the identity matrix of dimension N will be denoted as $\boldsymbol{I}_N$ where $\boldsymbol{e}_j$ is its $j-$th column vector, the zero matrix of dimension $N \times N$, as $\boldsymbol{O}_N$, the vector of zero elements of appropriate dimension as $\boldsymbol{0}$ and the columns of matrix $\boldsymbol{I}_N$ restricted over the set $\mathcal{S}$, as $\boldsymbol{I}_{\mathcal{S}}$.

II. Preliminaries

In this section, an overview of some of the basic properties of the RKHS is provided [1], [21]–[25]. An RKHS is a Hilbert space $\mathcal{H}$ over a field $\mathbb{F}$ for which there exists a positive definite function, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{F}$, such that for every $\boldsymbol{x} \in \mathcal{X}$, $\kappa(\cdot, \boldsymbol{x})$ belongs to $\mathcal{H}$ and $f(\boldsymbol{x}) = \langle f, \kappa(\cdot, \boldsymbol{x})\rangle_{\mathcal{H}}$, for all $f \in \mathcal{H}$; in particular, $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \langle \kappa(\cdot, \boldsymbol{y}), \kappa(\cdot, \boldsymbol{x})\rangle_{\mathcal{H}}$. The *Gram* matrix $\boldsymbol{K}$, corresponding to the kernel κ, i.e., the matrix with elements $\kappa_{ij} := \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, is positive definite for any selection of finite number of points $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$, $N \in \mathbb{N}^*$. Moreover, the fundamental Representer Theorem establishes that although an RKHS may have infinite dimension, the solution of any regularized regression optimization task lies in the span of N specific kernels, e.g., [1], [21]. In other words, each minimizer $f \in \mathcal{H}$ admits a representation of the form $f = \sum_{j=1}^N \alpha_j \kappa(\cdot, \boldsymbol{x}_j)$. However, in many applications (also within this approach) a *bias* term, c, is often included in the aforementioned expansion; i.e., we assume that f admits the following representation:

$$f = \sum_{j=1}^{N} \alpha_j \kappa(\cdot, \boldsymbol{x}_j) + c. \tag{2}$$

The use of the bias term is theoretically justified by the Semi-parametric Representer Theorem, e.g., [1], [21].

Although there are many kernels to choose from, in this manuscript the experiments are focused on the real *Gaussian radial basis function* (RBF), i.e., $\kappa_\sigma(\boldsymbol{x}, \boldsymbol{x}') := \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/\sigma^2\right)$, defined for $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^M$, where σ, is a free positive parameter that defines the shape of the kernel function. In the following, κ is adopted to denote the Gaussian RBF. An important property of this kernel is that the corresponding matrix, $\boldsymbol{K}$, given by $\kappa_{ij} := exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma^2})$, has full rank. The significance of the theorem is that the points $\kappa(\cdot, \boldsymbol{x}_1), \kappa(\cdot, \boldsymbol{x}_2), ..., \kappa(\cdot, \boldsymbol{x}_N) \in \mathcal{H}$ are linearly independent, i.e., span the N-dimensional subspace of $\mathcal{H}$, [21].

III. Problem Formulation and related works

A. Robust Ridge Regression in RKHS

Given the data set $\mathcal{D} = \{(y_i, \boldsymbol{x}_i)\}_{i=1}^N$, we assume that each observation y_i is related to the corresponding input vector, $\boldsymbol{x}_i$, via

$$y_i = \underline{f}(\boldsymbol{x}_i) + \underline{u}_i + \eta_i, \; i = 1, \dots, N, \tag{3}$$

where $\underline{f} \in \mathcal{H}$ and $\mathcal{H}$ is a specific RKHS. The variable $\underline{u}_i$ represents a possible outlier sample and η_i a noise component. In a more compact form, this can be cast as $\boldsymbol{y} = \underline{\boldsymbol{f}} + \underline{\boldsymbol{u}} + \boldsymbol{\eta}$, where $\underline{\boldsymbol{f}}$ is the vector containing the values $\underline{f}(\boldsymbol{x}_i)$ for all $i = 1, \dots, N$. As $\underline{\boldsymbol{u}}$ represents the vector of the (unknown) outliers, it is reasonable to assume that this is a sparse vector. Our goal is to estimate the input-output relation $\underline{f}$ from the noisy observations of the data set $\mathcal{D}$. This can be interpreted as the task of simultaneously estimating both a sparse vector $\boldsymbol{u}$ and as well as a function $f \in \mathcal{H}$, that maintains a low squared error for $L(\mathcal{D}, f, \boldsymbol{u}) = \sum_{i=1}^N (y_i - f(\boldsymbol{x}_i) - u_i)^2$. Moreover, motivated by the representer theorem, we adopt the representation in (2), as a means to represent the solution for f. Under these assumptions, equation (3) could be expressed in a compact form as

$$\boldsymbol{y} = \boldsymbol{K}\underline{\boldsymbol{\alpha}} + \underline{c}\boldsymbol{1} + \underline{\boldsymbol{u}} + \boldsymbol{\eta} = \boldsymbol{X}_{(0)} \begin{pmatrix} \underline{\boldsymbol{\alpha}} \\ \underline{c} \end{pmatrix} + \boldsymbol{v}, \tag{4}$$

where $\boldsymbol{K}$ is the kernel matrix, $\boldsymbol{X}_{(0)} = [\boldsymbol{K} \;\; \boldsymbol{1}]$ ($\boldsymbol{1}$ is the vector of ones) and $\boldsymbol{v} = \underline{\boldsymbol{u}} + \boldsymbol{\eta}$ is the total noise vector (outlier plus inlier). Accordingly, the squared error is written as $L(\mathcal{D}, \boldsymbol{\alpha}, c, \boldsymbol{u}) = \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha} - c\boldsymbol{1} - \boldsymbol{u}\|_2^2$, and the respective minimization task can be cast as:

$$\begin{aligned} \min_{\boldsymbol{u}, \boldsymbol{\alpha} \in \mathbb{R}^N, c \in \mathbb{R}} \quad & \|\boldsymbol{u}\|_0 \\ \text{s. t.} \quad & \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha} - c\boldsymbol{1} - \boldsymbol{u}\|_2^2 + \lambda \left\| \begin{pmatrix} \boldsymbol{\alpha} \\ c \end{pmatrix} \right\|_2^2 \le \varepsilon, \end{aligned} \tag{5}$$

[1]This matrix has the vector's coefficients on its *diagonal*, while all other entries are equal to zero.

for some predefined parameters $\varepsilon, \lambda > 0$, where we have also used a standard regularization technique in order to keep the norm of the vector for the kernel expansion coefficients low. An alternative regularization strategy, which is common in the respective literature (based on KRR), is to include the norm of f, i.e., $\|f\|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha}$, instead of the norm of the coefficients' vector, leading to the following task:

$$\begin{aligned} \min_{\boldsymbol{u},\boldsymbol{\alpha}\in\mathbb{R}^N, c\in\mathbb{R}} \quad & \|\boldsymbol{u}\|_0 \\ \text{s. t.} \quad & \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha} - c\mathbf{1} - \boldsymbol{u}\|_2^2 + \lambda \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} \leq \varepsilon. \end{aligned} \tag{6}$$

B. Related Works

Two methods that deal with the RKRR task have already been established. The first method is based on the minimization of the ℓ_1 norm for the sparse outlier vector (instead of the ℓ_0 norm) and the second one employs sparse Bayesian learning arguments.

RAM: Refined Alternating Directions Method of Multipliers
The method is based on the ℓ_1-norm minimization for the sparse outlier vector (similar to the formulation in (6)). Next, by expressing the task in the LASSO form, the authors have established the so-called AM solver, which deals with the task via the alternating directions method of multipliers (ADMM). Furthermore, by using this solution as an initialization they have improved the scheme via the use of the reweighted ℓ_1-norm technique, as proposed in [26]. The resulting method is called RAM and stands for refined AM solver. More details over the scheme can be found in [14].

RB-RVM: Robust Relevance Vector Machine - Sparse Bayesian Learning
The Sparse Bayesian learning scheme is based on the RVM rationale and it employees hyper-parameters in order to infer not only the unknown kernel coefficients but also the sparse outlier estimate. More details on this approach can be found in [15], [27].

IV. Kernel Greedy Algorithm for Robust Denoising (KGARD)

A. Motivation and Proposed Scheme

Our proposed scheme alternates between a regularized Least Squares step and an OMP selection step based on the residual. At this point, it must be pointed out that that raw residuals can fail to detect outliers at leverage points; this is also known as swamping and masking of the outliers, [2]. It is well known that this is related to the input data. In our particular non-linear regularized setting, this occurs when the diagonal elements of the matrix $\boldsymbol{H} = \boldsymbol{X}_{(0)}(\boldsymbol{X}_{(0)}^T \boldsymbol{X}_{(0)})^{-1} \boldsymbol{X}_{(0)}^T$ get values close to one. As a result (following the assumption in [2]), it is reasonable to assume that $\max_{1\leq i\leq N} h_{ii} = h << 1$; in other words and taking also into account that the outlier vector is sparse, we assume that the outliers can be successfully detected via the residual of the Least Squares step. For more details, read [2] and [28].

In the following, we build upon the two formulations (5) and (6), that attempt (and indeed succeed) to solve the robust

Algorithm 1 Kernel Greedy Algorithm for Robust Denoising: KGARD

```
1: procedure KGARD($\boldsymbol{K}, \boldsymbol{y}, \lambda, \epsilon$)
2:     $k \leftarrow 0$
3:     $\mathcal{S}_0 \leftarrow \{1, 2, ..., N+1\}, \mathcal{S}_0^c \leftarrow \{N+2, ..., 2N+1\}$
4:     $\hat{\boldsymbol{z}}_{(0)} \leftarrow \left(\boldsymbol{X}_{\mathcal{S}_0}^T \boldsymbol{X}_{\mathcal{S}_0} + \lambda \boldsymbol{B}_{\mathcal{S}_0}\right)^{-1} \boldsymbol{X}_{\mathcal{S}_0}^T \boldsymbol{y}$
5:     $\boldsymbol{r}_{(0)} \leftarrow \boldsymbol{y} - \boldsymbol{X}_{\mathcal{S}_0} \hat{\boldsymbol{z}}_{(0)}$
6:     while $\|\boldsymbol{r}_{(k-1)}\|_2 > \epsilon$ do
7:         $k \leftarrow k+1$
8:         $j_k \leftarrow \arg\max_{j\in\tilde{\mathcal{S}}_k^c} |r_{(k-1),j}|$          $\triangleright$ ($\tilde{\mathcal{S}}_k^c$ in (11)).
9:         $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{j_k + N + 1\}, \mathcal{S}_k^c \leftarrow \mathcal{S}_k^c - \{j_k + N + 1\}$
10:        $\hat{\boldsymbol{z}}_{(k)} \leftarrow \left(\boldsymbol{X}_{\mathcal{S}_k}^T \boldsymbol{X}_{\mathcal{S}_k} + \lambda \boldsymbol{B}_{S_k}\right)^{-1} \boldsymbol{X}_{\mathcal{S}_k}^T \boldsymbol{y}$
11:        $\boldsymbol{r}_{(k)} \leftarrow \boldsymbol{y} - \boldsymbol{X}_{\mathcal{S}_k} \hat{\boldsymbol{z}}_{(k)}$
12:    Output: $\hat{\boldsymbol{z}}_{(k)} = \left(\hat{\boldsymbol{\alpha}}_{(k)}^T, \hat{c}_{(k)}, \hat{\boldsymbol{u}}_{(k)}^T\right)^T$ after $k$ iterations.
```

Least Squares task. Obviously, the difference lies solely on the regularization term. In the first approach, the regularization is performed using the ℓ_2-norm of the unkown kernel parameters (which is a standard regularization technique in linear methods). In contrast, in the alternative formulation we perform the regularization via the $\mathcal{H}$-norm of f. The reason for this modification was the improved performance obtained in practice via the first approach.

Since both tasks in (5) and (6) are known to be NP-hard, a straight-forward computation of a solution seems impossible. However, under certain assumptions, greedy-based techniques often manage to provide accurate solutions to ℓ_0-norm minimization tasks, which are also guaranteed to be close to the optimal solution. The proposed KGARD algorithm, which is based on a modification of the popular Orthogonal Matching Pursuit (OMP), has been adapted to both formulations, i.e., (5) and (6).

First, one should notice that, the quadratic inequality constraint could also be written in a more compact form as follows:

$$J(\boldsymbol{z}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{z}\|_2^2 + \lambda \boldsymbol{z}^T \boldsymbol{B} \boldsymbol{z} \leq \varepsilon, \tag{7}$$

where $$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{K} & \mathbf{1} & \boldsymbol{I}_N \end{bmatrix}, \ \boldsymbol{z} = (\boldsymbol{\alpha}^T, c, \boldsymbol{u}^T)^T, \tag{8}$$

and for the choice of matrix $\boldsymbol{B}$ either one of the following matrices can be used,

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{I}_N & \mathbf{0} & \boldsymbol{O}_N \\ \mathbf{0}^T & 1 & \mathbf{0}^T \\ \boldsymbol{O}_N & \mathbf{0} & \boldsymbol{O}_N \end{bmatrix} \text{ or } \begin{bmatrix} \boldsymbol{K} & \mathbf{0} & \boldsymbol{O}_N \\ \mathbf{0}^T & 0 & \mathbf{0}^T \\ \boldsymbol{O}_N & \mathbf{0} & \boldsymbol{O}_N \end{bmatrix}, \tag{9}$$

depending on whether (5) or (6) is adopted, respectively.

The proposed method, as presented in Algorithm 1, attempts to solve the task (5) or (6), via a sparse greedy-based approach. The algorithm alternates between an LS task and a column selection step, that enlarges the solution subspace at each step, in order to minimize the residual error. The scheme shares resemblances to the OMP algorithm. Its main differences, are: (a) the solution of a regularized LS task at each iteration (instead of a simple LS task), i.e.,

$$\min_{\boldsymbol{z}} J_k(\boldsymbol{z}) = \min_{\boldsymbol{z}} \left\{ \|\boldsymbol{y} - \boldsymbol{X}_{\mathcal{S}_k} \boldsymbol{z}\|_2^2 + \lambda \boldsymbol{z}^T \boldsymbol{B}_{\mathcal{S}_k} \boldsymbol{z} \right\}, \tag{10}$$

and (b) the use of a specific initialization on the solution and the residual. These seemingly small differences lead to a completely distinct performance analysis for the method as compared to the standard OMP. The scheme is specified best, via the use of subsets, corresponding to a set of *active* and *inactive columns*, for any given matrix. In particular, the $2N+1$ column vectors of the matrices $\boldsymbol{X}$ and $\boldsymbol{B}$ are divided into two complementary subsets: the active set, $\mathcal{S}_k$, which contains the indices of the active columns of the matrix at step k, and the inactive set, $\mathcal{S}_k^c$, which contains the remaining ones; i.e., those that do not participate in the expansion. Thus, $\boldsymbol{X}_{\mathcal{S}_k}$ and $\boldsymbol{B}_{\mathcal{S}_k}$ denote the column vectors of matrices $\boldsymbol{X}$ and $\boldsymbol{B}$, respectively, restricted over the subset $\mathcal{S}_k$. Moreover, we define the set of indices

$$\tilde{\mathcal{S}}_k^c := \{i - N - 1 |\ i \in \mathcal{S}_k^c\}, \tag{11}$$

which is very helpfull for the description of the proposed method. While the set $\mathcal{S}_k^c$ refers to the columns of the augmented matrix $\boldsymbol{X}$, the set $\tilde{\mathcal{S}}_k^c$ refers to the columns of the identity matrix (the last part of matrix $\boldsymbol{X}$), i.e., matrix $\boldsymbol{I}_N$. In other words, $\tilde{\mathcal{S}}_k^c$ originates by subtracting the value $N+1$ from each one of the elements of $\mathcal{S}_k^c$. Initially, only the first $N+1$ columns of matrices $\boldsymbol{X}$ and $\boldsymbol{B}$, have been activated. Thus, $k=0$, leads to the initialization of the active set $\mathcal{S}_0 = \{1, 2, \dots, N+1\}$ with the corresponding matrices:

$$\boldsymbol{X}_{\mathcal{S}_0} = [\boldsymbol{K}\ \boldsymbol{1}],$$

and
$$\boldsymbol{B}_{\mathcal{S}_0} = \boldsymbol{I}_{N+1} \text{ or } \begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \\ \boldsymbol{0}^T & 0 \end{bmatrix},$$

depending on the model selection, i.e., (5) or (6) respectively. Hence, the solution to the initial LS problem, is given by

$$\hat{\boldsymbol{z}}_{(0)} := \arg\min_{\boldsymbol{z}}\{J_0(\boldsymbol{z})\} = \left(\boldsymbol{X}_{\mathcal{S}_0}^T \boldsymbol{X}_{\mathcal{S}_0} + \lambda \boldsymbol{B}_{\mathcal{S}_0}\right)^{-1} \boldsymbol{X}_{\mathcal{S}_0}^T \boldsymbol{y}.$$

Next, the method computes the residual $\boldsymbol{r}_{(0)} = \boldsymbol{y} - \boldsymbol{X}_{\mathcal{S}_0}\hat{\boldsymbol{z}}_{(0)}$ and identifies an outlier[2], as the largest value of the residual vector. The corresponding index, say $j_1 \in \tilde{\mathcal{S}}_k^c$, is added into the set of active columns, i.e., $\mathcal{S}_1 = \mathcal{S}_0 \cup \{j_k + N + 1\}$. Thus, the matrix $\boldsymbol{X}_{\mathcal{S}_0}$ is augmented by a column drawn from matrix $\boldsymbol{I}_N$, forming matrix $\boldsymbol{X}_{\mathcal{S}_1}$. Accordingly, the matrix $\boldsymbol{B}_{\mathcal{S}_0}$ is augmented by a zero row and a zero column, forming $\boldsymbol{B}_{\mathcal{S}_1}$. The new LS task is solved again (using matrices $\boldsymbol{X}_{\mathcal{S}_1}$, $\boldsymbol{B}_{\mathcal{S}_1}$) and a new residual $\boldsymbol{r}_{(1)}$ is computed. The process is repeated, until the residual drops below a predefined threshold.

Although, both approaches, (5) and (6), are suitable for dealing with the sparse minimization task, in practise the selection of (5) proves a better choice. To this end, in the future, the model (5) is adopted and thus the respective $\boldsymbol{B}$ matrix is used.

Remark 1. *In order to simplify the notation, in the next sections, we adopt $\boldsymbol{X}_{(k)}$ and $\boldsymbol{B}_{(k)}$ to refer to the matrices $\boldsymbol{X}_{\mathcal{S}_k}$ and $\boldsymbol{B}_{\mathcal{S}_k}$ at the k step.*

Remark 2. *Once a column has been selected at the k step, it cannot be selected again in any subsequent step, since the corresponding residual coordinate is zero. In other words, the algorithm always selects a column from the last part of $\boldsymbol{X}$, i.e., matrix $\boldsymbol{I}_N$, that is not included in $\mathcal{S}_k$.*

[2]If outliers are not present, the algorithm terminates and no outlier estimate exists in the solution $\hat{\boldsymbol{z}}_0$.

B. Efficient Implementations

As the outliers often comprise a small fraction of the data set, i.e., $k << N$, a fast implementation time for OMP-like schemes such as KGARD is expected. Initially, the inversion of matrix $\boldsymbol{X}_{(0)}^T\boldsymbol{X}_{(0)} + \lambda\boldsymbol{B}_{(0)}$ plus the multiplication of $\boldsymbol{X}_{(0)}^T\boldsymbol{y}$, requires $\mathrm{O}\left((N+1)^3\right)$ flops. At each one of the subsequent steps, the required complexity is $\mathrm{O}\left((N+k+1)^3\right)$, while the total cost for the method is $\mathrm{O}\left((N+1)^3(k+1) + (5/2)N^2k^2 + (4/3)Nk^3 + k^4/4\right)$, which is acceptable, since $k << N$ is assumed. However, the complexity of the method could be further reduced, since a large part of the inverted matrix remains unchanged. To this end, several methods could be employed, [29].

The first technique, which has been applied to the proposed scheme, is the *matrix inversion lemma* (MIL), which reduces the cost from cubic to square. However, an alternative technique which is even more efficient and is used throughout this paper is the *Cholesky decomposition* for the matrix to be inverted. This is summarized in the following steps:

- *Replace* step 4 of algorithm 1, with:
  Factorization step: $\boldsymbol{M}_{(0)} = \boldsymbol{L}_{(0)}\boldsymbol{L}_{(0)}^T$

  Solve $\boldsymbol{L}_{(0)}\boldsymbol{L}_{(0)}^T\hat{\boldsymbol{z}}_{(0)} = \boldsymbol{X}_{(0)}^T\boldsymbol{y}$ using:
  – forward substitution $\boldsymbol{L}_{(0)}\boldsymbol{q} = \boldsymbol{X}_{(0)}^T\boldsymbol{y}$
  – backward substitution $\boldsymbol{L}_{(0)}^T\hat{\boldsymbol{z}}_{(0)} = \boldsymbol{q}$
  Complexity: $\mathrm{O}\left((N+1)^3/3 + (N+1)^2\right)$
- *Replace* step 10 of algorithm 1, with:
  Compute $\boldsymbol{d}$ such that: $\boldsymbol{L}_{(k-1)}\boldsymbol{d} = \boldsymbol{X}_{(k-1)}^T\boldsymbol{e}_{j_k}$
  Compute: $b = \sqrt{1 - \|\boldsymbol{d}\|_2^2}$
  Matrix Update: $\boldsymbol{L}_{(k)} = \begin{bmatrix} \boldsymbol{L}_{(k-1)} & \boldsymbol{0} \\ \boldsymbol{d}^T & b \end{bmatrix}$
  Solve $\boldsymbol{L}_{(k)}\boldsymbol{L}_{(k)}^T\hat{\boldsymbol{z}}_{(k)} = \boldsymbol{X}_{(k)}^T\boldsymbol{y}$ using:
  – forward substitution $\boldsymbol{L}_{(k)}\boldsymbol{p} = \boldsymbol{X}_{(k)}^T\boldsymbol{y}$
  – backward substitution $\boldsymbol{L}_{(k)}^T\hat{\boldsymbol{z}}_{(k)} = \boldsymbol{p}$
  Complexity: $\mathrm{O}\left((9/2)N^2 + 5Nk + (3/2)k^2\right)$ per iteration.

Employing the Cholseky decomposition plus the update step leads to a reduction of the total computational cost to $\mathrm{O}\left((N+1)^3/3 + (N+1)^2 + k^3/2 + (5/2)Nk^2\right)$, which is the fastest implementation for this task (recall that $k << N$).

C. Further Improvements on KGARD's Performance

In order to simplify the theoretical analysis and reduce the corresponding equations, the proposed algorithm employs the same regularization parameter for all kernel coefficients. However, one may employ a more general scheme as follows:

$$\begin{aligned} \min_{\boldsymbol{u},\boldsymbol{a}\in\mathbb{R}^N, c\in\mathbb{R}} \quad & \|\boldsymbol{u}\|_0 \\ \text{s. t.} \quad & \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{a} - c\boldsymbol{1} - \boldsymbol{u}\|_2^2 + \|\boldsymbol{\Psi}\boldsymbol{a}\|_2^2 + \lambda c^2 \le \varepsilon, \end{aligned}$$

where $\boldsymbol{\Psi}$ is a more general regularization matrix (Tikhonov matrix). For example, as the accuracy of kernel based methods usually drops near the border of the input domain, it is reasonable to increase the regularization effect at these points. This can be easily implemented by employing a diagonal matrix with positive elements on the diagonal (that correspond to λ) and increase the regularization terms that correspond to the points near the border. This is demonstrated in the experimental section VI.

V. Theoretical Analysis

In the current section, the theoretical properties of the proposed robust kernel regression method, i.e., KGARD, are analysed. First, we establish that the method always converges in finite time and that the reconstruction error of the method is *strictly decreasing*. Next, we provide the necessary conditions so that the proposed method succeeds in identifying first the locations of all the outliers, for the case where only outliers exist in the noise. The derived theoretical condition for the second part (i.e., the outlier identification) is rather tight. However, as demonstrated in the experiments, the method achieves to recover the correct support of the sparse outlier vector in many cases where the theoretical result doesn't hold. This leads to the conclusion that the provided conditions can be loosen up significantly in the future. Moreover, in practice, where inlier noise also exists, the method succeeds to correctly identify the majority of the outliers. The reason that, the analysis is carried out for the case where inlier noise is not present, is due to the fact that the analysis gets highly involved. The absence of the inlier noise makes the analysis easier and it highlights some theoretical aspects on why the method works. It must be emphasized that, such a theoretical analysis is carried out for the first time and is absent in the previously published works.

A. Convergence Analysis

Regarding the convergence of the algorithm, it is easy to check that the proposed algorithm will always converge in finite time. Indeed, assuming the worst case scenario, where the algorithm continues until all columns of $\boldsymbol{I}_N$ are selected, we can easily see that the norm of the residual vector will eventually drop below ϵ. Of course, this is something that occurs in the case where the parameter ϵ is set extremely low. As a consequence, the procedure will continue and model all noise samples (even those originating from an inlier source) as impulses, filling up the vector $\boldsymbol{u}$ and producing a residual vector equal to $\boldsymbol{0}$. Obviously, if ϵ is carefully tuned and the outliers are sufficiently sparse, the algorithm will stop well before that. Hence, a sensible tuning of ϵ should be applied.

Moreover, note that, for all $\varepsilon \geq 0$, there exists $\boldsymbol{z}$ such that $J_k(\boldsymbol{z}) \leq \varepsilon$. This implies that the feasible set of (5) is always nonempty[3]. It is straightforward to prove that the set of *normal equations*, obtained from (10), at step k, is

$$(\boldsymbol{X}_{(k)}^T \boldsymbol{X}_{(k)} + \lambda \boldsymbol{B}_{(k)})\boldsymbol{z} = \boldsymbol{X}_{(k)}^T \boldsymbol{y}, \tag{12}$$

where $(\boldsymbol{X}_{(k)}^T \boldsymbol{X}_{(k)} + \lambda \boldsymbol{B}_{(k)})$ is invertible, i.e., (10) has a unique minimum, for all k. Recall that the matrix on the left side in (12) is (strictly) positive definite, hence invertible.

Alternatively, one could express (10) as follows[4]:

$$\min_{\boldsymbol{z}} \quad J_k(\boldsymbol{z}) = \left\| \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{pmatrix} - \boldsymbol{D}_{(k)}\boldsymbol{z} \right\|_2^2, \tag{13}$$

where $\boldsymbol{D}_{(k)} = \begin{bmatrix} \boldsymbol{X}_{(k)} \\ \sqrt{\lambda}\boldsymbol{B}_{(k)} \end{bmatrix}$. Problem (13) has a unique solution, if and only if the nullspaces of $\boldsymbol{X}_{(k)}$ and $\boldsymbol{B}_{(k)}$ intersect only trivially, i.e., $\mathcal{N}(\boldsymbol{X}_{(k)}) \cap \mathcal{N}(\boldsymbol{B}_{(k)}) = \{0\}$ [30], [31]. Hence, $\boldsymbol{M}_{(k)} = \boldsymbol{D}_{(k)}^T \boldsymbol{D}_{(k)}$ is (strictly) positive definite, as the columns of $\boldsymbol{D}_{(k)}$ are linearly independent and the minimizer $\boldsymbol{z}_* \in \mathbb{R}^{N+1+k}$ of (10) is unique, [32]. Furthermore, similarly to the discussion in Section III, an equivalent formulation for (10) is

$$\min_{\boldsymbol{z}} ||\boldsymbol{y} - \boldsymbol{X}_{(k)}\boldsymbol{z}||_2^2, \text{ s.t. } ||\boldsymbol{B}_{(k)}\boldsymbol{z}||_2 \leq \delta, \tag{14}$$

for some $\delta > 0$. In (14), the regularization term is replaced by a quadratic constraint. Equivalence between (10) and (14) has been well studied and established [33]. The reason for resorting to the latter formulation is to be used for the proof of the following lemma.

Lemma 1. *The norm of the residual for KGARD is strictly decreasing.*

Proof. Recall that during initialization, KGARD sets $\mathcal{S}_0$ to include only the first $N+1$ columns of matrices $\boldsymbol{X}$, $\boldsymbol{B}$ and let $\hat{\boldsymbol{z}}_{(0)}$ denote the initial solution of (14) and $\boldsymbol{r}_{(0)} = \boldsymbol{y} - \boldsymbol{X}_{(0)}\hat{\boldsymbol{z}}_{(0)}$ be the initial residual. Since our goal is to remove the unknown/unwanted additive noise, it is expected that $\boldsymbol{y} \notin \mathcal{R}(\boldsymbol{X}_{(0)})$ (the range of the matrix), regardless of the statistics of the additive noise (Gaussian, impulse or both). Suppose, now, that the ℓ_2 norm of the residual $\boldsymbol{r}_{(0)}$ is below our threshold parameter ϵ. In this case, the method is forced to stop; either no outlying values are identified or the threshold parameter is tuned extremely high. Nevertheless, the case of greater importance, is when outliers are present; the algorithm continues expanding the set of active columns with columns from the identity matrix and thus a sparse outlier vector is generated.

At each subsequent iteration, k, the algorithm selects an index from the set $\mathcal{S}_{k-1}^c$ of *inactive* columns from matrix $\boldsymbol{X}$. Then, $\mathcal{S}_{k-1}$ is enlarged by the selected index (say j_k) and the matrix $\boldsymbol{X}_{(k-1)}$ is augmented by the column vector $\boldsymbol{e}_{j_k}$ forming $\mathcal{S}_k$ and $\boldsymbol{X}_{(k)}$, respectively. Finally, the solution $\hat{\boldsymbol{z}}_{(k)} \in \mathbb{R}^{N+k+1}$ and the residual $\boldsymbol{r}_{(k)} = \boldsymbol{y} - \boldsymbol{X}_{(k)}\hat{\boldsymbol{z}}_{(k)}$ are computed, respectively, by solving (14) (step 10 of the algorithm). At $k+1$ step, the process is repeated and the matrices are augmented. At this stage we have, $\boldsymbol{X}_{(k+1)} = [\boldsymbol{X}_{(0)}\ \boldsymbol{e}_{j_1}\ \cdots\ \boldsymbol{e}_{j_k}\ \boldsymbol{e}_{j_{k+1}}] = [\boldsymbol{X}_{(k)}\ \boldsymbol{e}_{j_{k+1}}]$.

Now, let $\hat{\boldsymbol{z}}_{(k+1)} \in \mathbb{R}^{N+k+2}$ be the unique minimizer of $L_{k+1}(\boldsymbol{z}) = ||\boldsymbol{y} - \boldsymbol{X}_{(k+1)}\boldsymbol{z}||_2^2$ subject to the constraint $\|\boldsymbol{B}_{(k+1)}\boldsymbol{z}\|_2 \leq \delta$, i.e., the minimization of (14) at the $k+1$ step. Also, let $\mathsf{z}_{(k+1)} = (\hat{\boldsymbol{z}}_{(k)}^T, r_{(k),j_{k+1}})^T$. Observe that $\mathsf{z}_{(k+1)}$ belongs to the feasible set defined by the inequality constraint

[3]For example, if we select $\boldsymbol{z} = \left(\boldsymbol{0}^T, 0, \boldsymbol{y}^T\right)^T$, then $J_k(\boldsymbol{z}) = 0$.

[4]Notice that $\boldsymbol{B}$ is a projection matrix (this holds only for the regularization performed with the ℓ_2 norm).

of (14) at the current step[5], and hence $L_{k+1}(\hat{\boldsymbol{z}}_{(k+1)}) \leq L_{k+1}(\mathsf{z}_{(k+1)})$. Moreover, we have that

$$\begin{aligned} L_{k+1}(\mathsf{z}_{(k+1)}) &= \left\| \boldsymbol{y} - \begin{bmatrix} \boldsymbol{X}_{(k)} & \boldsymbol{e}_{j_{k+1}} \end{bmatrix} \cdot \begin{pmatrix} \hat{\boldsymbol{z}}_{(k)} \\ r_{(k),j_{k+1}} \end{pmatrix} \right\|_2^2 \\ &= \left\| \boldsymbol{y} - \boldsymbol{X}_{(k)}\hat{\boldsymbol{z}}_{(k)} - r_{(k),j_{k+1}}\boldsymbol{e}_{j_{k+1}} \right\|_2^2 \\ &= \left\| \boldsymbol{r}_{(k)} - r_{(k),j_{k+1}}\boldsymbol{e}_{j_{k+1}} \right\|_2^2 < \left\| \boldsymbol{r}_{(k)} \right\|_2^2, \quad (15) \end{aligned}$$

where the last strict inequality is due to the fact that $|r_{(k),j_{k+1}}| > 0$ (if $r_{(k),j_{k+1}} = 0$, then $\boldsymbol{r}_{(k)}$ is a zero vector, since its maximum value is 0 and the algorithm should have been terminated at iteration k). Thus, we conclude that $\|\boldsymbol{r}_{(k+1)}\|_2^2 = L_{k+1}(\hat{\boldsymbol{z}}_{(k+1)}) \leq L_{k+1}(\mathsf{z}_{(k+1)}) < \|\boldsymbol{r}_{(k)}\|_2^2$. □

B. Identification of the Outliers for the Noiseless Case

The following theorem establishes a bound on the largest singular value of matrix $\boldsymbol{X}_{(0)}$, which guarantees that the method first identifies the correct locations of all the outliers, for the case where only outliers exist in the noise. However, since the ϵ parameter controls the number of iterations, for which the method identifies an outlier, it is not guaranteed that it will stop once all the outliers are identified, unless the correct value is somehow given. Thus, it is possible that a few other locations, that do not correspond to outliers, are also identified. It must be pointed out that, such a result has never been established before by other comparative methods.

Theorem 1. *Let $\boldsymbol{K}$ be a full rank, square, real valued matrix. Suppose, that*

$$\boldsymbol{y} = [\boldsymbol{K}\ \boldsymbol{1}]\underbrace{(\underline{\boldsymbol{\alpha}}^T, \underline{c})^T}_{\underline{\boldsymbol{\theta}}} + \underline{\boldsymbol{u}},$$

where $\underline{\boldsymbol{u}}$ is a sparse (outlier) vector. KGARD is guaranteed to identify first the correct locations of all the outliers, if the maximum singular value of matrix $\boldsymbol{X}_{(0)} := [\boldsymbol{K}\ \boldsymbol{1}]$, satisfies:

$$\sigma_M(\boldsymbol{X}_{(0)}) < \gamma\sqrt{\lambda}, \quad (16)$$

where $$\gamma = \sqrt{\frac{\min|\underline{u}| - \sqrt{2\lambda}||\underline{\boldsymbol{\theta}}||_2}{2||\underline{\boldsymbol{u}}||_2 - \min|\underline{u}| + \sqrt{2\lambda}||\underline{\boldsymbol{\theta}}||_2}}, \quad (17)$$

$\min|\underline{u}|$ is the smallest absolute value of the sparse vector over the non-zero coordinates and $\lambda > 0$ is a sufficiently large[6] regularization parameter for KGARD.

The proof is briefly presented in the Appendix section. For a more detailed proof see [28].

Remark 3. *The theorem does not guarantee that only the locations of the true outliers will be identified. If the value of ϵ is too small, then KGARD once it identifies the location of the true outliers, it will next identify locations that do not correspond to outlier indices.*

[5]Geometrically the feasible set remains the same, while matrix $\boldsymbol{B}$ is augmented by zero elements at each step.

[6]Since the regularization parameter is defined by the user, we assume that such a value can be achieved, so that the γ parameter makes sense. More details can be found in the proof at the appendix section.

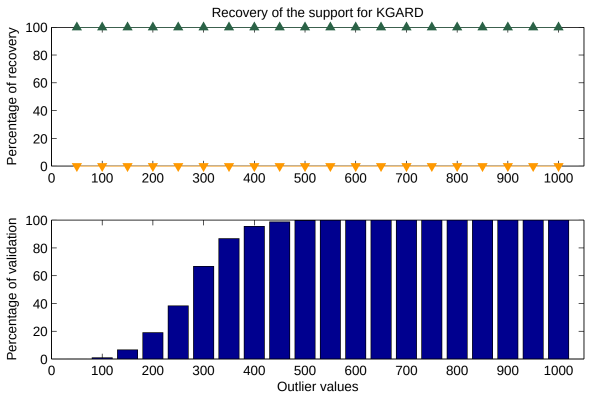

Fig. 1: Percentage of the correct (green pointing up) and wrong (orange pointing down) indices that KGARD has identified, while varying the values $\pm\underline{u}$ of the outliers at the fixed fraction of 10%. Although the condition (16) is valid only for values greater than ±600 (and with high probability valid for values 400-599), the support of the sparse outlier vector has been correctly identified for much smaller values of outlier noise, too.

Outlier fraction	Correct support	Wrong support	Outlier value u
5 %	100 %	0 %	450
10 %	100 %	0 %	600
15 %	100 %	0 %	650
20 %	100 %	0 %	700
25 %	100 %	0 %	750
30 %	100 %	0 %	950

TABLE I: Percentage of correct and wrong indices identified for all outlier values $\underline{u}$ ranging from 50 to 1000. The correct support corresponds to true outliers (indices in $\mathcal{T}$), while the wrong one corresponds to points which are wrongly classified as outliers (thus do not belong to $\mathcal{T}$). In the final column the minimum value $\underline{u}$ of outliers for which the support recovery condition is valid, is listed.

VI. Experiments

For the entire section of experiments, the Gaussian (RBF) kernel is employed and all the results are averaged over 1000 "Monte Carlo" runs (independent simulations). At each experiment, the parameters are optimized (via cross-validation) and the respective parameter values are given (for each method), so that results are reproducible. The specific MATLAB code can be found in http://bouboulis.mysch.gr/kernels.html.

A. Identification of the Outliers

In the current section, our main concern is to test on the validity of the condition (16) in practise. To this end, we have performed the following experiment, for the case where only outliers exist in the noise.

We consider $N = 100$ equidistant points over the interval $[0,1]$ and generate the output data via $\underline{f}(x_i) = \sum_{j=1}^N \underline{\alpha}_j \kappa(x_i, x_j)$, where κ is the Gaussian kernel with $\sigma = 0.1$ and the vector of coefficients $\underline{\boldsymbol{\alpha}} = [\underline{\alpha}_1, \ldots, \underline{\alpha}_N]$ is a sparse vector with the number of non-zero coordinates ranging between 2 and 23 and their values drawn from $\mathcal{N}(0, 0.5^2)$.

Method	MSE_{tr}	MSE_{val}	Cor. - Wr.	MIT	Noise
RB-RVM	0.0850	0.0851	-	0.298	20 dB - 5%
RAM $\lambda = 0.07, \mu = 2.5$	0.0344	0.0345	100% - 0.2%	0.005	20 dB - 5%
KGARD $\lambda = 0.2, \varepsilon = 10$	**0.0285**	**0.285**	100% - 0%	0.004	20 dB - 5%
RB-RVM	0.0911	0.0912	-	0.298	20 dB - 10%
RAM $\lambda = 0.07, \mu = 2.5$	0.0371	0.0372	100% - 0.1%	0.007	20 dB - 10%
KGARD $\lambda = 0.2, \varepsilon = 10$	**0.0305**	**0.0305**	100 % - 0 %	0.008	20 dB - 10%
RB-RVM	0.0992	0.0994	-	0.299	20 dB - 15%
RAM $\lambda = 0.07, \mu = 2$	0.0393	0.0393	100% - 0.6%	0.008	20 dB - 15%
KGARD $\lambda = 0.3, \varepsilon = 10$	**0.0330**	**0.0330**	100%- 0%	0.012	20 dB - 15%
RB-RVM	0.1189	0.1184	-	0.305	20 dB - 20%
RAM $\lambda = 0.07, \mu = 2$	**0.0421**	**0.0422**	100% - 0.4%	0.010	20 dB - 20%
KGARD $\lambda = 1, \varepsilon = 10$	0.0626	0.0626	100% - 0%	0.017	20 dB - 20%
RB-RVM	0.3630	0.3631	-	0.327	15 dB - 5%
RAM $\lambda = 0.15, \mu = 5$	0.1035	0.1036	100%- 0.7%	0.005	15 dB - 5%
KGARD $\lambda = 0.3, \varepsilon = 15$	**0.0862**	**0.0862**	100% - 0.1%	0.005	15 dB - 5%
RB-RVM	0.3828	0.3830	-	0.319	15 dB - 10%
RAM $\lambda = 0.15, \mu = 5$	0.1117	0.1118	100% - 0.4 %	0.006	15 dB - 10%
KGARD $\lambda = 0.3, \varepsilon = 15$	**0.0925**	**0.0925**	100% - 0%	0.008	15 dB - 10%
RB-RVM	0.4165	0.4166	-	0.317	15 dB - 15%
RAM $\lambda = 0.15, \mu = 5$	0.1186	0.1186	100% - 0.3%	0.007	15 dB - 15%
KGARD $\lambda = 0.3, \varepsilon = 15$	**0.1001**	**0.1003**	100% - 0%	0.012	15 dB - 15%
RB-RVM	0.4793	0.4798	-	0.312	15 dB - 20%
RAM $\lambda = 0.15, \mu = 4$	**0.1281**	**0.1282**	100% - 1.4 %	0.008	15 dB - 20%
KGARD $\lambda = 0.7, \varepsilon = 15$	0.1340	0.1349	100% - 0%	0.016	15 dB - 20%

TABLE II: Computed MSE for $\underline{f}(x) = 20sinc(2\pi x)$ over the training and validation set, percentage of correct and wrong support recovered and mean implementation time (MIT) in seconds, for each level of inlier noise and fraction of outliers.

Method	MSE_{tr}	MSE_{val}	Cor. - Wr. supp	MIT (sec)	Outliers
RB-RVM	3.9825	3.6918	-	0.416	5%
RAM $\lambda = 0.2, \mu = 22$	2.0534	1.8592	100% - 0.1 %	0.010	5%
KGARD $\lambda = 0.15, \varepsilon = 46$	**1.7381**	**1.5644**	100 % - 0.3 %	0.009	5%
RB-RVM	4.2382	3.8977	-	0.419	10%
RAM $\lambda = 0.2, \mu = 18$	2.2281	1.9926	100% - 0.9 %	0.013	10%
KGARD $\lambda = 0.15, \varepsilon = 44$	**1.8854**	**1.6750**	100 % - 0.5 %	0.016	10%
RB-RVM	4.5749	4.2181	-	0.418	15%
RAM $\lambda = 0.2, \mu = 17$	2.5944	2.2846	100% - 1.6 %	0.016	15%
KGARD $\lambda = 0.2, \varepsilon = 42$	**2.1968**	**1.9375**	99.9 % - 0.9 %	0.024	15%
RB-RVM	5.7051	5.0540	-	0.418	20%
RAM $\lambda = 0.2, \mu = 16$	3.0593	2.6703	99.9% - 2.3 %	0.020	20%
KGARD $\lambda = 0.4, \varepsilon = 42$	**3.0293**	**2.6113**	99.9 % - 1 %	0.033	20%

TABLE III: Performance evaluation for each method, for the case where the input data lies on the 2-dimensional space and the output $\underline{f} \in \mathcal{H}$ is considered as a linear combination of a few kernels. The inlier noise is considered random Gaussian with $\sigma = 3$ and for various fractions of outliers, the training and validation MSE, the percentage of correct support recovered and the mean implementation time (MIT), are listed.

Since no inlier noise exists, our corrupted data is given from (3) for $\eta_i = 0$ and outlier values $\pm\underline{u}$. Moreover, since the condition (16) is valid for fixed values of the parameters involved, we have measured the ability of KGARD to recover the support of the sparse outlier vector, i.e., $\mathcal{T} = \text{supp}(\underline{\boldsymbol{u}})$, while varying the values of the outliers. In Figure 1, the ability of KGARD to identify the exact sparse outlier vector support is demonstrated, for a fraction of outliers at 10%. On the vertical axis, we have measured the percentage of correct and wrong indices recovered, while varying the value u of the outliers. In parallel, the bar chart demonstrates the validity of the introduced condition (16). It is clear that, if the condition holds, KGARD identifies the correct support of the sparse outlier vector successfully. However, even if the condition is rarely satisfied, e.g., for $\underline{u} = 100$, the method still manages to identify the correct support. This fact leads to the conclusion that the condition imposed by (16) is rather strict. This is in line with most sparse modeling related conditions, which, in practice, fall short in predicting the exact recovery conditions.

Finally, in Table I, the previous experiment has been performed for various fractions of outliers. In the second and third column, we have listed the percentage of correct and wrong indices (truly) identified by the method, for all values of outliers ranging from 50 to 1000. Moreover, in the final column, the minimum value of outliers, which renders the condition valid, is shown. For example, in the second row and for 10% of outliers, the condition is valid only for values greater than 600 (last column of table I). However, the method manages to correctly identify the support (one-to-one index - columns two and three), not only for values $\underline{u}$ greater than 600, but for all outlier values, i.e, from the minimum value of 50 to the maximum value of 1000. It should also be noted that, experiments have been performed with the use of various non-linear functions (not only linear combinations of kernels) and results were similar to the ones presented here.

B. *Evaluation of the Method: Mean-Square-Error (MSE)*

In the current section, the previously established methods that deal with the non-linear robust estimation with kernels, i.e., the Bayesian approach RB-RVM and the weighted ℓ_1-norm approximation method (RAM), are compared against KGARD in terms of the mean-square-error (MSE) performance. Additionally, the evaluation is enhanced with a list of the percentage of the correct and wrong indices that each method has identified, for all methods except for the Bayesian approach (not directly provided by the RB-RVM method). Moreover, the *mean implementation time* (MIT) is measured for each experiment. Finally, following section IV-C, for the first experiment, we have increased the regularization value λ of KGARD near the edge points/borders, as a means to improve the performance. In particular, at the 5 first and 5 last points (borders), the regularizer is automatically multiplied by the factor of 5, with respect to the predefined value λ which is used on the interior points. The experiments are described in more detail next.

For the first experiment, we have selected the $sinc$ function, which is a popular one in machine learning. We consider 398 equidistant points over the interval $[-0.99, 1)$ for the input values and generated the uncorrupted output values via $\underline{f}(x_i) = 20sinc(2\pi x_i)$. Next, the set of points is split into two subsets, the training and the validation subset. The training subset, with points denoted by (y_i, x_i), consists of the $N = 199$ odd indexed points (first, third, e.t.c.), while the validation subset comprises the remaining points (denoted as (y_i', x_i')). The original data of the training set, is then contaminated by noise, as (3) suggests. The inlier part is considered to be random Gaussian noise of appropriate variance (measured in dB), while the outlier part consists of various fractions of outliers, with constant values ± 15, distributed uniformly over the support set. Finally, the kernel parameter σ has been set equal to $\sigma = 0.15$. Table II depicts the performance of each method, where the best results are marked in **bold**. In terms of the computed MSE, it is clear that KGARD attains a lower MSE for both the training and the validation error for all fractions of outliers, except for the fraction of 20%. This fact is also in line with the theoretical properties of the sparse greedy methods, since their performance boosts as the sparsity level of the approximation is low. On the other hand, the RAM solver seems more suitable for larger fractions of outliers. Moreover, the computational cost is comparable for both methods (RAM and KGARD), for small fractions of outliers. Regarding the identification of the sparse outlier vector support, although both methods correctly identify the indices that belong to the sparse outlier vector's support, i.e., $\mathcal{T} = \text{supp}(\underline{\boldsymbol{u}})$, RAM (wrongly) identifies more indices as outliers than KGARD.

For the second pilot experiment, KGARD's performance is tested for the case where the input data lies on a two-dimensional subspace. To this end, we consider 31 points in $[0,1]$ and separate these points, to form the training set, which comprises 16 odd indices and the rest 15, forming the validation set. Next, the 31^2 points are distributed over a squared lattice in plane $[0,1] \times [0,1]$, where each uncorrupted measurement is generated by $\underline{f}(\boldsymbol{x}_i) = \sum_{j=1}^{31^2} \underline{\alpha}_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, ($\sigma = 0.2$) and a sparse coefficient vector $\underline{\boldsymbol{\alpha}} = [\underline{\alpha}_1, \ldots, \underline{\alpha}_{31}]$ with non-zero values ranging between $4\% - 17.5\%$ and their values randomly drawn from $\mathcal{N}(0, 25.6^2)$. Thus, the training subset, consists of $N = 16^2$ points, while the remaining 15^2 correspond to the validation/test subset. According to equation (3), the original observations of the training set are corrupted by inlier noise originating from $\mathcal{N}(0, 3^2)$ and outlier values ± 40. The results are given in Table III for various fractions of outliers, with the best values of the MSE marked in **bold**. It is evident that, for the 2-dimensional non-linear denoising task, KGARD's performance outperforms its competitors (in terms of MSE), for all fractions of the outliers.

Finally, it should also be noted that, although RB-RVM does not perform at the highest level, it has the advantage that needs no tuning of parameters, albeit at substantially increased computational cost. On the contrary, the pair of tuning parameters for RAM, renders the method very difficult to be fully optimized (in terms of MSE), in practise. In contrast, taking into account the physical interpretation of ϵ and λ associated with KGARD, in the noise denoising task, we have developed a method for automatic user-free choice of these variables.

VII. Application in Image Denoising

In this section, in order to test the capabilities and verify the performance of the proposed algorithmic scheme, we use the KGARD framework to address one of the most popular problems that rise in the field of image processing: the task of removing noise from a digital image. The source of noise in this case can be either errors of the imaging system itself, errors that occur due to limitations of the imaging system, or errors that are generated by the environment. Typically, the noisy image is modeled as follows: $g(x, x') = \underline{g}(x, x') + v(x, x')$, for $x, x' \in [0,1]$, where $\underline{g}$ is the original noise-free image and v the additive noise. Given the noisy image g, the objective of any image denoising method is to obtain an estimate of the original image $\underline{g}$. In most cases, we assume that the image noise is Gaussian additive, independent at each pixel, and independent of the signal intensity, or that it contains spikes or impulses (i.e., salt and pepper noise).

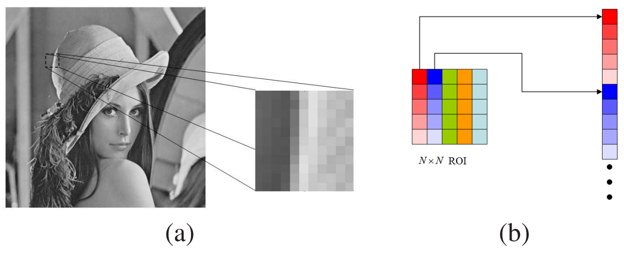

Fig. 2: (a) A square $N \times N$ region of intest (ROI). (b) Rearranging the pixels of a ROI.

However, there are cases where the noise model follows other probability density functions (e.g., the Poisson distribution or the uniform distribution). Although the wavelet-based image denoising methods dominate the research (see for example [34]–[36]), there are other methods that can be employed successfully, e.g., methods based on Partial Differential Equations, neighborhood filters, or methods of non linear modeling using local expansion approximation techniques [37]. The majority of the aforementioned methods assume a specific type of noise model. In fact, most of them require some sort of a priori knowledge of the noise distribution. In contrast to this approach, the more recently introduced denoising methods based on KRR make no assumptions about the underlying noise model and, thus, they can effectively treat more complex models, [17].

In this section, we demonstrate how the proposed KGARD algorithmic scheme can be used to treat the image denoising problem in cases where the noise model includes impulses. We will present two different denoising methods to deal with this type of noise. The first one is directly based on KGARD algorithmic scheme, while the second method splits the denoising procedure into two parts: the identification and removal of the impulses is first carried out, via the KGARD and then the output is fed into a cutting edge wavelet based denoising method to cope with the bounded noise component.

A. Modeling the Image and the Noise

In the proposed denoising method, we adopt the well known and popular strategy of dividing the "noisy" image into smaller $N \times N$ square regions of interest (ROIs), as it is illustrated in Figure 2. Then, we rearrange the pixels so that to form a row vector. Instead of applying the denoising process to the entire image, we process each ROI individually in sequential order. This is done for two reasons: (a) Firstly, the time needed to solve the optimization tasks considered in the next sections increases polynomially with N^2 and (b) working with each ROI separately enables us to change the parameters of the model in an adaptive manner, to account for the different level of details in each ROI. The rearrangement shown in Figure 2 implies that, the pixel (i, j) (i.e., i-th row, j-th column) is placed at the n-th position of the respective vector, where $n = (i-1) \cdot N + j$.

In KRR denoising methods, one assumes that each ROI represents the points on the surface of a continuous function, $\underline{g}$, of two variables defined on $[0,1] \times [0,1]$. The pixel values of the noise-free and the noisy digitized ROIs are represented as $\underset{\sim}{\zeta}_{ij} = \underset{\sim}{g}(x_i, x'_j)$ and ζ_{ij} respectively (both taking values in the interval $[0, 255]$), where $x_i = (i-1)/(N-1)$, $x'_j = (j-1)/(N-1)$, for $i, j = 1, 2, ..., N$. Moreover, as the original image $\underset{\sim}{g}$ is a relatively smooth function (with the exception close to the edges), we assume that it lies in an RKHS induced by the Gaussian kernel, i.e., $\underset{\sim}{g} \in \mathcal{H}$, for some $\sigma > 0$. Specifically, in order to be consistent with the representer theorem, we will assume that $\underset{\sim}{g}$ takes the form of a finite linear representation of kernel functions centered at all pixels, thus after pixel rearrangement we can write:

$$\underset{\sim}{g} = \sum_{n=1}^{N^2} \underset{\sim}{\alpha}_n \kappa(\cdot, \boldsymbol{x}_n), \tag{18}$$

where $\boldsymbol{x}_n = (x_i, x'_j)$ and $n = (i-1)\cdot N + j$. Hence, the intensity of the n-th pixel is given by

$$\underset{\sim}{\zeta}_n = \underset{\sim}{g}(\boldsymbol{x}_n) = \sum_{m=1}^{N^2} \underset{\sim}{\alpha}_m \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m). \tag{19}$$

The model considered in this paper assumes that the intensity of the pixels of the noisy ROI can be decomposed as $\zeta_{ij} = \underset{\sim}{\zeta}_{ij} + \underset{\sim}{u}_{ij} + \eta_{ij}$, for $i, j = 1, 2, ..., N$, where η_{ij} denotes the bounded noise component and $\underset{\sim}{u}_{ij}$ the possible appearance of an outlier at that pixel. In vector notation (after rearrangement), we can write $\boldsymbol{\zeta} = \underset{\sim}{\boldsymbol{\zeta}} + \underset{\sim}{\boldsymbol{u}} + \boldsymbol{\eta}$, where $\underset{\sim}{\boldsymbol{\zeta}}, \boldsymbol{\zeta}, \underset{\sim}{\boldsymbol{u}}, \boldsymbol{\eta}, \in \mathbb{R}^{N^2}$, $\|\boldsymbol{\eta}\|_2 \leq \epsilon$ and $\underset{\sim}{\boldsymbol{u}}$ is a sparse vector. Moreover, exploiting (19), we can write $\underset{\sim}{\boldsymbol{\zeta}} = \boldsymbol{K} \cdot \underset{\sim}{\boldsymbol{\alpha}}$, where $\kappa_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$. In this context, we can model the denoising task as the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{a}, \boldsymbol{u} \in \mathbb{R}^{N^2}, c \in \mathbb{R}} \quad & \|\boldsymbol{u}\|_0 \\ \text{s. t.} \quad & \|\boldsymbol{\zeta} - \boldsymbol{K}\boldsymbol{a} - c\boldsymbol{1} - \boldsymbol{u}\|_2^2 + \lambda\|\boldsymbol{a}\|_2^2 + \lambda c^2 \leq \varepsilon, \end{aligned} \tag{20}$$

for some predefined $\lambda, \varepsilon > 0$. In a nutshell, problem (20) solves for the sparsest outlier's vector $\boldsymbol{u}$ and the respective $\boldsymbol{a}$ (i.e., the coefficients of the kernel expansion) that keep the error low, while at the same time preserve the smoothness of the original noise-free ROI (this is done via the regularization of the constraint's inequality). The regularization parameter λ controls the smoothness of the solution. The larger the λ is, the smoother the solution becomes, i.e., $\hat{\underset{\sim}{\boldsymbol{\zeta}}} = \boldsymbol{K}\hat{\boldsymbol{\alpha}}$.

B. Implementation

The main mechanism of both algorithms that are presented in this section is simple. The image is divided into $N \times N$ ROIs and the KGARD algorithm is applied in each individual ROI sequentially. However, as the reconstruction accuracy drops near the borders of the respective domain, we have chosen to discard the values at those points. This means that although KGARD is applied to the $N \times N$ ROI, only the $L \times L$ values are used in the final reconstruction (those that are at the center of the ROI). In the sequel, we will name the $L \times L$ centered region as "reduced ROI" or rROI for short. We will also assume that the dimensions of the image are multipliers of L (if they are not, we can add dummy pixels to the end) and select N so that $N - L$ is an even number.

After the reconstruction of a specific rROI, the algorithm moves to the next one, i.e., it moves L pixels to the right, or, if the algorithm has reached the right end of the image, it moves at the beginning of the line, which is placed L pixels below. Observe that, for this procedure to be valid, the image has to be padded by adding $(N - L)/2$ pixels along all dimensions. In this paper, we chose to pad the image by repeating border elements[7]. For example, if we select $L = 8$ and $N = 12$ to apply this procedure on an image with dimensions[8] 32×32, we will end up with a total of 16 overlapping ROIs, 4 per line.

Another important aspect of the denoising algorithm is the automated selection of the parameters λ and ϵ, that are involved in KGARD. This is an important feature, as these parameters largely control both the quality of the estimation and the recovery of the outliers and have to be tuned for each specific ROI. Naturally, it would have been intractable to require a user pre-defined pair of values (i.e., λ, ϵ) for each specific ROI. Hence, we devised simple methods to adjust these values in each ROI depending on its features.

1) Automatic selection of the regularization parameter λ: This parameter controls the smoothing operation of the denoising process. The user enters a specific value for λ_0 to control the strength of the smoothening and then the algorithm adjusts this value at each ROI separately, so that λ is small at ROIs that contain a lot of "edges" and large at ROIs that contain smooth areas. Whether a ROI has edges or not is determined by the mean magnitude of the gradient at each pixel. The rationale is described below:

- Select a user-defined value λ_0.
- Compute the magnitude of the gradient at each pixel.
- Compute the mean gradient of each ROI, i.e., the mean value of the gradient's magnitude of all pixels that belong to the ROI.
- Compute the mean value, m, and the standard deviation, s, of the aforementioned mean gradients.
- ROIs with mean gradient larger than $m + s$ are assumed to be areas with fine details and the algorithm sets $\lambda = \lambda_0$.
- All ROIs with mean gradient lower than $m - s/10$ are assumed to be smooth areas and the algorithm sets $\lambda = 15\lambda_0$.
- For all other ROIs the algorithm sets $\lambda = 5\lambda_0$.

2) Automatic computation of the termination parameter ϵ: In the image denoising case, the stopping criterion of KGARD is slightly modified. Hence, instead of requiring the norm of the residual vector to drop below ϵ, i.e., $\|\boldsymbol{r}_{(k)}\|_2 \leq \epsilon$, we require the maximum absolute valued coordinate of $\boldsymbol{r}_{(k)}$ to drop below ϵ ($\|\boldsymbol{r}_{(k)}\|_\infty \leq \epsilon$). The estimation of ϵ for each particular ROI is carried out as follows. Initially, a user defined parameter E_0 is selected. At each step, a histogram chart with elements $|r_{(k),i}|$ is generated, using $\left\lceil \frac{N^2}{10} \right\rceil + 1$ equally spaced bins along the x-axis, between the minimum and maximum values of $|r_{(k),i}|$. Let $\boldsymbol{h}$ denote the heights of the bars of the histogram and h_m be the minimum height of the histogram bars. Next, two real numbers, i.e., E_1, E_2, are defined. In particular, the number E_1 represents the left endpoint of the first occurrence of a minimum-height bar (i.e., the first bar with height equal to h_m, moving from left to right). The number E_2 represents the left endpoint of the first bar, ℓ, with height h_ℓ (moving from left to right) that satisfies both $h_\ell - h_{\ell-1} \geq 1$ and

[7]This can be done with the "replicate" option of MatLab's function *padarray*.

[8]Observe that L divides 32.

Algorithm 2 KGARD for image denoising

```
1: Input: the original noisy image $\boldsymbol{I}$ and $\lambda_0$, $\sigma$, $E_0$, $N$, $L$.
2: Output: the denoised image $\hat{\boldsymbol{I}}$ and the outliers' image $\hat{\boldsymbol{O}}$.
3: Build the kernel matrix $\boldsymbol{K}$.
4: if the dimensions of the original image are not multiplies of $L$ then
5:     Add initial padding
6: Form $\hat{\boldsymbol{I}}$ and $\hat{\boldsymbol{O}}$ to the same dimensions as $\boldsymbol{I}$.
7: Add padding with size $N - L$ around the image.
8: Divide the image into $N \times N$ ROIs and compute the regularization
   parameters for each ROI.
9: for each ROI $\boldsymbol{R}$ do
10:     Rearrange the pixels of $\boldsymbol{R}$ to form the vector $\boldsymbol{\zeta}$.
11:     Run the modified KGARD algorithm on the set $\boldsymbol{\zeta}$ with parameter $\lambda$
        and stoping criterion as described in section VII-B2.
12:     Let $\hat{\boldsymbol{a}}$, $\hat{\boldsymbol{u}}$ be the solution of KGARD.
13:     Compute the denoised vector $\hat{\boldsymbol{\zeta}} = \boldsymbol{K}\hat{\boldsymbol{a}}$.
14:     Rearrange the elements of $\hat{\boldsymbol{\zeta}}$ to form the denoised ROI $\hat{\boldsymbol{R}}$.
15:     Extract the centered $L \times L$ rROI from $\hat{\boldsymbol{R}}$.
16:     Use the values of the rROI to set the values of the corresponding
        pixels in $\hat{\boldsymbol{I}}$.
17:     Rearrange the elements of $\hat{\boldsymbol{u}}$ to form the outliers' ROI.
18:     Extract the centered $L \times L$ values of the outliers' ROI.
19:     Use these values to set the values of $\hat{\boldsymbol{O}}$.
20:     Move to the next ROI.
21: Remove the initial padding on $\hat{\boldsymbol{I}}$ and $\hat{\boldsymbol{O}}$ (if needed).
```

$h_{\ell-1} \leq h_m + 5,\ \ell \geq 2$. This roughly corresponds to the first increasing bar, which in parallel is next to a bar with height close to the minimum height. Both E_1 and E_2 are reasonable choices for the value of ϵ (meaning that the bars to the right of these values may be assumed to represent outliers)[9]. Finally, the algorithm determines whether the histogram can be clearly divided into two parts; the first one represents the usual errors and the other the errors due to outliers by using a simple rule: if $\frac{\sqrt{\text{var}(\boldsymbol{h}_{(k)})}}{\text{mean}(\boldsymbol{h}_{(k)})} > 0.9$, then the two areas can be clearly distinguished, otherwise it is harder to separate these areas. Note that, we use the notation $\boldsymbol{h}_{(k)}$ to refer to the heights of the histogram bar at the k step of the algorithm. The final computation of ϵ (at step k) is carried out as follows:

$$\epsilon_{(k)} = \begin{cases} \min\{E_0, E_1, E_2\}, & \text{if } \frac{\sqrt{\text{var}(\boldsymbol{h}_{(k)})}}{\text{mean}(\boldsymbol{h}_{(k)})} > 0.9 \\ \min\{E_0, E_1\}, & \text{otherwise.} \end{cases} \tag{21}$$

It should be noted that, the user defined parameter E_0 has little importance in the evaluation of ϵ. One may set it constantly to a value near 40 (as we did in all provided simulations). However, in cases where the image is corrupted by outliers only, a smaller value may be advisable, although it does not have a great impact on the reconstruction quality.

3) Direct KGARD implementation: The first denoising method, which we call KGARD for short, is described in Algorithm 2. The algorithm requires five user-defined parameters: (a) the regularization parameter, λ_0, (b) the Gaussian kernel width, σ, (c) the OMP termination parameter ϵ, (d) the size of the ROI, N and (e) the size of the rROIs, that are used in the reconstruction, i.e., L. However, these parameters are somehow interrelated. We will discuss these issues in the next sections.

4) KGARD combined with BM3D (KG-BM3D): This is a two-step procedure, that combines the outliers detection properties of KGARD with the denoising capabilities of a standard off-the-shelf denoising method. In this setting (which is the one we propose), the KGARD is actually used to detect the outliers and remove them, while the BM3D wavelet-based denoising method [36] takes over afterwards to cope with the bounded noise. Hence, the KGARD algorithm is firstly applied onto the noisy image, to obtain the positions and values of the reconstructed outliers, which are then subtracted from the original noisy image and BM3D is applied to the result. This method requires the same parameters as KGARD, plus the parameter s, which is needed by the BM3D algorithm[10].

C. Parameter Selection

This section is devoted on providing guidelines for the selection of the user-defined parameters for the proposed denoising algorithms. Typical values of N range between 8 and 16. Values of N near 8, or even lower, increase the time required to complete the denoising process with no significant improvements in most cases. However, if the image contains a lot of "fine details" this may be advisable. In these cases, smaller values for the width of the Gaussian kernel, σ, may also enhance the performance, since in this case the regression task is more robust to abrupt changes. However, we should note that σ is inversely associated with the size[11] of the ROI, N, hence if one increases N, one should decrease σ proportionally, i.e., keeping the product $N \cdot \sigma$ constant. We have observed that the values $N = 12$ and $\sigma = 0.3$ (which result to a product equal to $N \cdot \sigma = 3.6$) are adequate to remove moderate noise from a typical image. In cases where the image is rich in details and edges, N and σ should be adjusted to provide a lower product (e.g., $N = 12$ and $\sigma = 0.15$, so that $N \cdot \sigma = 1.8$). For images corrupted by high noise, this product should become larger. Finally, λ controls the importance of regularization on the final result. Large values imply a strong smoothing operation, while small values (close to zero) reduce the effect of regularization leading to a better fit; however, it may lead to overfitting.

For the experiments presented in this paper, we fixed the size of the ROIs using $N = 12$ and $L = 8$. These are reasonable choices that provide fast results with high reconstruction accuracy. Hence, only the values for σ and λ_0 need to be adjusted according to the density of the details in the image and the amount of noise. We have found that the values of σ that provide adequate results range between 0.1 and 0.4. Similarly, typical values of λ_0 range from 0.1 to 1. Finally, the constant E_0 was set equal to 40 for all cases.

The parameter s of the BM3D method is adjusted according to the amount of noise presented in the image. It ranges between very small values (e.g, 5), when only a small amount of bounded noise is present, to significantly larger values (e.g., 20 or 40) if the image is highly corrupted.

[9]More details can be found in [28].

[10]BM3D is built upon the assumption that the image is corrupted by Gaussian noise. Hence, the parameter s is the variance of that Gaussian noise, if this is known a-priori, or some user-defined estimate. However, it has been demonstrated that BM3D can also efficiently remove other types of noise, if s is adjusted properly [17].

[11]For example, if $N = 12$ and $\sigma = 0.3$, then the kernel width is equal to 3.6 pixels. It is straightforward to see that, if N decreases to say 8, then the kernel width that will provide a length of 3.6 pixels is $\sigma = 0.45$.

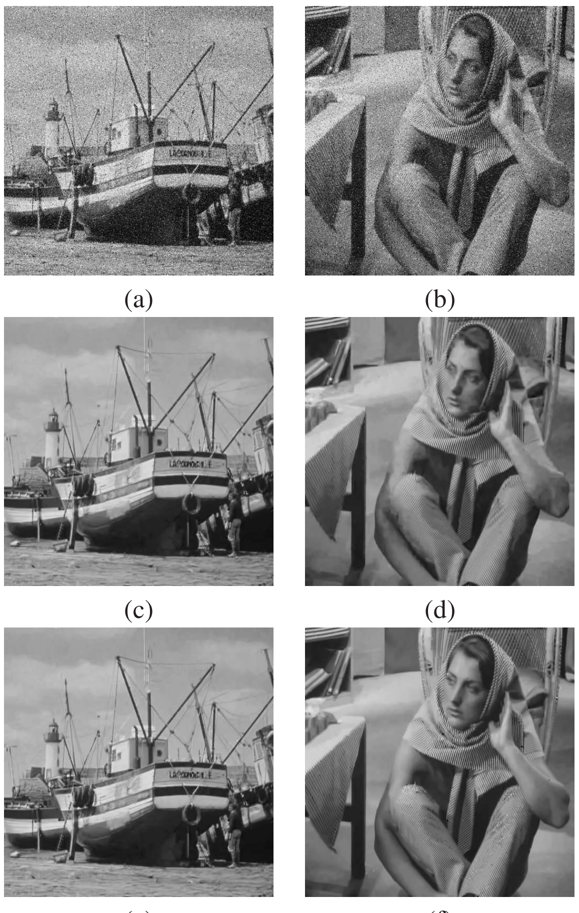

Fig. 3: (a),(b) The *boat* and *Barbara* images corrupted by 20 dB of Gaussian noise and 10% outliers. (c), (d) Denoising with BM3D (28.97 dB and 29.2 dB). (d), (e) Denoising with joint KG-BM3D (31.52 dB and 30.43 dB).

D. Experiments on Images Corrupted by Synthetic Noise

In this section, we present a set of experiments on grayscale images that have been corrupted by mixed noise, which comprises a Gaussian component and a set of impulses (± 100). The intensity of the Gaussian noise has been ranged between 15 dB and 25 dB and the percentage of impulses from 5% to 20%. The tests were performed on three very popular images: the *Lena*, the *boat* and the *Barbara* images, that are included in Waterloo's image repository. Each test has been performed 50 times and the respective mean PSNRs are reported. The parameters have been tuned so that to provide the best result (in terms of MSE). In Table IV, the two proposed methods are applied to the *Lena* image and they are compared with BM3D (the state of the art wavelet-based method) and an image denoising method based on (RB-RVM) ("G. N." stands for Gaussian Noise and "Imp." for Impulses). For the latter, we chose a simple implementation, similar to the one we propose in our methods: the image is divided into ROIs and the RB-RVM algorithm is applied to each ROI sequentially. The parameters were selected to provide the best possible results in terms of PSNR. The size of the ROIs has been set to $N = 12$ and $L = 8$ for the Lena and *boat* image. As the *Barbara* image has more finer details (e.g., the stripes of the pants) we have set $N = 12$ and $L = 4$ for this image. Moreover, one can observe that for this image, we have used a lower value for σ and λ as indicated in Section VII-C. Figure 3 demonstrates the obtained denoised images on a specific experiment (20 dB Gaussian noise and 10% outliers). It is clear that the proposed method (KG-BM3D) enhances significantly the denoising capabilities of BM3D, especially for low and moderate intensities of the Gaussian noise. If the Gaussian component becomes prominent (e.g., at 15 dB) then the two methods provide similar results. Regarding the computational load, it only takes a few seconds in a standard PC for each one of the two methods to complete the denoising process.

Finally, it is noted that we chose not to include RAM or any ℓ_1-based denoising method, as this would require efficient techniques to adaptively control its parameters, i.e., λ, μ at each ROI (similar to the case of KGARD), which remains an open issue. Having to play with both parameters, makes the tuning computationally demanding. This is because the number of iterations for the method to converge to a reasonable solution increases substantially, once the parameters are moved away from their optimal (in terms of MSE) values[12].

Appendix

Appendix A. Proof of Theorem 1

Proof. Our analysis is based on the *singular value decomposition* (SVD) for matrix $\boldsymbol{X}_{(0)} = [\boldsymbol{K}\ \boldsymbol{1}]$. Since matrix $\boldsymbol{X}_{(0)}^T\boldsymbol{X}_{(0)}$ is positive semi-definite, all of its eigenvalues are non-negative. Let $\boldsymbol{X}_{(0)} = \boldsymbol{Q}\boldsymbol{S}\boldsymbol{V}^T$, where $\boldsymbol{Q}, \boldsymbol{V}$ are orthogonal and $\boldsymbol{S}$ is the matrix of dimension $N \times (N+1)$ of the form $\boldsymbol{S} = \begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{0}\end{bmatrix}$ and $\boldsymbol{\Sigma}$ is a diagonal matrix with entries $\sigma_i \geq 0,\ i = 1, ..., N$. For simplification, the notation σ_M will be used to denote the maximum singular value of matrix $\boldsymbol{X}_{(0)}$.

The proposed method attempts to solve at each step, the regularized Least Squares (LS) task (10) for the selection of matrix $\boldsymbol{B}$. The latter task is equivalent to a LS problem in the augmented space[13] at each k-step, i.e., in (13), where $\boldsymbol{X}_{(k)} = \begin{bmatrix}\boldsymbol{X}_{(k-1)} & \boldsymbol{e}_{j_k}\end{bmatrix}$ and $\boldsymbol{B}_{(k)} = \begin{bmatrix}\boldsymbol{B}_{(k-1)} & \boldsymbol{0} \\ \boldsymbol{0}^T & 0\end{bmatrix}$. Thus, the LS solution at each k-step could be expressed as:

$$\hat{\boldsymbol{z}}_{(k)} = (\boldsymbol{X}_{(k)}^T\boldsymbol{X}_{(k)} + \lambda\boldsymbol{B}_{(k)})^{-1}\boldsymbol{X}_{(k)}^T\boldsymbol{y} \tag{22}$$

and the respective residual is

$$\boldsymbol{r}_{(k)} = \boldsymbol{y} - \boldsymbol{X}_{(k)}\hat{\boldsymbol{z}}_{(k)} = \boldsymbol{y} - \boldsymbol{X}_{(k)}(\boldsymbol{X}_{(k)}^T\boldsymbol{X}_{(k)} + \lambda\boldsymbol{B}_{(k)})^{-1}\boldsymbol{X}_{(k)}^T\boldsymbol{y}. \tag{23}$$

Step $k = 0$:
Initially, $\boldsymbol{B}_{(0)} = \boldsymbol{I}_{N+1}$ and $\mathcal{S}_0 = \{1, \dots, N+1\}$ (no index has been selected for the outlier estimate), thus $\boldsymbol{X}_{(0)} = [\boldsymbol{K}\ \boldsymbol{1}]$. Hence, the expression for the initial LS solution $\hat{\boldsymbol{z}}_{(0)}$ is obtained from equation (23) for $k = 0$. Employing the SVD decomposition for matrix $\boldsymbol{X}_{(0)}$, we have

$$\boldsymbol{X}_{(0)}^T\boldsymbol{X}_{(0)} + \lambda\boldsymbol{I}_{N+1} = \boldsymbol{V}\underbrace{\begin{bmatrix}\boldsymbol{\Sigma}^2 + \lambda\boldsymbol{I}_N & \boldsymbol{0} \\ \boldsymbol{0}^T & \lambda\end{bmatrix}}_{\boldsymbol{\Lambda}}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T. \tag{24}$$

[12] If the parameters are not optimally tuned, the denoising process may take more than an hour to complete in MATLAB on a moderate computer.

[13] This is due to the fact that $\boldsymbol{B}$ is a projection matrix (based on the ℓ_2 regularization model).

Method	Parameters	G. N.	Imp.	PSNR
BM3D	$s = 30$	25 dB	5%	32.2 dB
RB-RVM	$\sigma = 0.3$	25 dB	5%	31.78 dB
KGARD	$\sigma = 0.3,\ \lambda = 1$	25 dB	5%	33.91 dB
KG-BM3D	$\sigma = 0.3,\ \lambda = 1,\ s = 10$	25 dB	5%	**36.2 dB**
BM3D	$s = 30$	25 dB	10%	30.84 dB
RB-RVM	$\sigma = 0.3$	25 dB	10%	31.25 dB
KGARD	$\sigma = 0.3,\ \lambda = 1,\ \epsilon = 40$	25 dB	10%	33.49 dB
KG-BM3D	$\sigma = 0.3,\ \lambda = 1,\ s = 10$	25 dB	10%	**35.67 dB**
BM3D	$s = 30$	20 dB	5%	31.83 dB
RB-RVM	$\sigma = 0.4$	20 dB	5%	29.3 dB
KGARD	$\sigma = 0.3,\ \lambda = 1$	20 dB	5%	32.35 dB
KG-BM3D	$\sigma = 0.3,\ \lambda = 1,\ s = 15$	20 dB	5%	**34.24 dB**
BM3D	$s = 35$	20 dB	10%	30.66 dB
RB-RVM	$\sigma = 0.4$	20 dB	10%	29.09 dB
KGARD	$\sigma = 0.3,\ \lambda = 1$	20 dB	10%	31.94 dB
KG-BM3D	$\sigma = 0.3,\ \lambda = 1,\ s = 15$	20 dB	10%	**33.81 dB**
BM3D	$s = 35$	15 dB	5%	30.87 dB
RB-RVM	$\sigma = 0.6$	15 dB	5%	26.74 dB
KGARD	$\sigma = 0.3,\ \lambda = 1.5$	15 dB	5%	29.12 dB
KG-BM3D	$\sigma = 0.3,\ \lambda = 1,\ s = 25$	15 dB	5%	**31.18 dB**
BM3D	$s = 40$	15 dB	10%	29.94 dB
RB-RVM	$\sigma = 0.4$	15 dB	10%	25.85 dB
KGARD	$\sigma = 0.3,\ \lambda = 2$	15 dB	10%	28.47 dB
KG-BM3D	$\sigma = 0.3,\ \lambda = 1,\ s = 25$	15 dB	10%	**30.77 dB**

TABLE IV: Denoising performed on the *Lena* image corrupted by various types and intensities of noise using the proposed methods, the RB-RVM approach and the state of the art wavelet method BM3D.

Combining (23) for $k = 0$ with (24), leads to

$$\boldsymbol{r}_{(0)} = \boldsymbol{y} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{y}, \tag{25}$$

where $\boldsymbol{G} = \boldsymbol{\Sigma}(\boldsymbol{\Sigma}^2 + \lambda\boldsymbol{I}_N)^{-1}\boldsymbol{\Sigma}$ is a diagonal matrix with entries $g_{ii} = \frac{\sigma_i^2}{\sigma_i^2+\lambda}$, $i = 1, 2, ..., n$. Furthermore, substituting $\boldsymbol{y} = \boldsymbol{X}_{(0)}\underline{\boldsymbol{\theta}} + \underline{\boldsymbol{u}}$ in (25) leads to

$$\boldsymbol{r}_{(0)} = \underline{\boldsymbol{u}} + \boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}, \tag{26}$$

where $\boldsymbol{F} = \boldsymbol{S} - \boldsymbol{G}\boldsymbol{S} = [\underbrace{\boldsymbol{\Sigma} - \boldsymbol{G}\boldsymbol{\Sigma}}_{\boldsymbol{\Phi}}\ \ \boldsymbol{0}]$. Matrix $\boldsymbol{\Phi}$ is also diagonal, with values $\phi_{ii} = \frac{\lambda\sigma_i}{\sigma_i^2+\lambda}$, $i = 1, 2, ..., N$. At this point it is required to explore some of the unique properties of matrices $\boldsymbol{G}$ and $\boldsymbol{F}$. Recall that the (matrix) 2-norm of a diagonal matrix is equal to the maximum absolute value of the diagonal entries. Hence, it is clear that

$$||\boldsymbol{G}||_2 = \sigma_M^2/(\sigma_M^2 + \lambda) \text{ and } ||\boldsymbol{F}||_2 = ||\boldsymbol{\Phi}||_2 \leq \sqrt{\lambda}/2, \tag{27}$$

since $g(\sigma) = \frac{\sigma^2}{\sigma^2+\lambda}$ is a strictly increasing function of $\sigma \geq 0$ and $\phi(\sigma) = \frac{\lambda\sigma}{\sigma^2+\lambda}$ receives a unique maximum, which determines the upper bound for the matrix 2-norm.

Finally, it should be noted that if no outliers exist in the noise,the algorithm terminates due to the fact that the norm of the initial residual is less than (or equal to) ϵ. However, this scenario is rather insignificant since no robust modeling is required. Thus, if our goal is for the method to be able to handle various types of noise that includes outliers (e.g. Gaussian noise plus impulses), we assume that $\|\boldsymbol{r}_{(0)}\|_2 > \epsilon$. In such a case KGARD identifies an outlier selecting an index from the set $\tilde{\mathcal{S}}_0^c = \{1, 2, ..., N\}$.

At the first selection step, as well as at every next step, we should impose a condition so that the method identifies and selects an index that belongs to the support of the sparse outlier vector. To this end, let $\mathcal{T}$ denote the support of the sparse outlier vector $\underline{\boldsymbol{u}}$. In order for KGARD to select a column $\boldsymbol{e}_i$ from matrix $\boldsymbol{I}_N$ that belongs to $\mathcal{T}$, we should impose

$$|r_{(0),i}| > |r_{(0),j}|, \text{ for all } i \in \mathcal{T} \text{ and } j \in \mathcal{T}^c. \tag{28}$$

The key is to establish appropriate bounds, which guarantee the selection of a correct index that belongs to $\mathcal{T}$. Therefore, we first need to develop bounds on the following inner products. Using (27), the Cauchy-Schwarz inequality and the fact that $\boldsymbol{Q}, \boldsymbol{V}$ are orthonormal, it is easy to verify that

$$|\langle e_l, \boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}}\rangle| \leq \frac{\sqrt{\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 \tag{29}$$

as well as $$|\langle e_l, \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}\rangle| \leq \frac{\sigma_M^2}{\sigma_M^2 + \lambda}\|\underline{\boldsymbol{u}}\|_2, \tag{30}$$

for all $l = 1, 2, ..., N$. Thus, for any $i \in \mathcal{T}$, we have that

$$|r_{(0),i}| > \min|\underline{u}| - \frac{\sqrt{2\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 - \frac{\sigma_M^2}{\sigma_M^2 + \lambda}\|\underline{\boldsymbol{u}}\|_2, \tag{31}$$

and $$|r_{(0),j}| < \frac{\sqrt{2\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 + \frac{\sigma_M^2}{\sigma_M^2 + \lambda}\|\underline{\boldsymbol{u}}\|_2, \tag{32}$$

for all $j \in \mathcal{T}^c$, where equation (26) and inequalities (29) and (30) have also been used. Hence, imposing (28) leads to (16). It should be noted that, a reason that could lead to the violation of (16) is for the term $\min|\underline{u}| - \sqrt{2\lambda}\|\underline{\boldsymbol{\theta}}\|_2$ to be non-positive. Thus, since the regularization parameter is fine-tuned by the user, we should select $\lambda < (\min|\underline{u}|/\|\underline{\boldsymbol{\theta}}\|_2)^2/2$. If the condition is guaranteed, then at the first selection step, a column indexed $j_1 \in \mathcal{T}$ is selected. The set of active columns that participates in the LS solution of the current step is then $\mathcal{S}_1 = \{j_1\} \subseteq \mathcal{T}$ and thus $\boldsymbol{X}_{(1)} = \begin{bmatrix}\boldsymbol{X}_{(0)} & \boldsymbol{e}_{j_1}\end{bmatrix}$ and $\boldsymbol{B}_{(1)} = \begin{bmatrix}\boldsymbol{I}_{N+1} & \boldsymbol{0} \\ \boldsymbol{0}^T & 0\end{bmatrix}$.

General k **step**:
At the k step, $\mathcal{S}_k = \{j_1, j_2, ..., j_k\} \subset \mathcal{T}$ and thus $\boldsymbol{X}_{(k)} = \begin{bmatrix}\boldsymbol{X}_{(0)} & \boldsymbol{I}_{\mathcal{S}_k}\end{bmatrix}$ and $\boldsymbol{B}_{(k)} = \begin{bmatrix}\boldsymbol{I}_{N+1} & \boldsymbol{O}_{(N+1)\times k} \\ \boldsymbol{O}^T_{(N+1)\times k} & \boldsymbol{O}_k\end{bmatrix}$. After the selection of the first column, the LS step requires the inversion of the matrix

$$\boldsymbol{D}_{(k)}^T\boldsymbol{D}_{(k)} = \begin{bmatrix}\boldsymbol{X}_{(0)}^T\boldsymbol{X}_{(0)} + \lambda\boldsymbol{I}_{N+1} & \boldsymbol{X}_{(0)}^T\boldsymbol{I}_{\mathcal{S}_k} \\ \boldsymbol{I}_{\mathcal{S}_k}^T\boldsymbol{X}_{(0)} & \boldsymbol{I}_k\end{bmatrix}.$$

By applying the *Matrix inversion Lemma* to $\boldsymbol{D}_{(k)}^T\boldsymbol{D}_{(k)}$ combined with (24) and then substituting into (23) leads to:

$$\boldsymbol{r}_{(k)} = \boldsymbol{P}_{(k)}\underline{\boldsymbol{u}} + \boldsymbol{P}_{(k)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} - \boldsymbol{P}_{(k)}\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}, \tag{33}$$

where $\boldsymbol{P}_{(k)} = \boldsymbol{I}_N + \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{I}_{\mathcal{S}_k}\boldsymbol{W}_{(k)}^{-1}\boldsymbol{I}_{\mathcal{S}_k}^T - \boldsymbol{I}_{\mathcal{S}_k}\boldsymbol{W}_{(k)}^{-1}\boldsymbol{I}_{\mathcal{S}_k}^T$ and $\boldsymbol{W}_{(k)} = \boldsymbol{I}_k - \boldsymbol{I}_{\mathcal{S}_k}^T\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{I}_{\mathcal{S}_k}$. If we wish for the algorithm to select an index from the set $\mathcal{T}$, we should impose $|r_{(k),i}| > |r_{(k),j}|$, for all $i \in \mathcal{T}/\mathcal{S}_k, j \in \mathcal{T}^c$. Now $\boldsymbol{P}_{(k)}(\underline{\boldsymbol{u}} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}) = \boldsymbol{u}_{(k)} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{u}_{(k)}$, where $\boldsymbol{u}_{(k)} = \underline{\boldsymbol{u}}_{\mathcal{T}/\mathcal{S}_k} + \boldsymbol{I}_{\mathcal{S}_k}\boldsymbol{W}_{(k)}^{-1}\boldsymbol{I}_{\mathcal{S}_k}^T\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}_{\mathcal{T}/\mathcal{S}_k}$. Hence, the final form of the residual is:

$$\boldsymbol{r}_{(k)} = \boldsymbol{u}_{(k)} + \boldsymbol{P}_{(k)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{u}_{(k)}. \tag{34}$$

For $l \notin \mathcal{S}_k$, we conclude that

$$\boldsymbol{P}_{(k)}^T e_l = e_l + \boldsymbol{I}_{\mathcal{S}_k}\boldsymbol{W}_{(k)}^{-1}\boldsymbol{I}_{\mathcal{S}_k}^T\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T e_l,$$

is a $(k+1)$-sparse vector. Furthermore, it is readily seen that,

$$\left\|\boldsymbol{W}_{(k)}^{-1}\boldsymbol{I}_{\mathcal{S}_k}^T\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T e_l\right\|_2 \leq \frac{\sigma_M^2}{\lambda} < 1, \tag{35}$$

which leads to $\left\|\boldsymbol{P}_{(k)}^{T}\boldsymbol{e}_l\right\|_2 < \sqrt{2}$. Moreover,

$$|\langle \boldsymbol{e}_l, \boldsymbol{P}_{(k)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}}\rangle| < \frac{\sqrt{2\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 \text{ and } \|\boldsymbol{u}_{(k)}\|_2 < \|\underline{\boldsymbol{u}}\|_2. \tag{36}$$

Accordingly, the bounds for the residual are now expressed as

$$|r_{(k),i}| > \min|\underline{u}| - \frac{\sqrt{2\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 - \frac{\sigma_M^2}{\sigma_M^2+\lambda}\|\underline{\boldsymbol{u}}\|_2\,, \tag{37}$$

for any $i \in \mathcal{T}/\mathcal{S}_k$, and

$$|r_{(k),j}| < \frac{\sqrt{2\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 + \frac{\sigma_M^2}{\sigma_M^2+\lambda}\|\underline{\boldsymbol{u}}\|_2\,, \tag{38}$$

for all $j \in \mathcal{T}^c$, where (34) and (36) are used. Finally, imposing the lower bound of (37) to be greater than the upper bound of (38) leads to the condition (16). At the k step, it has been proved that unless the residual length is below the predefined threshold the algorithm will select another correct atom from the identity matrix and the procedure will repeat until $\mathcal{S}_k = \mathcal{T}$. At this point, KGARD has correctly identified all possible outliers and it is up to the tuning of the ϵ parameter whether the procedure terminates (and thus no extra indices are classified as outliers) or it continues and models other extra samples as outliers. □

References

[1] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*. Academic Press, 2015.

[2] P. J. Huber, *Wiley Series in Probability and Mathematics Statistics*. Wiley Online Library, 1981.

[3] R. A. Maronna, R. D. Martin, and V. J. Yohai, *Robust statistics*. J. Wiley, 2006.

[4] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*. John Wiley & Sons, 2005, vol. 589.

[5] P. J. Huber, "The 1972 wald lecture robust statistics: A review," *The Annals of Mathematical Statistics*, pp. 1041–1067, 1972.

[6] P. J. Rousseeuw and B. C. Van Zomeren, "Unmasking multivariate outliers and leverage points," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 633–639, 1990.

[7] A. M. Leroy and P. J. Rousseeuw, "Robust regression and outlier detection," *J. Wiley&Sons, New York*, 1987.

[8] S. A. Razavi, E. Ollila, and V. Koivunen, "Robust greedy algorithms for compressed sensing," in *Signal Processing Conference (EUSIPCO), Proceedings of the 20th European*. IEEE, 2012, pp. 969–973.

[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, 2011.

[10] S. Boyd, "Alternating direction method of multipliers," in *Talk at NIPS Workshop on Optimization and Machine Learning*, 2011.

[11] D. P. Wipf and B. D. Rao, "Sparse bayesian learning for basis selection," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2153–2164, 2004.

[12] Y. Jin and B. D. Rao, "Algorithms for robust linear regression by exploiting the connection to sparse signal recovery," in *International Conference on Acoustics Speech and Signal Processing (ICASSP)*. IEEE, 2010, pp. 3830–3833.

[13] G. Papageorgiou, P. Bouboulis, and S. Theodoridis, "Robust linear regression analysis-a greedy approach," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 3872–3887, 2014.

[14] G. Mateos and G. B. Giannakis, "Robust nonparametric regression via sparsity control with application to load curve data cleansing," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1571–1584, 2012.

[15] K. Mitra, A. Veeraraghavan, and R. Chellappa, "Robust RVM regression using sparse outlier model," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 1887–1894.

[16] P. Bouboulis and S. Theodoridis, "Kernel methods for image denoising," in *Regularization, optimization, kernels, and support vector machines*, J. Suykens, M. Signoretto, and A. Argyriou, Eds., 2015.

[17] P. Bouboulis, K. Slavakis, and S. Theodoridis, "Adaptive kernel-based image denoising employing semi-parametric regularization," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1465–1479, 2010.

[18] Y. C. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Conference on Signals, Systems and Computers, Conference Record of The Twenty-Seventh Asilomar*. IEEE, 1993, pp. 40–44.

[19] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[20] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[21] A. J. Smola and B. Schölkopf, *Learning with Kernels*. The MIT Press, 2002.

[22] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[23] K. Slavakis, P. Bouboulis, and S. Theodoridis, *Signal Processing Theory and Machine Learning: Online Learning in Reproducing Kernel Hilbert Spaces*, ser. Academic Press Library in Signal Processing, 2014, ch. 17.

[24] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, 4th Edition*. Academic press, 2008.

[25] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, vol. 68, no. 3, pp. 337–404, May 1950.

[26] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted 1 minimization," *Journal of Fourier analysis and applications*, vol. 14, no. 5-6, pp. 877–905, 2008.

[27] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *The journal of machine learning research*, vol. 1, pp. 211–244, 2001.

[28] G. Papageorgiou, P. Bouboulis, and S. Theodoridis, "Robust nonlinear regression analysis: A greedy approach employing kernels and application to image denoising," January 2016. [Online]. Available: http://arxiv.org/abs/1601.00595

[29] B. L. Sturm and M. G. Christensen, "Comparison of orthogonal matching pursuit implementations," in *Signal Processing Conference (EUSIPCO), Proceedings of the 20th European*. IEEE, 2012, pp. 220–224.

[30] W. Gander, *On the linear least squares problem with a quadratic constraint*. Computer Science Department, Stanford University, 1978.

[31] ——, "Least squares with a quadratic constraint," *Numerische Mathematik*, vol. 36, no. 3, pp. 291–307, 1980.

[32] Å. Björck, *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics, 1996, no. 51.

[33] M. Rojas and D. C. Sorensen, "A trust-region approach to the regularization of large-scale discrete forms of ill-posed problems," *SIAM Journal on Scientific Computing*, vol. 23, no. 6, pp. 1842–1860, 2002.

[34] J. Portilla, V. Strela, M. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Transactions on Image Processing*, vol. 12, no. 11, pp. 1338–1351, 2003.

[35] L. Sendur and I. Selesnick, "Bivariate shrinkage functions for wavelet-based denoising exploiting interscale dependency," *IEEE Transactions on Signal Processing*, vol. 50, no. 11, pp. 2744–2756, 2002.

[36] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3d transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[37] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Tranactions on Image Processing*, vol. 16, no. 2, pp. 349–366, 2007.